Taking Advantage of Symmetries: Gathering of Asynchronous Oblivious Robots on a Ring*

Ralf Klasing[1], Adrian Kosowski[1,2], and Alfredo Navarra[3]

[1] LaBRI - Université Bordeaux 1 - CNRS, 351 cours de la Liberation, 33405 Talence cedex, France
{Ralf.Klasing,Adrian.Kosowski}@labri.fr

[2] Department of Algorithms and System Modeling, Gdańsk University of Technology, Narutowicza 11/12, 80952 Gdańsk, Poland
kosowski@sphere.pl

[3] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy
navarra@dipmat.unipg.it

Abstract. One of the recently considered models of robot-based computing makes use of identical, memoryless mobile units placed in nodes of an anonymous graph. The robots operate in Look-Compute-Move cycles; in one cycle, a robot takes a snapshot of the current configuration (Look), takes a decision whether to stay idle or to move to one of the nodes adjacent to its current position (Compute), and in the latter case makes an instantaneous move to this neighbor (Move). Cycles are performed asynchronously for each robot.

In such a restricted scenario, we study the influence of symmetries of the robot configuration on the feasibility of certain computational tasks. More precisely, we deal with the problem of gathering all robots at one node of the graph, and propose a solution based on a symmetry-preserving strategy. When the considered graph is an undirected ring and the number of robots is sufficiently large (more than 18), such an approach is proved to solve the problem for all starting situations, as long as gathering is feasible. In this way we also close the open problem of characterizing symmetric situations on the ring which admit a gathering [R. Klasing, E. Markou, A. Pelc: Gathering asynchronous oblivious mobile robots in a ring, Theor. Comp. Sci. 390(1), 27-39, 2008].

The proposed symmetry-preserving approach, which is complementary to symmetry-breaking techniques found in related work, appears to be new and may have further applications in robot-based computing.

Keywords: Asynchronous system, Mobile robots, Oblivious robots, Gathering problem, Ring.

* The research was partially funded by the State Committee for Scientific Research (Poland) Grant 4 T11C 047 25, by the ANR-project "ALADDIN" (France), by the project "CEPAGE" of INRIA (France), and by European projects COST Action 293 "Graphs and Algorithms in Communication Networks" (GRAAL) and COST Action 295 "Dynamic Communication Networks" (DYNAMO).

T.P. Baker, A. Bui, and S. Tixeuil (Eds.): OPODIS 2008, LNCS 5401, pp. 446–462, 2008.

1 Introduction

The difficulty of many computational problems involving mobile entities (robots) is aggravated when robots cannot communicate directly, but have to take decisions about their moves only by observing the environment. One of the most restrictive scenarios considered in literature is the asynchronous Look-Compute-Move model for memoryless units which has been studied both for robots on the plane (the *continuous model* [13,20]) and for robots located on the nodes of a graph (the *discrete model* [10,11,16]). Herein we focus on computations in the discrete model which is described in more detail below.

1.1 The Discrete Model

Consider an anonymous graph in which neither nodes nor links have any labels. Initially, some of the nodes of the graph 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 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 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 finite but unbounded, and is decided by the adversary for each robot. The only constraint is that moves are instantaneous, and hence any robot performing a Look operation sees all other robots at nodes of the ring and not on edges. However, a robot r may perform a Look operation at some time t, perceiving robots at some nodes, then Compute a target neighbor at some time $t' > t$, and Move to this neighbor at some later time $t'' > t'$, at which some robots are in different nodes from those previously perceived by r because in the meantime they performed their Move operations. Hence, robots may move based on significantly outdated perceptions. It should be stressed that robots are memoryless (oblivious), i.e., they do not have any memory of past observations. Thus, the target node (which is either the current position of the robot or one of its neighbors) is decided by the robot during a Compute operation solely on the basis of the location of other robots perceived in the previous Look operation. Robots are anonymous and execute the same deterministic algorithm. They cannot leave any marks at visited nodes, nor send any messages to other robots.

It is assumed that the robots have the ability to perceive, during the Look operation, if there is one or more robots located at the given node of the graph. This capability of robots is important and well-studied in the literature on robot gathering under the name of *multiplicity detection* [13,20]. In fact, without this capability, many computational problems (such as the gathering problem considered herein) are impossible to solve for all non-trivial starting configurations. It should be stressed that, during a Look operation, a robot can only tell if at some node there are no robots, there is one robot, or there is more than one robot: a robot does not see the difference between a node occupied by a or b robots, for distinct $a, b > 1$.

Problems studied so far in the discrete model include gathering on the ring [16], exploration of the ring [10], and tree exploration [11].

1.2 Our Results

In this paper, we consider one of the most fundamental problems of self-organization of mobile entities, known in the literature as the *gathering* problem. Robots, initially situated at different locations, have to gather at the same location (not determined in advance) and remain in it. Our considerations focus on gathering robots in the discrete model for the undirected ring; such a scenario poses a number of problems due to the high number of potential symmetries of the robot configuration. This problem was initially studied in [16], where certain configurations were shown to be gatherable by means of symmetry-breaking techniques, but the question of the general-case solution was posed as an open problem. Herein we provide procedures for gathering all configurations on the ring with more than 18 robots for which gathering is feasible, and give a full characterization of all such configurations (Theorem 6). In fact, we provide a new technique for dealing with symmetric configurations: our approach is based on preserving symmetry rather than breaking it.

1.3 Related Work

The problem of gathering mobile robots in one location has been extensively studied in the literature. Many variations of this task have been considered in different computational models. Robots move either in a graph, cf. e.g. [2,8,9,12,17], or in the plane [1,3,4,5,6,7,13,19,20,21], they are labeled [8,9,17], or anonymous [1,3,4,5,6,7,13,19,20,21], gathering algorithms are probabilistic (cf. [2] and the literature cited there), or deterministic [1,3,4,5,6,7,8,12,13,17,19,20,21]. Deterministic algorithms for gathering robots in a ring (which is a task closest to our current setting) have been studied e.g. in [8,9,12,14,17]. In [8,9,17] symmetry was broken by assuming that robots have distinct labels, and in [12] it was broken by using tokens. The very weak assumption of anonymous identical robots was studied in [1,3,4,5,6,7,13,19,20,21] where robots could move freely in the plane. The scenario was further refined in various ways. In [4,14] it was assumed that robots have memory, while in [1,3,5,6,7,13,19,20,21] robots were oblivious, i.e., it was assumed that they do not have any memory of past observations. Oblivious robots operate in Look-Compute-Move cycles, similar to those described in our scenario. The differences are in the amount of synchrony assumed in the execution of the cycles. In [3,21] cycles were executed synchronously in rounds by all active robots, and the adversary could only decide which robots are active in a given cycle. In [4,5,6,7,13,19,20] they were executed asynchronously: the adversary could interleave operations arbitrarily, stop robots during the move, and schedule Look operations of some robots while others were moving. It was proved in [13] that gathering is possible in the asynchronous model if robots have the same orientation of the plane, even with limited visibility. Without orientation, the gathering problem was positively solved in [5], assuming that robots have the

capability of multiplicity detection. A complementary negative result concerning the asynchronous model was proved in [20]: without multiplicity detection, gathering robots that do not have orientation is impossible.

2 Terminology and Preliminaries

We consider an n-node anonymous ring without orientation. Initially, some nodes of the ring are occupied by robots and there is at most one robot in each node.

During a Look operation, a robot perceives the relative locations on the ring of multiplicities and single robots. We remind that a multiplicity occurs when more than one robot occupies the same location. For the purpose of the definition only, let us call one of the directions on the cycle *clockwise*, and the other *anti-clockwise*. Then, for a fixed robot r, let $S_C(r)$ denote the ordered sequence of distances from r to all single robots when traversing the cycle in the clockwise direction, and let $S_A(r)$ be the ordered sequence of such distances when moving anti-clockwise. Sets $M_C(r)$ and $M_A(r)$ are likewise defined for distances from r to all multiplicities. Then the *view* $V(r)$ provided to the robot r is defined as the set of ordered pairs $V(r) = \{(S_C(r), M_C(r)), (S_A(r), M_A(r))\}$. If there are no multiplicities, we will drop the second sequence in each case and write the view simply as the set of two sequences $V(r) = \{S_C(r), S_A(r)\}$.

The current configuration C 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, but disregarding the location of robot r; formally, $C = \{\{(S_C(r) \oplus i, M_C(r) \oplus i), (S_A(r) \ominus i, M_A(r) \ominus i)\} : i \in [1, n]\}$, where operations $\oplus$ and $\ominus$ denote modulo n addition and subtraction, respectively. Note that the configuration is independent of the choice of robot r and of the choice of the clockwise direction.

A configuration C is called *periodic* if it is invariable under rotation, i.e. $C = C \oplus k$ for some integer $k \in [1, n-1]$. A configuration C is called *symmetric* if the ring has a geometrical *axis of symmetry*, which reflects single robots into single robots, multiplicities into multiplicities, and empty nodes into empty nodes. Note that a symmetric configuration is not periodic if and only if it has exactly one axis of symmetry [16]. A symmetric configuration C with an axis of symmetry s has an *edge-edge symmetry* if s goes through (the middles of) two antipodal edges; it has a *node-on-axis symmetry* if at least one node is on the axis of symmetry.

A *pole* is an intersection point of a line with the ring (this may either be a node or in between two nodes). For configurations with a single axis of symmetry, nodes on the axis of symmetry are natural gathering points. The pole of the axis of symmetry used by the considered algorithm for gathering is known as the *North pole*, the other pole is called the *South pole*.

The set of nodes of the ring forming a path between two robots, excluding endpoints, is called an *arc*. Two robots are called *neighbors* if at least one of the two arcs of the ring between them does not contain any robots. When uniquely defined, the arc of the ring between two neighboring robots u, v with no robots on it is called the *gap* $u - v$. The length of gap $u - v$ is denoted as $|u - v|$, obviously $|u - v| = |v - u|$. Two robots forming a multiplicity are assumed to

form a gap of length 0. A gap of minimum length in a given configuration is simply called *minimal*.

The notation for gaps is extended to allow for *chains*, $u_1 - u_2 - \ldots - u_k$, i.e. sequences of robots separated by gaps. If some robots $u_i - \ldots - u_j$ form a multiplicity M, then the considered chain may be written compactly as $u_1 - \ldots - u_{i-1} - M - u_{j+1} - \ldots - u_k$.

We now introduce the concept of *extrapolated length* $|u \rightarrow v|$ of a gap $u - v$, useful for breaking ties in the gathering process. Let $u - v - v_1 - v_2 - \ldots - v_s$ be the longest possible chain such that for all i, $v_i \neq u$ and v_i does not belong to a multiplicity. Then $|u \rightarrow v| = (|u - v|, |u - v_1|, |u - v_2|, \ldots, |u - v_s|)$. Values of extrapolated gap lengths are compared lexicographically.

A key operation used in the gathering process is known as the *contraction* of a gap. Let $u - v$ be an arbitrary gap belonging to some chain $t - u - v - w$, such that $|u \rightarrow t| > |v \rightarrow w|$. Then the *contraction of* $u - v$ is the operation of moving robot u a single node towards robot v.

Note that if a configuration C' was formed from a configuration C by contraction of some gap $u - v$ (by moving u) in a chain $t - u - v - w$, then it is clear that in C' we have $|t - u| > |v - w|$. The corresponding *de-contraction of* $u - v$ in C' is uniquely defined as the operation of moving robot u a single node away from robot v unless some other symmetry has been determined.

3 Gathering Algorithm

Our algorithm describe the Compute part of the cycle of robots' activities. In order to simplify notation, they are often expressed using configurations (identical for all robots sharing the same snapshot of the system), and not locally-centered views. For example, if we require only robots specifying certain geometrical criteria to move, then each robot will be able to recognize whether to perform the specified action or not.

A gathering algorithm in [16] called RigidGathering provides a solution for all *rigid* configurations, i.e. configurations which are not symmetric and not periodic. It uses a sequential approach: in every configuration, exactly one specific robot is chosen by the algorithm (regardless of the robot running the algorithm), and this robot is then allowed to perform a move.

In our approach, we define an algorithm able to manage all symmetric and gatherable configurations, by allowing at any given time exactly two symmetric robots, u and $\bar{u}$, to make corresponding moves, hence preserving symmetry. Observe that there are two possible move scenarios leading to different types of new configurations:

- *Both robots, u and $\bar{u}$, make their moves simultaneously.* In this case, in the new configuration the axis of symmetry remains unchanged. For the correctness of the algorithm, it is essential to ensure that no new axes of symmetry are formed (since otherwise the new configuration cannot be gathered).
- *One of the robots, say u, performs its move before the other robot $\bar{u}$.* All other robots must be able to recognize that the current configuration is one move

away from a symmetry, and robot $\bar{u}$ is now the only one allowed to perform a move. Observe that it is then irrelevant whether robot $\bar{u}$ performed its Look operation before or after robot u was moved; the outcome of its move is exactly the same.

The algorithm proposed herein detects configurations which have exactly one axis of symmetry of the node-on-axis type (which we call *A-type configurations*) and those which are exactly one step away from such a configuration (*B-type configurations*). For all other gatherable non-symmetrical configurations (*C-type configurations*), a step of the RigidGathering algorithm from [16] is performed. It is assumed that the number of robots is larger than 18 (for an explanation of this value, cf. Remark 1). Thus, for example if the system starts in an A-type configuration, it remains in A-type configurations possibly alternating with B-type configurations. If the system starts without an axis of symmetry, it may either perform a gathering passing through C-type configurations only, or may at some point switch from a C-type configuration to a B-type configuration, and then remain confined to B-type an A-type configurations. In consequence, the eventual convergence of our algorithm to a gathering relies only on the convergence of the RigidGathering algorithm from [16], and on the convergence of the rules we introduce for A-type and B-type configurations.

Our algorithm runs in four main phases; these are informally outlined in the following subsection, and formalized in Subsection 3.2.

3.1 Illustration of Approach

Let us suppose that the system starts in an A-type configuration. (Note that in view of impossibility results from [16] (see Theorem 5), symmetric configurations which are not A-type configurations are never gatherable.) We temporarily also assume here that there are initially no robots on the axis of symmetry.

The four phases of our algorithm can be outlined as follows. In the first phase of the algorithm, we lead the system to an A-type configuration with exactly two (symmetrical) multiplicities. In the second phase, all of the other robots (with the exception of two symmetrically located robots called *guards*) are gathered into the multiplicities. In the third phase, the multiplicities are moved to their final gathering point on the axis of symmetry, away from the guards (remember that there is a node-on-axis symmetry in our case). Finally, in the fourth phase the guards join the single remaining multiplicity in the gathering point.

The current phase of the algorithm can be determined by only looking at the state of the system; this will be discussed in more detail later. The single axis of symmetry is maintained throughout the process. In the first phase, the locations of all minimal gaps are used for this purpose. In the second phase the axis is determined by positions of the multiplicities, while in the third phase by the positions of the guards. Finally, in the fourth phase the gathering point with the only multiplicity is known.

Referring to Figures 1 and 2, we now describe in more details the basic intuitions of our algorithm. In both figures, configurations a describe two possible

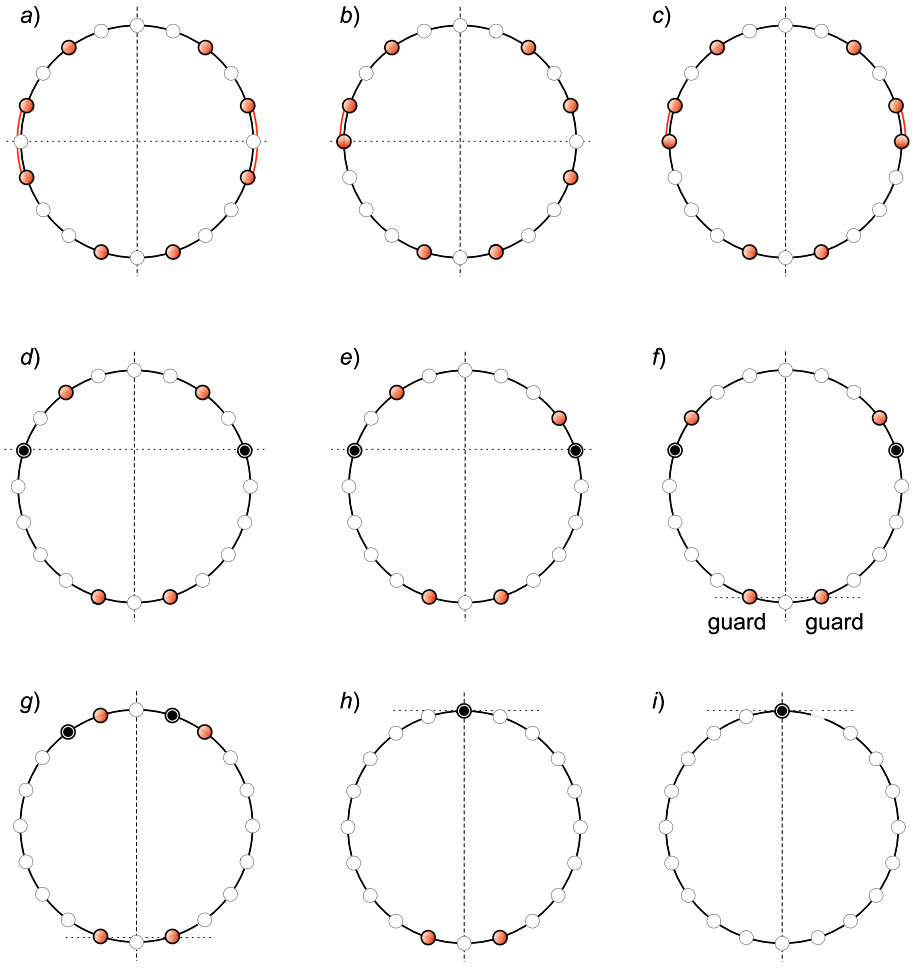

Fig. 1. An example of a scenario for a symmetric configuration. White nodes represent empty nodes, shaded nodes are nodes occupied by a single robot, black nodes are nodes occupied by at least two robots, i.e., multiplicities. The North pole is at the top of the axis of symmetry. The dashed horizontal line can be understood as a helper line for recognizing the axis of symmetry.

initial states of the system (A-type configurations). In the first phase, the objective is to create two symmetric multiplicities such that both arcs of the circle between them contain at least two robots, neither of which is at a distance of one from a multiplicity. The normal move (Fig. 1*a*) consists in the contraction of two symmetrical minimal gaps. The pair of minimal gaps is selected in such a way that the contraction does not create two multiplicities which violate the imposed constraint on robots on the arcs between them; if there exists a minimal gap crossing the axis of symmetry, this is not chosen either. It may happen that no minimal gap appropriate for contraction exists (Fig. 2*a*). In that case,

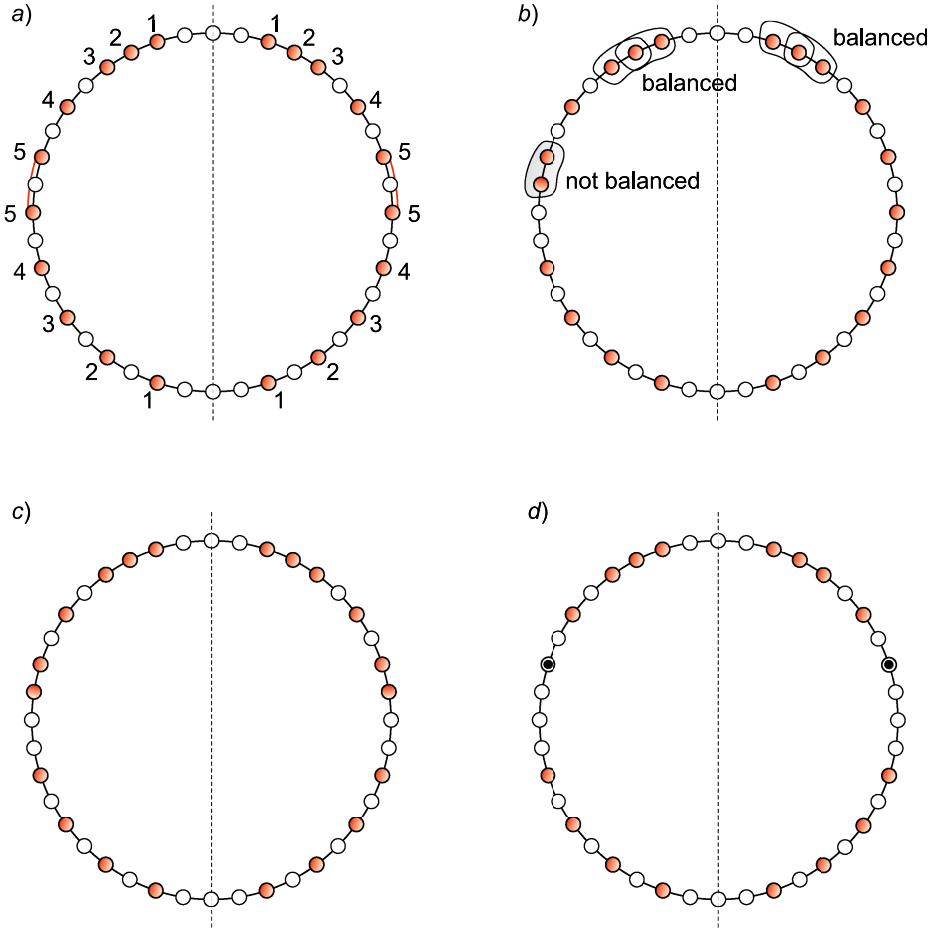

Fig. 2. An example of a scenario for a symmetric configuration (contraction of equatorial gaps) Details of the construction are given in Subsection 3.3

we select for contraction the pair of (not necessarily minimal) gaps which are central in terms of the number of robots separating them from the poles of the axis of symmetry (gaps between robots 5–5 in Fig. 2*a*); if there are two pairs of candidate gaps, a tie-breaking mechanism is applied.

The performed contractions result in a new symmetrical configuration (configurations *c* in both figures), possibly preceded by a state of violated symmetry in a B-type configuration (configurations *b*). The process of selecting the gap for contraction allows the robots to recreate configuration *a* knowing configuration *b* only, and to proceed from there to configuration *c*. Configuration *c* is in fact an A-type configuration just as configuration *a*, and the procedure repeats until the two sought multiplicities are created (configurations *d*). At this point, the first phase of the algorithm is complete. Note that the contraction rules applied in

Fig. 2 require a sufficiently large number of robots (more than 18, see Remark 1) to guarantee correctness.

The next phases of the algorithm are shown in Fig. 1 only. In phase 2 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 has more robots than the southern arc; in 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 (Fig. 1e, f). Note that the definition of the North and South is consistently preserved throughout the process. Phase 2 ends when nearly all the robots have been merged into the multiplicities, and the remaining robots occupy not more than 6 nodes in an arrangement matching a specific pattern (Fig. 1f). Two robots, separated by gaps from the multiplicities, always remain on the southern arc and act as the guards of the axis of symmetry throughout phase 3. The multiplicities are moved step by step towards the North pole; note that not all the robots in a multiplicity have to move simultaneously (Fig. 1g). When the pattern shown in Fig. 1h is achieved, phase 4 starts, and the two remaining robots are moved step by step until they reach the North pole (Fig. 1i), and the algorithm is complete.

3.2 Formalization of Approach

The distinction among the four phases of the proposed algorithm is in fact possible knowing only the current configuration C. To do this, we now introduce some further notation.

A configuration can also be represented in the form of a string of characters as follows: starting from an arbitrary node and moving around the cycle in a chosen direction, for each node we append a character representing whether the node is empty, contains a single robot, or a multiplicity. We say that configuration C matches a *chain pattern* $[P]$, $C \in [P]$, if there exists a string representation of C belonging to the language described by the tokens in $[P]$. For some integer values a and b, token $\sigma_{a:b}$ is understood as between a and b occurrences of single robots (possibly separated by any number of empty nodes) followed by at least one empty node. Token $\mu_{a:b}$ is understood as between a and b occurrences of consecutive non-empty nodes, at least one of which is a multiplicity, followed by at least one empty node. Ranges of the form $a : a$ are simply written as a. For example, in Fig. 1 the pattern $[\mu_{1:2}, \sigma_{1:2}, \mu_2]$ is matched by configurations f and g.

Herein we restrict ourselves to a presentation of the algorithm for the case of an initial configuration with exactly one axis of symmetry, having a node-on-axis type symmetry, without any robots on the axis. The case that allows robots to reside on the axis of symmetry can be addressed by a minor modification of the algorithm as outlined at the end of the section.

The proposed algorithm performs the gathering in four basic phases, as defined in Table 1.

When performing its Compute step, each robot can clearly determine which phase of the algorithm it is currently running (cases not covered in the table

Table 1. Division into phases, assuming no robots on the axis of symmetry in the initial state: $m(C)$ — number of multiplicities in configuration C, $p(C)$ — total number of different nodes occupied by robots in C

Phase	Multiplicities	Occupied nodes	Additional constraints
1	$m(C) < 2$	$p(C) > 6$	none
2	$m(C) = 2$	$p(C) \geq 6$	if $p(C) = 6$, then $C \notin [\mu_{1:2}, \sigma_2, \mu_{1:2}]$
3	$m(C) = 2$	$4 \leq p(C) \leq 6$	if $p(C) = 6$, then $C \in [\mu_{1:2}, \sigma_2, \mu_{1:2}]$
4	$m(C) \geq 1$	$p(C) \leq 3$	none

cannot appear in the initial state and do not occur later due to the construction of the algorithm). The algorithm is defined so as to guarantee that when two robots are allowed to move simultaneously, their views correspond to the same phase of the algorithm. Bearing this in mind, we can now consider the four phases separately in the following subsections.

3.3 Phase 1: Obtaining Two Non-adjacent Multiplicities

The algorithm is defined by the following elements:

- A subroutine defining a move for an A-type configuration which leads to a new A-type configuration, assuming that both the robots which are chosen to move perform their action simultaneously.
- A subroutine for detecting the preceding A-type configuration when the current state of a system is a B-type configuration.

The procedure for A-type configurations is presented as Algorithm 1. A gap $u - v$ is called *equatorial* with respect to a line s if the number of robots on the arc from u to one pole of s and from v to the other pole of s differs by at most 1 (a multiplicity is counted as 2 robots).

For completeness of the procedure, it is necessary to provide some mechanism of choosing one of several possible candidate gaps. Such ties are easily broken, since for a given configuration it is possible to define a partial order on the set of robots in which only symmetrical robots are not comparable [16].

The definition of the procedure always allows a move of exactly two symmetrical robots. We now show that the above set of rules is sufficient to gather an A-type configuration, provided that both symmetrical robots always perform their Look operations as well as Move operations simultaneously (we will call this a *symmetry-preserving scheduler*).

Case of a Symmetry-Preserving Scheduler. Before proceeding with the proofs, we recall the obvious geometrical fact that if for a configuration on the ring it is in some way possible to distinguish (select) exactly two arcs, then the configuration can only have zero, one, or two perpendicular axes of symmetry.

Algorithm 1. Procedure for A-type configurations (Phase 1)

(*i*) Choose a pair of minimal gaps $u - v$ and $\bar{u} - \bar{v}$ such that the following conditions are fulfilled:
- $u - v$ does not intersect the axis of symmetry ($u \neq \bar{v}$),
- the contraction of $u - v$ and $\bar{u} - \bar{v}$ does not create two multiplicities with no other robots in between them,
- the contraction of $u - v$ and $\bar{u} - \bar{v}$ does not create two multiplicities with exactly two robots in between, adjacent to these multiplicities,

then perform the contractions of $u - v$ and $\bar{u} - \bar{v}$.

(*ii*) If no such pair exists, perform the contraction of chosen gaps $u - v$ and $\bar{u} - \bar{v}$ which are equatorial with respect to the axis of symmetry. If there are two pairs of equatorial gaps of different lengths, the shorter pair is always chosen for contraction.

Theorem 1. *Under a symmetry-preserving scheduler, the new configuration after performing rule* (*i*) *is also an A-type configuration.*

Proof. Indeed, consider the contraction of minimal gap $u-v$ in a chain $t-u-v-w$ and its complement $\bar{u}-\bar{v}$ in chain $\bar{t}-\bar{u}-\bar{v}-\bar{w}$. The obtained configuration has exactly two minimal gaps, $u-v$ and $\bar{u}-\bar{v}$, and $|t-u| \neq |v-w|$ by the properties of a contraction. Thus, after the move the axis of symmetry remains unchanged and no new axes are created, since gap $u - v$ must be reflected into $\bar{u} - \bar{v}$. □

For a given configuration C, we will call a gap $u - v$ *balanced* if for the chain $s-t-u-v-w-x$ we have $|t-u| = |v-w|$ or $|u-v| \in \{|s-t|, |t-u|, |v-w|, |w-x|\}$.

Remark 1. If for a given A-type configuration rule (*i*) cannot be applied, we can make the following statements:

- The set of minimal gaps consists of between 1 and 10 gaps formed by at most 12 robots — at most 6 robots surrounding each pole of the axis of symmetry (3 robots on one side and 3 on the other); otherwise, a minimal gap formed by any other robots can always be contracted.
- All the minimal gaps are balanced; this can be shown by a simple enumeration of all possibilities.
- Taking into account the assumption that there are more than 18 robots on the ring, there exist on the cycle exactly two symmetrical maximal arcs containing more than $\frac{18-12}{2} = 3$ (i.e. at least 4) robots each, such that none of these robots are part of some minimal gap.

Taking into account the above remark, we proceed to prove the following.

Theorem 2. *Under a symmetry-preserving scheduler, the new configuration after performing rule* (*ii*) *is also an A-type configuration.*

Proof. We need to consider two cases:

(1) if the contraction of $u - v$ and $\bar{u} - \bar{v}$ creates no new minimal distances, then the axis of symmetry and the set of minimal gaps remain unchanged. There could exist at most one more candidate for an axis of symmetry for the new configuration, perpendicular to the original axis. Since the number of robots on both sides of an axis of symmetry is the same, either the new axis crosses the newly contracted gaps $u - v$ and $\bar{u} - \bar{v}$ or it crosses u and $\bar{u}$. In the first case we have a contradiction since for the chains $t - u - v - w$ we have $|t - u| \neq |v - w|$. In the second case, a contradiction arises as well, since the shortest equatorial gaps have been contracted and hence the shortest gaps cannot be reflected by the new axis into the longest ones.

(2) if the contraction of $u - v$ and $\bar{u} - \bar{v}$ creates two new minimal distances, then these are the only two non-balanced minimal distances on the ring. By a similar argument as before, these two non-balanced minimal distances must be reflected by the axis into each other, so the axis of symmetry is unique. □

Finally, we make a note on the convergence of the performed process.

Theorem 3. *Under a symmetry-preserving scheduler, Phase 1 is completed after a finite number of steps.*

Proof. If rule (i) is performed then the length of the minimal gap decreases in each step. Otherwise, the length of the equatorial gap decreases, while the length of the minimal gap remains unchanged (since all minimal gaps are then concentrated around the poles). The process obviously converges to a minimal gap length of 0, hence we obtain 2 multiplicities and, by Table 1, Phase 1 is complete. □

Extension to the General Scheduler. We now proceed to define the second required subroutine, namely, a procedure to show for a B-type configuration a unique preceding A-type configuration.

Depending on the rule used in the preceding A-type configuration and the outcome of the move, we have the following cases:

B1. The current configuration was obtained by contracting a minimal gap in an A-type configuration using rule (i).
B2: The current configuration was obtained by contracting an equatorial gap in an A-type configuration using rule (ii), but without creating any new minimal gaps in the process.
B3: The current configuration was obtained by contracting an equatorial gap in an A-type configuration using rule (ii), but creating a new minimal gap in the process.

Before proceeding any further, for a configuration we define a *compass axis* as any line s fulfilling the following constraints:

- s is an axis of symmetry of the set of balanced minimal gaps,
- the number of robots on both sides of s is equal,

– all the balanced minimal gaps are contained within a set of 12 robots — 6 robots surrounding each pole of s (3 robots on one side and 3 on the other).

We are now ready to prove the following theorem.

Theorem 4. *The sets of A-, B1-, B2-, and B3-type configurations are all pairwise disjoint.*

Proof. A B1-type configuration has exactly one non-balanced minimal gap. In consequence, such a configuration obviously cannot have an axis of symmetry.

A B2-type configuration has the same set of minimal gaps as the original A-type configuration, hence we can make use of Remark 1 also for this configuration. In consequence, a B2-type configuration has between 1 and 10 minimal gaps, all of which are balanced, and exactly one compass axis identical to the axis of symmetry of the original A-type configuration. Since the compass axis of a configuration is the only possible candidate for its axis of symmetry, and a B2-type configuration is exactly one move apart from an A-type configuration having this axis as an axis of symmetry, a B2-type configuration has no axes of symmetry.

A B3-type configuration has the same set of balanced minimal gaps as the original A-type configuration, and additionally one more non-balanced gap obtained as a result of the contraction (thus between 2 and 11 minimal gaps in total). As in the previous case, this means that a B3-type configuration has exactly one compass axis and no axis of symmetry.

Table 2. Telling apart different types of configurations: $q(C)$ — total number of minimal gaps in C, $q_b(C)$ — total number of balanced minimal gaps in C, $s(C)$ — number of axes of symmetry

Type	Minimal gaps	Balanced minimal gaps	Axes of symmetry
A	irrelevant	irrelevant	$s(C) = 1$
B1	$q(C) = 1$	$q_b(C) = 0$	$s(C) = 0$
B2	$1 \leq q(C) \leq 10$	$q_b(C) = q(C)$	$s(C) = 0$
B3	$2 \leq q(C) \leq 11$	$q_b(C) = q(C) - 1$	$s(C) = 0$

Taking into account the above observations (see Table 2), we obtain that for a given configuration C we can determine if it is an A-type configuration, or a candidate for a B1, B2, or B3-type configuration. In the latter cases, there exists exactly one possibility of recreating the potentially preceding A-type configuration. For a B1-type configuration, it is necessary to de-contract the unique minimal gap. For a B2 or B3-type configuration, the shortest of the gaps equatorial with respect to the compass axis should be de-contracted. If this preceding configuration is indeed an A-type configuration, then the next move is uniquely defined by imitating the stated procedure for A-type configurations. Otherwise, configuration C is some other (type C) configuration which does not require special treatment and can be solved following [16]. □

3.4 Phase 2: Partial Gathering with 2 Multiplicities

The first phase ends when two symmetrical multiplicities are created. Throughout the second phase of the algorithm, the two existing multiplicities M_1 and M_2 make no moves. Multiplicities M_1 and M_2 divide the ring into two parts, which we will call *northern* (around the North pole) and *southern* (around the South pole). Each of these parts initially contains at least two robots not directly adjacent to a multiplicity. Throughout the process North and South are defined in such a way as to fulfill the following conditions:

- the number of nodes in the northern part is odd,
- if both parts have an odd number of nodes, the southern part always contains not less than one robot, and not less robots than the northern part,
- if both parts have an odd number of nodes and contain the same number of robots, consider the chain $r_N - M_1 - r_S$ with robot r_N in the northern part and robot r_S in the southern part; then $|M_1 \rightarrow r_S| > |M_1 \rightarrow r_N|$.

The gathering procedure, presented as Algorithm 2, is defined so as to move all but at most 4 of the single robots into the two existing multiplicities (without creating any new multiplicities).

Algorithm 2. Procedure for Phase 2

(*i*) If the northern part contains at least one robot, move a robot in the northern part, such that there are no robots between itself and one of the multiplicities, towards this multiplicity (in case of choice of robots, select the one with a longer way left to go; if the distance is the same, both robots are allowed to move).

(*ii*) Otherwise, perform an analogous operation in the southern part but for the two symmetric nodes closest to the pole (these nodes will play as guards in the next phase).

It is important to note that the adopted definition of North and South guarantees that the same labeling of the poles is maintained throughout the process.

In accordance with Table 1, the phase ends when all but at most four single robots have been merged with the multiplicities. The last pair of robots in the southern part has not yet made a move and is separated by at least one empty field from a multiplicity; these robots will serve as guards in the last phases of the algorithm.

3.5 Phase 3: Gathering 2 Multiplicities Using Guards

The third phase of the algorithm is performed when $C \in [\mu_{1:2}, \sigma_2, \mu_{1:2}]$. The two robots u and v corresponding to the token σ_2 define a unique axis of symmetry, orthogonal to the gap $u - v$. The remaining robots (and multiplicities) can move towards the North pole of this axis; for a given configuration, only those robots which have the longest way to go are allowed to move. In this way the configuration pattern is maintained throughout the process, until the moving

robots converge on the three nodes near their destination. The configuration pattern then changes to $C \in [\mu_{1:3}, \sigma_2]$, and can be likewise maintained until all robots except for the guards gather at the required pole in a single multiplicity ($C \in [\mu_1, \sigma_2]$).

3.6 Phase 4: Withdrawing Guards to the Gathering Point

In this phase, the unique multiplicity on the North pole determines the gathering point for the remaining guards. The guards can be moved towards the multiplicity following the rule that if the guards are at a different distance from the multiplicity, the guard further away should move (in case of a tie, both guards are allowed to move). The configuration is maintained in the pattern $C \in [\mu_1, \sigma_2]$. Only in at most two final moves we have $C \in [\mu_{1:3}]$ or $C \in [\mu_{1:2}]$ (still with exactly one multiplicity). Eventually, $C \in [\mu_1]$ and the gathering is complete.

3.7 Remarks on the Algorithm

An extended version of the algorithm which is capable of additionally gathering the case of symmetrical configurations with at least one robot on the unique axis of symmetry can be designed analogously in four phases:

- Phase 1 of the algorithm remains unaffected. In the definition of the equatorial gap, the robot in the pole should be ignored.
- Phases 2, 3 and 4 are slightly modified to allow for a single guard robot on the South pole (instead of a pair of guard robots in the southern part).

Note that in the case of robots on the axis of symmetry it may also be possible to design algorithms which break the symmetry by immediately moving the robot located on the axis, as in the case of an odd number of robots described in [16]. In our approach symmetry is never broken until the robots from the poles are moved into a multiplicity (in particular, if there is a single guard robot on the South pole, in Phase 4 it has to be moved to the multiplicity on the North pole).

For configurations with more than 18 robots, our algorithm is complementary to the impossibility result shown in [16].

Theorem 5 ([16]). *Gathering is not feasible for initial configurations which are periodic or have an edge-edge symmetry.*

In this way, we have obtained the sought characterization of initial configurations on the ring.

Theorem 6. *For more than* 18 *anonymous and oblivious robots located on different nodes of a ring, gathering is feasible if and only if the initial configuration is not periodic and does not have an edge-edge symmetry.*

4 Conclusions

We have studied the gathering problem in the discrete model, solving it on a ring for any number of robots larger than 18. The applied technique relies on preserving symmetries (as a matter of fact, our algorithm occasionally creates symmetric configurations from asymmetrical initial configurations).

Theorem 6 implies that, for any number of robots larger than 18, gathering is feasible if and only if, in the initial configuration the robots can elect a node (not necessarily occupied by a robot). Although it is conjectured in [16] that such a claim should also hold in cases with an even number of robots between 4 and 18, this is not always true. For instance, the only possible configuration of 4 robots on a 5-node cycle is not gatherable, although the single empty node can be initially elected as a candidate for the gathering point. Providing an additional characterization for the cases of between 4 and 18 robots is an interesting open problem. Some partial results in this direction have recently been shown in [15].

A natural next step is to consider the gathering problem for other graph classes with high symmetry (such as toruses), and if possible propose an algorithmic approach which solves the problem in the general case. The gathering problem could also be considered for variants of the model, such as robots having limited visibility, although such restrictions often lead to a large number of initial configurations for which gathering is impossible. It is not clear whether allowing robots to have small (constant) memory would help address such problems with achieving a gathering. Finally, it is interesting to ask whether the technique of preserving symmetries proposed herein can also be applied in other contexts.

References

1. Agmon, N., Peleg, D.: Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. SIAM Journal on Computing 36(1), 56–82 (2006)
2. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Kluwer Academic Publishers, Dordrecht (2002)
3. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. IEEE Transactions on Robotics and Automation 15(5), 818–828 (1999)
4. Cieliebak, M.: Gathering Non-oblivious Mobile Robots. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 577–588. Springer, Heidelberg (2004)
5. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Solving the Robots Gathering Problem. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1181–1196. Springer, Heidelberg (2003)
6. Cohen, R., Peleg, D.: Robot Convergence via Center-of-Gravity Algorithms. In: Kralovic, R., Sýkora, O. (eds.) SIROCCO 2004. LNCS, vol. 3104, pp. 79–88. Springer, Heidelberg (2004)
7. Cohen, R., Peleg, D.: Convergence of Autonomous Mobile Robots with Inaccurate Sensors and Movements. SIAM Journal on Computing 38(1), 276–302 (2008)
8. De Marco, G., Gargano, L., Kranakis, E., Krizanc, D., Pelc, A., Vaccaro, U.: Asynchronous deterministic rendezvous in graphs. Theoretical Computer Science 355, 315–326 (2006)

9. Dessmark, A., Fraigniaud, P., Kowalski, D., Pelc, A.: Deterministic rendezvous in graphs. Algorithmica 46, 69–96 (2006)
10. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Computing without communicating: ring exploration by asynchronous oblivious robots. In: Tovar, E., Tsigas, P., Fouchal, H. (eds.) OPODIS 2007. LNCS, vol. 4878, pp. 105–118. Springer, Heidelberg (2007)
11. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Remembering without Memory: Tree Exploration by Asynchronous Oblivious Robots. In: Shvartsman, A.A., Felber, P. (eds.) SIROCCO 2008. LNCS, vol. 5058, pp. 33–47. Springer, Heidelberg (2008)
12. Flocchini, P., Kranakis, E., Krizanc, D., Santoro, N., Sawchuk, C.: Multiple Mobile Agent Rendezvous in a Ring. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 599–608. Springer, Heidelberg (2004)
13. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of Asynchronous Robots with Limited Visibility. Theoretical Computer Science 337(1-3), 147–168 (2005)
14. Gąsieniec, L., Kranakis, E., Krizanc, D., Zhang, X.: Optimal Memory Rendezvous of Anonymous Mobile Agents in a Unidirectional Ring. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 282–292. Springer, Heidelberg (2006)
15. Haba, K., Izumi, T., Katayama, Y., Inuzuka, N., Wada, K.: On the Gathering Problem in a Ring for 2n Autonomous Mobile Robots, Technical Report COMP2008-30, IEICE, Japan (2008)
16. Klasing, R., Markou, E., Pelc, A.: Gathering asynchronous oblivious mobile robots in a ring. Theoretical Computer Science 390(1), 27–39 (2008)
17. Kowalski, D., Pelc, A.: Polynomial deterministic rendezvous in arbitrary graphs. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 644–656. Springer, Heidelberg (2004)
18. Lynch, N.: Distributed Algorithms. Morgan Kaufmann, San Francisco (1996)
19. Prencipe, G.: CORDA: Distributed Coordination of a Set of Autonomous Mobile Robots. In: Proceedings of the European Research Seminar on Advances in Distributed Systems (ERSADS), pp. 185–190 (2001)
20. Prencipe, G.: On the Feasibility of Gathering by Autonomous Mobile Robots. In: Pelc, A., Raynal, M. (eds.) SIROCCO 2005. LNCS, vol. 3499, pp. 246–261. Springer, Heidelberg (2005)
21. Suzuki, I., Yamashita, M.: Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. SIAM Journal on Computing 28(4), 1347–1363 (1999)