Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Gathering in dynamic rings [☆]

Giuseppe Antonio Di Luna [a,*], Paola Flocchini [b], Linda Pagli [c], Giuseppe Prencipe [c], Nicola Santoro [d], Giovanni Viglietta [b]

[a] LiF, Aix-Marseille University, Marseille, France
[b] SITE, University of Ottawa, Ottawa, Canada
[c] Dipartimento di Informatica, University of Pisa, Italy
[d] Carleton University, Ottawa, Canada

A R T I C L E I N F O

Article history:
Received 29 October 2017
Received in revised form 8 April 2018
Accepted 15 October 2018
Available online xxxx

Keywords:
Dynamic networks
Mobile agents
Gathering

A B S T R A C T

The *gathering* (or *multi-agent rendezvous*) problem requires a set of mobile agents, arbitrarily positioned at different nodes of a network to group within finite time at the same location, not fixed in advanced.
The extensive existing literature on this problem shares the same fundamental assumption: the topological structure does not change during the rendezvous or the gathering; this is true also for those investigations that consider faulty nodes. In other words, they only consider *static graphs*.
In this paper we start the investigation of gathering in *dynamic graphs*, that is networks where the topology changes continuously and at unpredictable locations.
We study the feasibility of gathering mobile agents, identical and without explicit communication capabilities, in a *dynamic ring* of anonymous nodes; the class of dynamics we consider is the classic *1-interval-connectivity*; i.e., dynamic graphs that are however connected at any point in time. We focus on the impact that factors such as *chirality* (i.e., a common sense of orientation) and *cross detection* (i.e., the ability to detect, when traversing an edge, whether some agent is traversing it in the other direction), have on the solvability of the problem; and we establish several results.
We provide a complete characterization of the classes of initial configurations from which the gathering problem is solvable in presence and in absence of cross detection and of chirality. The feasibility results of the characterization are all constructive: we provide distributed algorithms that allow the agents to gather within low polynomial time. In particular, the algorithms for gathering with cross detection are time optimal.
We also show that cross detection is a powerful computational element. Indeed, we prove that, without chirality, knowledge of the ring size n is strictly more powerful than knowledge of the number k of agents; on the other hand, with chirality, knowledge of n can be substituted by knowledge of k, yielding the same classes of feasible initial configurations.
From our investigation it follows that, for the gathering problem, the computational obstacles created by the dynamic nature of the ring can be overcome by the presence of chirality or of cross-detection.

[☆] A preliminary version of the results contained in this paper appeared in [21].
[*] Corresponding author.
E-mail addresses: di-luna.g@univ-amu.fr (G.A. Di Luna), paola.flocchini@uottawa.ca (P. Flocchini), linda.pagli@unipi.it (L. Pagli), giuseppe.prencipe@unipi.it (G. Prencipe), santoro@scs.carleton.ca (N. Santoro), gvigliet@uottawa.ca (G. Viglietta).

https://doi.org/10.1016/j.tcs.2018.10.018
0304-3975/© 2018 Elsevier B.V. All rights reserved.

1. Introduction

1.1. Background and problem

The *gathering* problem requires a set of k mobile computational entities, dispersed at different locations in the spacial universe they inhabit, to group within finite time at the same location, not fixed in advanced. This problem models many situations that arise in the real world, e.g., searching for or regrouping animals, people, equipment, and vehicles.

This problem, known also as *multi-agent rendezvous*, has been intensively and extensively studied in a variety of fields, including operations research (e.g., [1]) and control (e.g., [36]), the original focus being on the *rendezvous* problem, i.e. the special case $k = 2$.

In distributed computing, this problem has been extensively studied both in continuous and in discrete domains. In the *continuous* case, both the gathering and the rendezvous problems have been investigated in the context of swarms of autonomous mobile *robots* operating in one- and two-dimensional spaces, requiring them to meet at (or converge to) the same point (e.g., see [10,11,17,25,26,37]).

In the *discrete* case, the mobile entities, usually called *agents*, are dispersed in a network modeled as a graph and are required to gather at the same node (or at the two sides of the same edge) and terminate (e.g., see [2,16,22,24,30–33,40,41]). The main obstacle for solving the problem is *symmetry*, which can occur at several levels (topological structure, nodes, agents, communication), each playing a key role in the difficulty of the problem and of its resolution. For example, when the network nodes are uniquely numbered, solving the gathering problem is trivial. On the other hand, when the network nodes are anonymous, the network topology is highly symmetric, the mobile agents are identical, and there is no means of communication, the problem is clearly impossible to solve by deterministic means. The quest has been for minimal empowering assumptions which would make the problems deterministically solvable.

A very common assumption is for the agents to have distinct *identities* (e.g., see [13,16,41]). This enables different agents to execute different deterministic algorithms; under such an assumption, the problem becomes solvable, and the focus is on the complexity of the solution.

An alternative type of assumption consists in empowering the agents with some minimal form of *explicit communication*. In one approach, this is achieved by having a whiteboard at each node giving the agents the ability to leave notes in each node they travel (e.g., [2,9,22]); in this case, some form of gathering can occur even in presence of some faults [9,22]. A less explicit and more primitive form of communication is by endowing each agent with a constant number of movable tokens, i.e. pebbles that can be placed on nodes, picked up, and carried while moving (e.g., [12]).

A less demanding assumption is that of having the homebases (i.e., the nodes where the agents are initially located) identifiable by a mark, identical for all homebases, and visible to any agent passing by it. This setting is clearly much less demanding that agents having identities or explicit communication; originally suggested in [3], it has been used and studied e.g., in [24,33,39].

Summarizing, the existing literature on the gathering and rendezvous problems is extensive and the variety of assumptions and results is abundant (for surveys see [32,38]). However, regardless of their differences, all these investigations share the same fundamental assumption that the topological structure does not change during the rendezvous or the gathering; this is true also for those investigations that consider faulty nodes (e.g., see [5,9,22]). In other words, they only consider *static graphs*.

Recently, within distributed computing, researchers started to investigate *dynamic graphs*, that is graphs where the topological changes are not localized and sporadic; on the contrary, the topology changes continuously and at unpredictable locations, and these changes are not anomalies (e.g., faults) but rather integral part of the nature of the system [8].

The study of distributed computations in highly dynamic graphs has concentrated on problems of information diffusion, reachability, agreement, and exploration (e.g., [4,6,7,18,19,27–29,34,35]).

In this paper we start the investigation of gathering in dynamic graphs by studying the feasibility of this problem in *dynamic rings*. Note that rendezvous and gathering in a ring, the prototypical symmetric graph, have been intensively studied in the static case (e.g., see the monograph on the subject [32]). The presence, in the static case, of a mobile faulty agent that can block other agents, considered in [14,15], could be seen as inducing a particular form of dynamics. Other than that, nothing is known on gathering in dynamic rings.

1.2. Main contributions

In this paper, we study gathering of k agents, identical and without communication capabilities, in a dynamic ring of n anonymous nodes with identically marked homebases. The class of dynamics we consider is the classic *1-interval-connectivity* (e.g., [20,27,34,35]); that is, the system is fully synchronous and under a (possibly unfair) adversarial schedule that, at each time unit, chooses which edge (if any) will be missing. Notice that this setting is not reducible to the one considered in [14,15].

In this setting, we investigate under what conditions the gathering problem is solvable. In particular, we focus on the impact that factors such as *chirality* (i.e., common sense of orientation) and *cross detection* (i.e., the ability to detect, when traversing an edge, whether some agent is traversing it in the other direction), have on the solvability of the problem. Since,

Table 1
Each entry shows the set of feasible configurations and the time complexity of the gathering algorithm; $\mathcal{C}$, $\mathcal{P}$ and $\mathcal{E}$ denote the set of all possible configurations, periodic configurations, and configurations with an unique symmetry axis passing through edges of the ring, respectively.

	no chirality	chirality
cross detection	feasible: $\mathcal{C} \setminus \mathcal{P}$ time: $\mathcal{O}(n)$ knowledge: n (Sec. 4.1)	feasible: $\mathcal{C} \setminus \mathcal{P}$ time: $\mathcal{O}(n)$ knowledge: n or k (Sec. 4.3)
no cross detection	feasible: $\mathcal{C} \setminus (\mathcal{P} \cup \mathcal{E})$ time: $\mathcal{O}(n^2)$ knowledge: n (Sec. 5.2)	feasible: $\mathcal{C} \setminus \mathcal{P}$ time: $\mathcal{O}(n \log n)$ knowledge: n or k (Sec. 5.1)

as we prove, gathering at a single node cannot be guaranteed in a dynamic ring, we allow gathering to occur either at the same node, or at the two end nodes of the same link.

A main result of our investigation is the complete characterization of the classes $\mathcal{F}(X, Y)$ of initial configurations from which the gathering problem is solvable with respect to chirality ($X \in \{\texttt{chirality}, \neg\texttt{chirality}\}$) and cross detection ($Y \in \{\texttt{detection}, \neg\texttt{detection}\}$).

In obtaining this characterization, we establish several interesting results. For example, we show that, without chirality, cross detection is a powerful computational element; in fact, we prove (Theorems 1 and 5):

$$\mathcal{F}(\neg\texttt{chirality}, \neg\texttt{detection}) \subsetneq \mathcal{F}(\neg\texttt{chirality}, \texttt{detection})$$

Furthermore, in such systems knowledge of the ring size n cannot be substituted by knowledge of the number of agents k (at least one of n and k must be known for gathering to be possible); in fact, we prove that, with cross detection but without chirality, knowledge of n is strictly more powerful than knowledge of k.

On the other hand, we show that, with chirality, knowledge of n can be substituted by knowledge of k, yielding the same classes of feasible initial configurations. Furthermore, with chirality, cross detection is no longer a computational separator; in fact (Theorems 3 and 4)

$$\mathcal{F}(\texttt{chirality}, \neg\texttt{detection}) = \mathcal{F}(\texttt{chirality}, \texttt{detection})$$

We also observe that

$$\mathcal{F}_{static} = \mathcal{F}(\texttt{chirality}, *) = \mathcal{F}(\neg\texttt{chirality}, \texttt{detection})$$

where $\mathcal{F}_{static}$ denotes the set of initial configurations from which gathering is possible in the static case. In other words: *with chirality or with cross detection, it is possible to overcome the computational obstacles created by the highly dynamic nature of the system.*

All the feasibility results of this characterization are constructive: for each situation, we provide a distributed algorithm that allows the agents to gather within low polynomial time. In particular, the algorithms for gathering with cross detection, terminating in $O(n)$ time, are time *optimal*. Moreover, our algorithms are *effective*; that is, starting from any arbitrary configuration C in a ring with conditions X and Y, within finite time the agents determine whether or not $C \in \mathcal{F}(X, Y)$ is feasible, and gather if it is. See Table 1 for a summary of some of the results and the sections where they are established. A preliminary version of the results contained in this paper appeared in [21].

2. Model and basic limitations

2.1. Model and terminology

Let $\mathcal{R} = (v_0, \dots v_{n-1})$ be a synchronous dynamic ring where, at any time step $t \in N$, one of its edges might not be present; the choice of which edge is missing (if any) is controlled by an adversarial scheduler, not restricted by fairness assumptions. Such a dynamic network is known in the literature as a *1-interval connected* ring.

Each node v_i is connected to its two neighbors v_{i-1} and v_{i+1} via distinctly labeled ports q_{i-} and q_{i+}, respectively (all operations on the indices are modulo n); the labeling of the ports is arbitrary and thus might not provide a globally consistent orientation. Each port of v_i has an *incoming* buffer and an *outgoing* buffer. Finally, the nodes are *anonymous* (i.e., have no distinguished identifiers).

Agents Operating in $\mathcal{R}$ is a set $\mathcal{A} = \{a_0, \dots, a_{k-1}\}$ of computational entities, called agents, each provided with memory and computational capabilities. The agents are *anonymous* (i.e., without distinguishing identifiers) and all execute the same protocol.

When in a node v, an agent can be *at* v or in one of the port buffers. Any number of agents can be in a node at the same time; an agent can determine how many other agents are in its location and where (in incoming buffer, in outgoing buffer, at the node). Initially the agents are located at distinct locations, called homebases; nodes that are homebases are specially marked so that each agent can determine whether or not the current node is a homebase. Note that, as discussed later, this assumption is necessary in our setting.

Each agent a_j has a consistent private orientation λ_j of the ring which designates each port either *left* or *right*, with $\lambda_j(q_{i-}) = \lambda_j(q_{k-})$, for all $0 \le i, k < n$. The orientation of the agents might not be the same. If all agents agree on the orientation, we say that there is *chirality*.

The agents are *silent*: they not have any explicit communication mechanism.

The agents are *mobile*, that is they can move from node to neighboring node. More than one agent may move on the same edge in the same direction in the same round.

We say that the system has *cross detection* if whenever two or more agents move in opposite directions on the same edge in the same round, the involved agents detect this event; however they do not necessarily know the number of the involved agents in either direction.

Synchrony and behavior The system operates in synchronous time steps, called *rounds*.

In each round, every agent is in one of a finite set of system states $\mathcal{S}$, which includes two special states: the initial state Init and the terminal state Term.

At the beginning of a round r, an agent a at a node v determines the number and the positions of all the agents in v; notice that this information, called *snapshot*, is the same for all the agents in v at the beginning of round r.

It then executes its protocol (the same for all agents) using as input this snapshot, its state, and the content of its local memory; notice that, if cross detection is available, the information on whether or not the agent crossed some agents in the previous round is part of its local memory.

The output of the execution of the protocol is the decision on whether or not to move and, if so, in which direction ($direction \in \{left, right, nil\}$).

If $direction = nil$, the agent places itself at v (if currently on a port). If $direction \neq nil$, the agent moves in the outgoing buffer of the corresponding port (if not already there); if the link is present, it arrives in the incoming buffer of the corresponding port of the destination node at the beginning of round $r + 1$; otherwise the agent stays in the outgoing buffer until round $r + 1$. As a consequence, an agent can be in an outgoing buffer at the beginning of round $r + 1$ if and only if the corresponding link was not present at round r. In the following, if an agent is in an outgoing buffer whose corresponding edge is missing, we will say that the agent is *blocked*.

We will say that a set of agents forms a *group* if they are in the same node, in the same state, and all of them have the same direction of movement. In our algorithm we will ensure that, once a group is formed, it will never split; thus, a group can only increase in size.

Gathering problem Let $(\mathcal{R}, \mathcal{A})$ denote a system defined as above. In $(\mathcal{R}, \mathcal{A})$, gathering is achieved in round r if all agents in $\mathcal{A}$ are on the same node or on two neighboring nodes in round r; in the first case, gathering is said to be *strict*.

An algorithm *solves* GATHERING if, starting from any configuration from which gathering is possible, within finite time all agents are in the terminal state, are gathered, and are aware that gathering has been achieved.

A solution algorithm is *effective* if starting from any configuration from which gathering is *not* possible, within finite time all agents detect such impossibility.

2.2. Configurations and elections

The locations of the k home bases in the ring is called a *configuration*. Let $\mathcal{C}$ be the set of all possible configurations with k agents.

Given a configuration $C \in \mathcal{C}$, let $h_0, \ldots, h_{k-1}$ denote the nodes corresponding to the marked homebases (in clockwise order), and let d_i $(0 \le i \le k-1)$ denote the distance (i.e., number of edges) between h_i and h_{i+1} (in clockwise order), where all operations on the indices are modulo k. Let $\delta^{+j} = \langle d_j, d_{j+1}, \ldots, d_{j+k-1}\rangle$ and $\delta^{-j} = \langle d_{j-1}, \ldots, d_{j-(k-1)}\rangle$ denote the clockwise and the counter-clockwise sequence, respectively, of *inter-distances* with respect to h_j; the unordered pair $(\delta^{+j}, \delta^{-j})$ describes the configuration from the point of view of node h_j.

A configuration is *periodic* with period p (with $p|k$) if $\delta^{+i} = \delta^{+(i+p)}$ for all $i = 0, \ldots k-1$ (and, thus, also $\delta^{-i} = \delta^{-(i+p)}$); it is *aperiodic* otherwise. Let $\mathcal{P}$ denote the set of periodic configurations.

Let $\Delta^+ = \{\delta^{+j} : 0 \le j < k-1\}$ and $\Delta^- = \{\delta^{-j} : 0 \le j < k-1\}$. We will denote by δ_{min} the minimum sequence in $\Delta^+ \cup \Delta^-$ according to the ascending lexicographical order.

Among the aperiodic configurations, particular ones are those where $\delta_{min} = \delta^{+i} = \delta^{-j}$ with $i \neq j$; such aperiodic configurations are called *double palindrome* because the two sequences between the corresponding home bases h_i and h_j are both palindrome. A double palindrome configuration has thus a unique axis of symmetry, equidistant from h_i and h_j. If such an axis passes through two edges (i.e., the distances between h_i and h_j are both odd), we say that the configuration is *edge–edge*, and we denote by $\mathcal{E}$ the set of edge–edge configurations. Note that is an aperiodic configuration is not a non double-palindrome, then it is *asymmetric*.

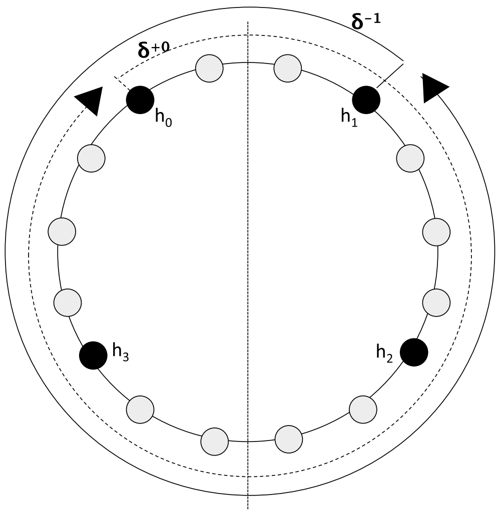


Fig. 1. Double palindrome configuration, the home bases h_0, h_1, h_2, h_3 are the bold nodes. The distances between them are $d_0 = 3$, $d_1 = 4$, $d_2 = 5$, $d_3 = 4$. The symmetry axis is the dashed line.

Example 1. Let $k = 4$ and h_0, h_1, h_2, h_3 be the four home bases with $d_0 = 3$, $d_1 = 4$, $d_2 = 5$, $d_3 = 4$. In this case we have $\delta_{min} = \delta^{+0} = \delta^{-1} = \langle 3, 4, 5, 4\rangle$ and the unique axis of symmetry passes through two edges (one half-way between h_0 and h_1, the other half-way between h_2 and h_3). In this example, the two palindrome sequences are $< 3 >$ and $< 4, 5, 4 >$ (see Fig. 1).

For static rings, a characterization of configurations where a node (or an edge) can be elected can be easily derived from well known results for rings where Ids drawn from a multiset are associated to the nodes [23]; in the context of [23] "configurations" were *strings of Ids*, while in our context, where nodes are anonymous but home bases are marked, configurations are *strings of distances*.

Property 1. *[23] In a static ring without chirality, a leader node can be elected from configuration C if and only if $C \in \mathcal{C} \setminus (\mathcal{P} \cup \mathcal{E})$; a leader edge can be elected if and only if $C \in \mathcal{C} \setminus \mathcal{P}$. With chirality, a leader node can be elected if and only if $C \in \mathcal{C} \setminus \mathcal{P}$.*

The *canonical* way to elect a leader node (or a leader edge when electing a node is impossible) from configuration $C \in \mathcal{C}$ can be informally described as follows: If C is asymmetric, the leader is the unique homebase that starts the lexicographically smallest inter-distance sequence. If C is double palindrome, let h and h' be the two homebases that start (in opposite direction) the two identical lexicographically smallest sequences: if $C \in \mathcal{E}$ the leader edge is the edge in the middle of the shortest portion of the ring delimited by h and h' (note that both portions have odd distance and there is a central edge); otherwise ($C \notin \mathcal{E}$) at least one of the two portions of the ring between h and h' has even distance and a central node is identified as the leader. In Example 1, the leader edge would be the closest central edge between home bases h_0 and h_1.

2.3. Basic limitations and properties

First observe that, in our setting, it is necessary for the homebases to be distinguishable from the other nodes.

Property 2. *If the homebases are not distinguishable from the other nodes, then* Gathering *is unsolvable in $(\mathcal{R}, \mathcal{A})$; this holds regardless of chirality, cross detection, and knowledge of k and n.*

Proof. Let the homebases be not distinguishable from the other nodes in $(\mathcal{R}, \mathcal{A})$. Consider an execution in which all the entities have the same chirality and no link ever disappears. Because of anonymity of the nodes and of the agents, and since homebases are not marked as such, in each round all agents will perform exactly the same action (i.e., stay still or move in the same direction). Thus the distance between neighboring agents will never change, and hence gathering will never take place if $k > 2$. For $k = 2$, by choosing the initial distance between the two agents to be greater than one, the same argument leads to the same result. □

Thus, in the following, we assume that the homebases are identical but distinguishable from the other nodes.

A very basic limitation that holds even if the ring is static is the following.

Property 3. *[2] In $(\mathcal{R}, \mathcal{A})$, if neither n nor k are known, then* Gathering *is unsolvable; this holds regardless of chirality and cross detection.*

Hence, at least one of n or k must be known.

An important limitation follows from the dynamic nature of the system:

Property 4. *In $(\mathcal{R}, \mathcal{A})$, strict* Gathering *is unsolvable; this holds regardless of chirality, cross detection, and knowledge of k and n.*

Proof. Consider the following strategy of the adversarial scheduler. It selects two arbitrary agents, a and b; at each round, the adversary will not remove any edge, unless the two agents would meet in the next step. More precisely, if the two agents would meet by both independently moving on different edges e' and e'' leading to the same vertex, then the adversary removes one of the two edges; if instead one agent is not moving from its current node v, while the other is moving on an edge e incident to v, the adversary removes edge e. This strategy ensures that a and b will never be at the same node at the same time. □

Hence, in the following we will not require gathering to be strict.

An obvious but important limitation, inherent to the nature of the problem, holds even in static situations:

Property 5. Gathering *is unsolvable if the initial configuration $C \in \mathcal{P}$; this holds regardless of chirality, cross detection, and knowledge of k and n.*

Proof. It is sufficient to consider an execution in which all the entities have the same chirality and no link ever disappears. Depending on their initial positions, the agents can be partitioned into k/p congruent classes, each composed of p agents, where p is the period of the initial configuration. In each round, all agents of the same class will perform exactly the same action (i.e., stay still or move in the same direction) based on the same observation. Thus the distance between two consecutive agents of the same class will never change; hence gathering will never take place. □

Hence, in the following we will focus on initial configurations not in $\mathcal{P}$.

3. General solution structure

We present several solution algorithms, depending on whether or not there is chirality and/or cross detection. All solution algorithms have the same general structure and use the same building block and variables.

General structure All the algorithms are composed of two phases.

The goal of Phase 1 is for the agents to explore the ring. In doing so, they may happen to solve Gathering as well. If they complete Phase 1 without gathering, the agents are able to elect a node or an edge (depending on the specific situation) and the algorithm proceeds to Phase 2.

In Phase 2 the agents try to gather around the elected node (or edge); however, gathering on that node (or edge) might not be possible due to the fact that all edges might not be present at all times.

Different strategies are devised, depending on the setting (availability or lack of cross detection and presence or not of chirality) to guarantee that in finite time the problem is solved in spite of the choice of schedule of missing links decided by the adversary.

In the following, we will describe the two phases for each setting; intuitively, cross detection is useful to simplify termination in Phase 2, chirality helps in breaking symmetries.

Exploration building block At each round, an agent evaluates a set of predicates: depending on the result of this evaluation, it chooses a direction of movement and possibly a new state. In its most general form, the evaluation of the predicates occurs through the building block procedure Explore ($dir \mid p_1 : s_1;\ p_2 : s_2;\ \ldots;\ p_h : s_h$), where *dir* is either *left* or *right*, p_i is a predicate, and s_i is a state. In Procedure Explore, the agent evaluates the predicates $p_1, \ldots, p_h$ in order; as soon as a predicate is satisfied, say p_i, the procedure exits and the agent does a transition to the specified state, say s_i. If no predicate is satisfied, the agent tries to move in the specified direction *dir* and the procedure is executed again in the next round. In particular, the following predicates are used:

- *meeting*: satisfied when the agent (either in a port or at a node) in the current round detects an increase in the numbers of agents it sees. In other words, the number of agents seen in the current round is greater than the number of agents seen in the last round.

- *meetingSameDir*: satisfied when the agent detects, in the current round, new agents moving in the same direction as its own (i.e., new agents in an incoming or outgoing buffer corresponding to its current direction). In more detail: the agent keeps a variable g that indicates the number of agents in its own group; initially, $g = 1$. The agent increments g at round r in two situations: (1) the agent is waiting in an outgoing buffer and it sees x agents in the incoming buffer (of the same node) that are moving in its own direction; (2) the agent is waiting in an incoming buffer and it sees x agents in the outgoing buffer (of the same node) that are moving in its own direction. In both cases it sets $g = g + x$. Predicate *meetingSameDir* is satisfied in a round if and only if g increases in that round. Finally, notice that, by design, the number of agents inside a group never decreases.
- *meetingOppositeDir*: satisfied when the agent detects, in the current round, agents moving in its opposite direction. This occurs when seeing agents in an incoming or outgoing buffer corresponding to the direction opposite to the one of the agents.
- *crossed*: satisfied when the agent, while traversing a link, detects in the current round other agent(s) moving on the same link in the opposite direction.
- *seeElected*: satisfied when the agent, having elected a node (resp., an edge), has reached the elected node (resp., an endpoint of the elected edge).

Furthermore, the agents keeps six variables during the execution of the algorithm. Two of them are never reset during the execution; namely:

- $Ttime$: the total number of rounds (initially set to 0) since the beginning of the execution of the algorithm.
- $TotalAgents$: the number of total agents (initially set to 1). This variable will be set only after the agent completes a whole loop of the ring, and will be equal to k.

The other four variables are periodically reset; specifically:

- r_{ms}: the last round when the agent meets at a node someone that is moving in the same direction (i.e., the last round at which *meetingSameDir* was satisfied); this value, initially set to 0, is updated each time a new agent is met, and it is reset at each change of state or direction of movement.
- $Btime$: the number of rounds the executing agent has been blocked trying to traverse a missing edge since r_{ms}. This variable is reset to 0 each time the agent either traverses an edge or changes direction to traverse a new edge.
- $Etime$, $Esteps$: the total number of rounds and edge traversals, respectively. These values are reset at each new call of procedure Explore or when r_{ms} is changed.
- $Agents$: the number of agents at the node, including agents inside ports, of the executing agent. This value is set at each round.

4. Gathering with cross detection

In this section, we study gathering in dynamic rings when there is cross detection; that is, an agent crossing a link can detect whether other agents are crossing it in the opposite direction. Recall that, by Property 3, at least one of n and k must be known.

We first examine the problem without chirality and show that, with knowledge of n, it is solvable in all configurations that are feasible in the static case; furthermore, this is done in optimal time $\Theta(n)$. On the other hand, with knowledge of k alone, the problem is unsolvable.

We then examine the problem with chirality, and show that in this case the problem is solvable in all configurations that are feasible in the static case even with knowledge of k alone; furthermore, this is done in optimal time $\Theta(n)$.

4.1. With cross detection: without chirality

In this section, we present and analyze the algorithm, Gather(Cross, $\not{C}$hir), that solves Gathering in rings of known size with cross selection but without chirality.

The two phases of the algorithm are described and analyzed below.

4.1.1. Algorithm Gather(Cross, $\not{C}$hir): Phase 1

The overall idea of this phase, shown in Fig. 2, is to let the agents move long enough along the ring to guarantee that, if they do not gather, they all manage to fully traverse the ring in spite of the link removals. In Phase 1 there are two important checkpoints, at rounds $6n$ and $12n$.

Let A be the set of all agents that at round $6n$ are moving in the same direction. We will show that these agents are able to check if a specific predicate (formally introduced later) is satisfied. If the predicate is satisfied, all agents in A know that they belong to the same group. Otherwise (the predicate is not satisfied), the agents in A know that they all have explored the ring and thus they can all compute the initial placement of the homebases. However, they do not know yet what happened to agents that are not in A.

```
States: {Init, SwitchDir, KeepDir, Term}.
In state Init:
    Explore ($left$ | $Ttime = 6n \wedge \neg Pred$: SwitchDir; $Ttime = 6n \wedge Pred$: KeepDir)
In state SwitchDir:
    Explore($right$ | $Ttime = 12n \wedge (r_{ms} < 9n \wedge Esteps < n-1) \wedge (Agents = TotalAgents \wedge \neg meetingOppositeDir)$: Term; $Ttime = 12n$: Phase 2)
In state KeepDir:
    Explore ($left$ | $crossed \vee meetingOppositeDir$: Term; $Ttime = 12n \wedge r_{ms} < 9n \wedge Esteps < n-1$: Term; $Ttime = 12n$: Phase 2)

$Pred \equiv [r_{ms} < 3n \wedge Esteps < n-1]$

Algorithm Gather(Cross, $\not{C}$hir), Phase 1.
```

Fig. 2. Phase 1 of Algorithm Gather(Cross, $\not{C}$hir).

Between rounds $6n$ and $12n$, all agents try to do a whole loop around the ring; we will prove that, either all agents have explored the ring (and they are all aware of that), or they all have gathered and detected the completion of gathering. If they have not gathered by round 12, then they start Phase 2.

Detailed algorithm More precisely, for the first $6n$ rounds each agent attempts to move to the left (according to its orientation). At round $6n$, the agent checks if it met someone new going in its same direction more than $3n$ rounds ago and, since then, it has traversed less than $n-1$ links (predicate $Pred \equiv (r_{ms} < 3n \wedge Esteps < n-1)$). If $Pred$ holds, then (as we show) all agents moving in the same direction of this agent are now together, and $Pred$ holds for all of them; they all enter state KeepDir and continue in the same direction for an additional $6n$ rounds. If instead $Pred$ is false, then (as we show) all agents moving in the same direction of this agent have explored the entire ring, thus know the total number k of agents (local variable $TotalAgents$) and the configuration of homebases, and $Pred$ is false for all of them; all of them switch direction, enter state SwitchDir, and attempt to move for the next $6n$ rounds. During this time, if an agent in state KeepDir crosses or meets an agent moving in the opposite direction, it terminates.

The second checkpoint occurs at round $12n$. At that time, an agent in state SwitchDir terminates if all the following conditions hold: (i) it crossed less than $n-1$ links, (ii) it met someone less than $9n$ rounds ago, (iii) it never met anybody in opposite direction, and (iv) there are k agents on the current node; otherwise, it starts Phase 2.

At round $12n$, an agent in state KeepDir terminates if it crossed less than n links and met someone less than $9n$ rounds ago; otherwise, it starts Phase 2.

As we will show, if an agent terminates in Phase 1, then every agent terminates and gathering is achieved. On the other hand, if no agent terminates in Phase 1, all of them have done a complete tour of the ring and start Phase 2.

We now formally prove the properties of Phase 1.

Lemma 1. *Let A be a set of agents moving in the same direction during an interval of time I lasting at least $3n-1$ rounds. If an agent $a^* \in A$ moves less than $n-1$ steps in I, then there exists a round $r \in I$ where all agents in A are in the same group.*

Proof. If there is a round $r' \in I$ when a^* is blocked, then every $a \in A$ that at round r' is not at the same node of a^* does move, due to the 1-interval connectivity of the ring. Since a^* moves less than $n-1$ steps in an interval lasting at least $3n-1$ rounds, then the number of rounds in which a^* is blocked is at least $2n+1$. Thus, all agents in A that are not already in the same node as a^* have moved towards a^* of at least $2n+1$ steps. On the other hand, every time a^* moves, the other agents might be blocked; however, by hypothesis, this has happened less than n times.

Since the initial distance between a^* and an agent in A is at most $n-1$, it follows such a distance increases less than $n-1$ (due to a^* moving), but it decreases by $2n+1$ (due to a^* being blocked); thus the distance is zero (i.e., they are at the same node) by the end of interval I. □

Because of absence of chirality, the set $\mathcal{A}$ of agents can be partitioned into two sets where all the agents in the same set share the same orientation of the ring; let A_r and A_l be the two sets.

Lemma 2. *Let $A \in \{A_r, A_l\}$. If at round $6n$* Pred *is satisfied for an agent $a^* \in A$, then all agents in A are in the same group at round $6n$. Moreover,* Pred *is satisfied for all agents in A.*

Proof. By definition of $Pred$ and by Lemma 1, at round $6n$ all agents in A are at the same node of a^*. Also, let r be the first round when all agents in A meet at the same node: by definition, the value of r_{ms} for all agents under consideration is exactly r. From this observation and since $Pred$ holds for a^*, it follows that $Pred$ must be satisfied for all agents in A. □

Lemma 3. *Let $A \in \{A_r, A_l\}$. If* Pred *is not satisfied at round $6n$ for agent $a^* \in A$, then at round $6n$ all agents in A have done a complete tour of the ring (and hence know the number of total agents, k); moreover,* Pred *is not satisfied for all agents in A.*

Proof. Let us assume by contradiction that there exists $a' \in A$ that has not done a complete tour of the ring after $6n$ rounds; that is, a' has moved less than $n-1$ steps in the first $3n-1$ rounds. By Lemma 1, all agents in A are in the same node as a^* by round $r \leq 3n-1$. Therefore, *Pred* would be satisfied for any of the agents in A, including a^*: a contradiction.

To prove the second part of the lemma, note that *Pred* cannot be satisfied for any agent in A: in fact, by Lemma 2, this would prevent the existence of an agent in A for which *Pred* is not satisfied. Thus, the lemma follows. □

Lemma 4. *If one agent terminates in* Phase 1*, then all agents terminate and gathering has been correctly achieved. Otherwise, no agent terminates and all of them have done a complete tour of the ring.*

Proof. Notice that, by construction, the agents do not change their direction before round $6n$.

Let us first consider the case when at round $r = 0$ the agents do not have the same orientation. We distinguish three possible cases, depending on what happens at round $6n$.

1. *At round* $6n$*, all agents change direction.* By Lemma 2, it follows that at round $6n$ all of them completed a loop of the ring. According to SwitchDir, an agent, to enter the Term state, has to verify both (a) $Agents = TotalAgents$ and (b) $\neg$*meetingOppositeDir*: to verify (a), the agents have to meet at the same node, thus *meetingOppositeDir* has to be true, hence (b) can not be satisfied. It follows that the agents cannot terminate at round $12n$, and the lemma follows.
2. *At round* $6n$*, no agent changes direction.* Thus, according to the algorithm, *Pred* is satisfied for all agents, that will enter KeepDir state; also, by Lemma 2, all agents that share the same direction are in the same group (i.e., there are two groups of agents moving in opposite direction).
   By definition of KeepDir, if between round $6n$ and $12n$ an agent crosses or meets another agent, they both terminate; hence, all the agents in their respective group terminate, and the lemma follows. If no crossing occurs between round $6n$ and $12n$, then both group of agents are necessarily blocked at the ends of the missing link (otherwise the two groups would have crossed or met). Thus, at round $12n$, $r_{ms} < 9n$ (last reset of r_{ms} occurred at round $6n$) and $Esteps < n-1$ (otherwise, again, the two groups would have crossed or met), for any agent; hence all agents terminate at round $12n$, and the lemma follows.
3. *At round* $6n$*, only some agents change direction.* By Lemmas 2 and 3, it follows that, after round $6n$, all agents will move in the same direction.
   Let us assume that, at round $12n$, condition $r_{ms} < 9n \wedge Esteps < n-1$ holds for some agent a^*, by a similar argument used in Lemma 2 we can show that the condition holds for all agents. If a^* did not switch direction at round $6n$, a^* terminates at round $12n$, say at node v (KeepDir); hence, by Lemma 1, all agents gather at v. Otherwise, if a^* switched direction at round $6n$, since all agents are moving in the same direction, condition *meetingOppositeDir* is false from round $6n$ on; moreover, by Lemma 3, a^* computed the number k of total agents at round $6n$. Therefore, a^* terminates at round $12n$, say at node v (SwitchDir). Finally, by Lemma 1, all agents gather at v, and the lemma follows.
   On the other hand, if condition $r_{ms} < 9n \wedge Esteps < n-1$ does not hold for any agent at round $12n$, no agent can enter the Term state. Also, following an argument similar to the one used in Lemma 3, we have that all agents have done a complete loop of the ring after $6n$ rounds, and the lemma follows.

The other case left to consider is when at round $r = 0$ the agents have the same orientation. We distinguish two cases.

1. *There is an agent that does not change direction at round* $6n$*.* Then, at this time, all agents are in the same group and none of them switches direction (Lemma 2). Thus, if the agents terminate at round $12n$, gathering is solved, and the lemma follows. Otherwise, by KeepDir, predicate $r_{ms} < 9n \wedge Esteps < n-1$ is not satisfied at round $12n$ for any of them (they are all in the same group) and they have all done a complete loop of the ring (last reset of r_{ms} occurred at round $6n$, hence $Esteps \geq n$ for all agents), so they start Phase 2, and the lemma follows.
2. *There is an agent that switches direction at round* $6n$*.* Then, at this time, all of them switch direction, and have done a complete loop of the ring (Lemma 3). The proof follows with an argument similar to the one of previous case. □

4.1.2. *Algorithm* Gather(Cross, ¢hir)*:* Phase 2

If the agents execute Phase 2 then, by Lemma 4, they know both the position of all the homebases and the number of agents k; that is, they know the initial configuration C. If $C \in \mathcal{P}$, gathering is impossible (Property 5) and they become aware of this fact. Otherwise, if $C \in \mathcal{E}$ they can elect an edge e_L, and if $C \in \mathcal{C} \setminus (\mathcal{P} \cup \mathcal{E})$ they can elect one of the nodes as leader v_L (Property 1). For simplicity of exposition and without loss of generality, in the following we assume that Phase 2 of the algorithm, shown in Fig. 3, starts at round 0.

In Phase 2, an agent first resets all its local variables, with the exception of $TotalAgents$, that stores the number of agents k; between rounds 0 and $3n$, each agent moves toward the elected edge/node following the shortest path (shortestPathDirectionElected()). If at round $3n$ an agent has reached the elected node or an endpoint of the elected edge, it stops and enters the ReachedElected state. Otherwise (i.e., at round $3n$, the agent is not in state ReachedElected), it switches to the ReachingElected state. If all agents are in the node ($Agents = TotalAgents$), they terminate. If they do not terminate, all agents start moving: the ReachingElected agents continuing in the same direction, while the ReachedElected

```
States: {Phase 2, ReachedElected, ReachingElected, Joining, Waiting, ReverseDir, Term}.
In state Phase 2:
    if $C \in \mathcal{P}$ then
        unsolvable()
        Go to State Term
    resetAllVariables except $TotalAgents$
    $dir$ = shortestPathDirectionElected()
    EXPLORE ($dir$ | $seeElected$: ReachedElected; $Ttime = 3n$: ReachingElected)
In state ReachedElected:
    $dir$ = opposite(dir)
    if $Ttime \geq 3n$ then
        EXPLORE ($dir$ | $Agents = TotalAgents \vee Btime = 2n$: Term; $crossed$: Joining)
In state Joining:
    $dir$ = opposite(dir)
    EXPLORE ($dir$ | $Agents = TotalAgents \vee Btime = 2n \vee crossed$: Term; $Esteps = 1$: ReverseDir)
In state ReachingElected:
    EXPLORE ($dir$ | $Agents = TotalAgents \vee Btime = 2n$: Term; $crossed$: Waiting; $meetingSameDir$: ReachedElected; $meetingOppositeDir \vee seeElected$:
ReverseDir )
In state Waiting:
    EXPLORE (nil | $Etime > 2n$: Term; $meeting$: ReverseDir)
In state ReverseDir:
    $dir$ = opposite(dir)
    Go to State ReachedElected

Algorithm GATHER(CROSS, ⊄HIR), Phase 2.
```

Fig. 3. Phase 2 of Algorithm GATHER(CROSS, ⊄HIR).

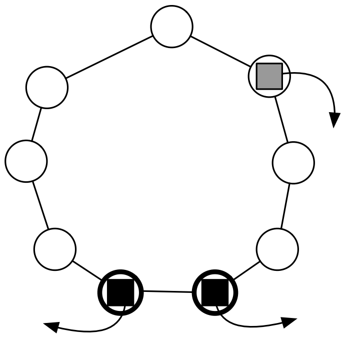


Fig. 4. Phase 2, round $3n$: in this case we have three groups of agents. The bold nodes are the endpoints of the leader edge. The black square are groups with state ReachedElected, the grey square with state ReachingElected. The movement direction of each group is indicated by the arrow.

agents reverse direction. In Lemma 5, we will show that at round $3n$ the agents are partitioned in at most three groups (see Fig. 4).

After round $3n$, each agent, regardless of its state, terminates immediately if all k agents are in the same node, or if it is blocked on a missing edge for $2n$ rounds. In other situations, the behavior of each agent a^* depends on its state, as follows.

State ReachedElected If a^* crosses a group of agents, it enters the Joining state. In this new state, say at node v, the agent switches direction in the attempt to catch and join the agent(s) it just crossed. If a^* leaves v without crossing any agent ($Esteps = 1$), a^* enters again the ReachedElected state, switching again direction (i.e., it goes back to direction originally chosen when Phase 2 started). If instead a^* leaves v and it crosses some agents, it terminates: this can happen because also the agents that a^* crossed try to catch it (and all other agents in the same group with a^*). As we will show, in this case all agents can correctly terminate.

State ReachingElected if a^* crosses someone, it enters the Waiting state, and it stops moving. If while in the Waiting state a^* meets someone new before $2n$ rounds, it enters the ReachedElected state and switches direction. Otherwise, at round $2n$ round it terminates.

If a^* is blocked on a missing edge and it is reached by other agents, then it switches state to ReachedElected keeping its direction ($meetingSameDir$ is satisfied).

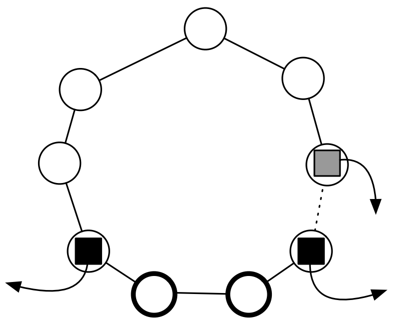


Fig. 5. Phase 2, agents blocked by an edge removal. The bold nodes are the endpoints of the leader edge. The black square are groups with state ReachedElected, the grey square with state ReachingElected. The movement direction of each group is indicated by the arrow. The dashed edge is an edge removed. Notice, that if the edge keeps being removed for $2n$ rounds the three groups will gather around it.

Finally, If a^* is able to reach the elected node/edge (*seeElected* is satisfied), it enters the ReachedElected state, and switches direction. Notice that, this does not happen if a^* crossed someone before reaching the elected node/edge, in which case it immediately enters in Waiting state.

Lemma 5. *At round* $3n$ *of* Phase 2*, there is at most one group of agents in state ReachingElected, and at most two groups of agents in state ReachedElected.*

Proof. In Phase 2, all agents start moving towards the nearest elected endpoint/node. The lemma clearly follows for agents in state ReachedElected: in fact, the two groups (one of them possibly empty) are formed by all the agents that have successfully reached the elected endpoint/node from each of the two directions.

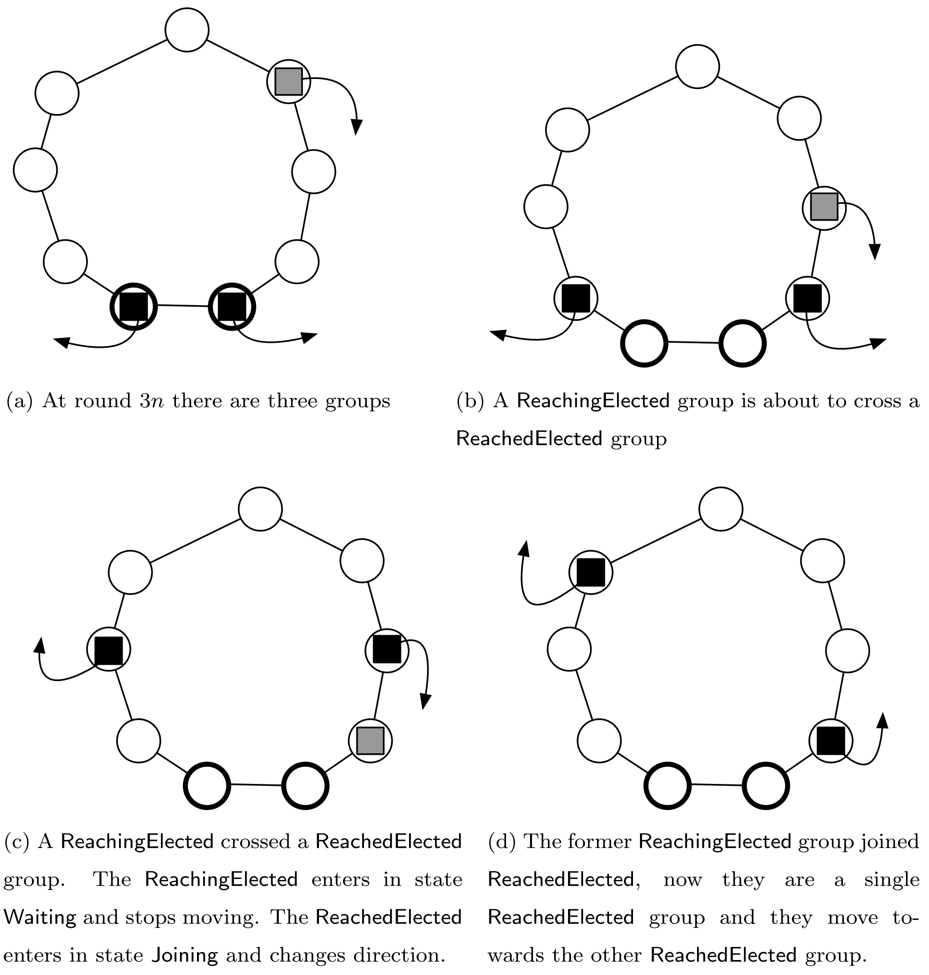
If an agent is not able to reach the elected endpoint/node within $3n$ rounds, it must have been blocked for at least $2n + 1$ rounds; notice that this cannot happen to two agents walking on disjoint paths toward the elected endpoint/node. Therefore, by Lemma 1, there can be at most one group of agents in state ReachingElected, and the lemma follows. □

Note that, if at round $3n$ there are two groups of agents in state ReachedElected, they have opposite moving directions; also, they are either at the same leader node, or at the two endpoints of the leader edge.

Lemma 6. *If an agent* a^* *terminates executing* Phase 2*, then all other agents will terminate, and gathering is correctly achieved.*

Proof. If a^* terminates because $Agents = k$, the lemma clearly follows. Let us consider the other termination conditions.

1. *a^* is either in state ReachedElected or ReachingElected, and $Btime = 2n$.* Agent a^* is blocked on one endpoint of the missing edge; thus, after $2n$ steps, all agents with opposite direction are on the other endpoint of the missing edge. Notice, that $Btime$ is reset every time an agent with the same direction reaches the node where a^* is blocked, therefore if $Btime = 2n$ all agents with the same direction of a^* are waiting on the same endpoint of a^*. Note that this holds also if the other agents are all in the ReachingElected state and reach the elected endpoint/node in (at most) n rounds: in this case, in fact, they would switch direction, and go back to the other endpoint of the missing edge in at most other n steps, see Fig. 5 for an example.
   Therefore, the other agents will either terminate because they wait for $2n$ rounds at the other endpoint of the missing edge, or because they reach the same endpoint node where a^* terminated ($Agents = TotalAgents$ is thus satisfied); hence they correctly gather, and the lemma follows.
2. *a^* is in state* Joining *and crossed is satisfied.* First notice that, if a^* crosses some agent(s), then the crossed agent(s) are in state Joining as well (the agents in the Waiting state do not try to actively cross an edge); thus, they were in the ReachedElected state before crossing. However, this is possible only if there is no group of agents in state ReachingElected: at round $3n$, the two groups ReachedElected start moving in opposite directions from the same node or from two endpoints of the same edge. Therefore, when they cross, one of them has already met the group ReachingElected, if it exists, and when that happens the group ReachingElected merges with the group ReachedElected. This implies that, when two groups ReachedElected cross, all agents are in Joining. Therefore, when they cross again, all agents are on the two endpoints of the same edge, and the lemma follows.
3. *a^* is in state* Waiting*, and $Etime > 2n$.* By Lemma 5, a^* has crossed a group of agent in state ReachedElected. These agents, by entering the Joining state, actively try to reach the node where a^* is (in Waiting). If the Joining group does not reach a^* in $2n$ rounds, then the edge connecting them is necessarily missing. Also note that, if there is another ReachedElected group, it has to reach the agents in the Joining state within $2n$ rounds. Now, these two groups will either terminate by waiting $2n$ rounds, or because they are able to reach the Waiting agent a^*, finally detecting that $Agents = TotalAgents$. In all cases, the agents correctly terminate solving the gathering, and the lemma follows. □


(a) At round $3n$ there are three groups

(b) A ReachingElected group is about to cross a ReachedElected group

(c) A ReachingElected crossed a ReachedElected group. The ReachingElected enters in state Waiting and stops moving. The ReachedElected enters in state Joining and changes direction.

(d) The former ReachingElected group joined ReachedElected, now they are a single ReachedElected group and they move towards the other ReachedElected group.

Fig. 6. Example run of Phase 2 starting from round $3n$, the executions goes from Figure (a) to (c) in order. The bold nodes are the endpoints of the leader edge. The black square are groups with state ReachedElected, the grey square with state ReachingElected. The movement direction of each group is indicated by the arrow.

Lemma 7. Phase 2 *terminates in at most* $10n$ *rounds.*

Proof. By Lemma 5, at the end of round $3n$, the following holds:

1. If there is only one group with state ReachingElected, the agents terminate on condition $Agents = TotalAgents$.
2. If there is only one group with state ReachedElected, the agents terminate on condition $Agents = TotalAgents$.
3. If there are two groups with state ReachedElected, they have opposite direction of movements (otherwise, they would be in the same group). Therefore, within n rounds, they have to be at distance 1 from each other: they terminate within the next $2n + 1$ rounds either by crossing in state Joining, or on condition $Btime = 2n$.
4. If there are two groups of agents in the ReachedElected state, say G and G', and one group of agents in the ReachingElected state, say G^*, then G and G' have opposite direction of movements (otherwise, they would be in the same group); hence one of them, say G, has direction of movement opposite to the one of G^*. Therefore, within n rounds, G and G^* have to be at distance 1 from each other. If they do not cross each other within the next $2n$ rounds, they will terminate on condition $Btime = 2n$, and the lemma follows.
   Otherwise (they cross within the next $2n$ rounds), two cases can occur: (A) they both terminate, one group on condition $Btime = 2n$ and the other one on condition $Etime > 2n$ in the Waiting state (between the two groups there is the missing edge); or (B) they will join within the next $2n$ rounds. In Case (A) the lemma follows. In Case (B), they either terminate on condition $Agents = TotalAgents$, and the lemma follows; or the ReachingElected group enters the ReachedElected state (via Waiting), and starts moving towards the other ReachedElected group (For an example of this case see Fig. 6). In this last case, the proof follows from previous Case 3.
5. If there is one group in the ReachedElected state and one in the ReachingElected state, we have two possible cases. (A) The two groups are moving towards each other: in this case the proof follows similarly to the previous Case 3.

(B) The two groups move in the same direction. If the group ReachingElected does not reach the elected endpoint/node within $2n+1$ rounds, the two groups necessarily meet, and thus terminate; hence the lemma follows. Otherwise, after ReachingElected reaches the elected endpoint/node, this group enters the ReachedElected state, and the proof follows similarly to the previous Case 3. □

Hence we have

Theorem 1. *Without chirality,* GATHERING *is solvable in rings of known size with cross detection, starting from any $C \in \mathcal{C} \setminus \mathcal{P}$. Moreover, there exists an algorithm solving* GATHERING *that terminates in $\mathcal{O}(n)$ rounds for any $C \in \mathcal{C} \setminus \mathcal{P}$ and, if $C \in \mathcal{P}$, the algorithm detects that the configuration is periodic.*

Proof. If algorithm GATHER(CROSS, $\not{C}$HIR) terminates in Phase 1 then, by Lemma 4, it correctly solves gathering and it terminates by round $12n$. If it terminates in Phase 2, then by Lemma 6, it correctly solves gathering, and by Lemma 7 will do so in at most 10 additional rounds. Notice, that in Phase 1, either the agents discover the initial configuration C or they gather. Once they know C, they can detect if the problem is solvable or not. This proves the last statement of the theorem □

4.2. *Knowledge of n is more powerful than knowledge of k*

One may ask if it is possible to obtain the same result of Theorem 1 if knowledge of k was available instead of n; recall that at least one of n and k must be known (Property 3). Intuitively, knowing k, if an agent manages to travel all along the ring, it will discover also the value of n. Unfortunately, the following Theorem shows that, from a computational point of view, knowledge of the ring size is strictly more powerful than knowledge of the number of agents.

Theorem 2. *In rings with no chirality,* GATHERING *is impossible without knowledge of n when starting from a configuration $C \in \mathcal{E}$. This holds even if there is cross detection and k is known.*

Proof. By contradiction. Let us suppose to have two agents a and b on a ring R where the distances between the homebases h_1 and h_2 are $d_1 < d_2$ and they are both odd. Let e_1 be the central edge between h_1 and h_2 in the smallest portion of the ring (i.e., at distance $\frac{(d_1-1)}{2}$ from h_1 and h_2) and e_2 the central edge on the other side (i.e., at distance $\frac{(d_2-1)}{2}$ from h_1 and h_2). Let us consider an execution E of a correct algorithm $\mathcal{A}$ starting from this configuration. The adversary decides opposing clockwise orientation for these two agents, and it only removes edges e_1 and e_2 during the execution of the algorithm. We will show that, by appropriately removing only these two edges, the adversary can prevent the two agents to ever see each other. At the beginning the agents move towards each other (w.l.o.g., in the direction of e_1). The adversary lets them move until they are about to traverse edge e_1; at this point edge e_1 is removed and both agents are blocked with symmetric histories. After a certain amount of time, they will either both reverse direction or terminate. The same removal scheduling is taken whenever they are about to cross either e_1 or e_2. The adversary keeps following this schedule until both agents decide to terminate. Notice that for A to be correct they can only terminate on the endpoints of one of the edges e_1 or e_2. Let $r' = f(R)$ be the round when the agents terminate in execution E.

Let us now consider the same algorithm on a ring R' of size greater than $4f(R)+2$ where the two agents are initially placed at distance greater than $2f(R)$. Consider agent a: the adversary removes the edge at distance $\frac{d_1-1}{2}$ on its right and the one at distance $\frac{d_2-1}{2}$ on its left whenever a tries to traverse them. In doing so a does not perceive any difference with respect to execution E, and therefore terminates at round $r' = f(R)$. At this point, the other agent b cannot be at the other extreme of the edge where a terminated, therefore, the adversary now blocks b from any further move, preventing gathering. A contradiction. □

4.3. *With cross detection: with chirality*

Let us now consider the simplest setting, where the agents have cross detection capability as well as a common chirality. In this case, the impossibility result of the previous Section does not hold, and a solution to GATHERING exists also when k is known but n is not.

The solution consists of a simplification of Phase 1 of Algorithm GATHER(CROSS, $\not{C}$HIR), also extended to the case of k known, followed by Phase 2 of Algorithm GATHER(CROSS, $\not{C}$HIR).

4.3.1. *Algorithm* GATHER(CROSS, CHIR)*:* Phase 1

In case of known n, each agent executes Phase 1 of Algorithm GATHER(CROSS, $\not{C}$HIR) moving clockwise until round $6n$ (if not terminating earlier) and then executing Phase 2 of Algorithm GATHER(CROSS, $\not{C}$HIR). By Lemma 2 we know that, if termination did not occur by this round, then the ring has been fully traversed by all agents.

In case k is known (but n is not), each agent moves counterclockwise terminating if the k agents are all at the same node. As soon as it passes by $k+1$ homebases, it discovers n. At this point, it continues to move in the same direction switching to Phase 2 at round $3n+1$ (unless gathering occurs before). In fact, by Lemma 1, we know that, if an

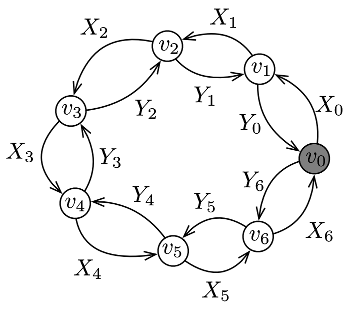


Fig. 7. Example of the *Logic Ring*.

agent does not perform $n-1$ steps in the first $3n-1$ rounds, then all agents are in a single group and, knowing k, they can immediately terminate. This means that after $3n$ rounds, if the agents have not terminated, they have however certainly performed a loop of the ring, they know n (having seen $k+1$ home bases) and they switched to Phase 2 by round $3n+1$.

4.3.2. Algorithm GATHER(CROSS, CHIR)*:* Phase 2

When Phase 2 starts, both n and k are known and Phase 2 of Algorithm GATHER(CROSS, CHIR) is identical to the one of Algorithm GATHER(CROSS, $\not{C}$HIR).

We then have:

Theorem 3. *With chirality, cross detection and knowledge of either n or k,* GATHERING *is solvable in at most $\mathcal{O}(n)$ rounds from any configuration $C \in \mathcal{C} \setminus \mathcal{P}$.* □

5. Without cross detection

In this section we study the gathering problem when there is no cross detection.

We focus first on the case when the absence of cross detection is mitigated by the presence of chirality. We show that gathering is possible in the same class of configurations as with cross detection, albeit with a $O(n \log n)$ time complexity.

We then examine the most difficult case of absence of both cross detection and chirality. We prove that in this case the class of feasible configurations is smaller (i.e., cross detection is a computational separator). We show that gathering can be performed from all feasible configurations in $O(n^2)$ time.

5.1. Without cross detection: with chirality

The structure of the algorithm, GATHER($\not{C}$ROSS,CHIR), still follows the two phases. However, when there is chirality but no cross detection, the difficulty lies in the termination of Phase 2.

5.1.1. Algorithm GATHER($\not{C}$ROSS,CHIR)*:* Phase 1

Notice that the Phase 1 of Algorithm GATHER(CROSS,CHIR) described in Section 4.3 does not really make use of cross detection. So the same Algorithm can be employed in this setting in both cases when n or k are known. Phase 1 terminates then in $\mathcal{O}(n)$ rounds.

5.1.2. Algorithm GATHER($\not{C}$ROSS,CHIR)*:* Phase 2

Because of chirality, a leader node can be always elected, even when the initial configuration is in $\mathcal{E}$ (Property 1). We will show how to use this fact to modify Phase 2 of Algorithm GATHER(CROSS,CHIR) to work without assuming cross detection. We will do so by designing a mechanism that will force the agents *never to cross each other*. The main consequence of this fact is that, whenever two agents (or two groups of agents) would like to traverse the same edge in opposite direction, only one of the two will be allowed to move thus "merging" with the other. This mechanism is described below.

Basic no-crossing mechanism To avoid crossings, each agent constructs an edge labeled bidirectional directed ring with n nodes (called *Logic Ring*) and it moves on the actual ring according to the algorithm, but also to specific conditions dictated by the labels of its *Logic Ring*. Note that, to build the *Logic Ring*, knowledge of n is required. Therefore, in our algorithms, *Logic Ring* will only be used once agents have acquired the knowledge of n.

In the *Logic Ring*, each edge of the actual ring is replaced by two labeled oriented edges in the two directions. The label of each oriented edge e_i, $0 \le i \le n-1$, is either X_i or Y_i, where X_i and Y_i are infinite sets of integers. Labels $X_0, \ldots, X_{n-1}$ are assigned to consecutive edges in counter-clockwise direction starting from the leader node, while $Y_0, \ldots, Y_{n-1}$ are assigned in clockwise direction (see Fig. 7).

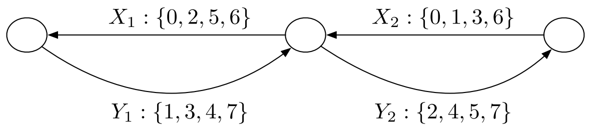


Fig. 8. Example of labels for a ring of size $n = 8$, where $p = 3$. The elements in the set are limited to interval [0, 7]. Notice, that $X_1 \cup Y_1 = \emptyset$ and $X_i \cup Y_j \neq \emptyset$ for any $i \neq j$. Moreover, all X_j contain element 6 and all Y_j contain element 7.

Intuitively, we want to construct these sets of integers associated to labels in such a way that X_i and Y_i have an empty intersection. In this way, the following meta-rule of movement will prevent any crossing:

An agent is allowed to traverse an edge of the ring at round r only if r is contained in the label associated to the corresponding oriented edge of the Logic Ring.

For this construction, we define $X_i = \{s + m \cdot (2p+2) \mid (s \in S_i \vee s = 2p), \forall m \in \mathbb{N}\}$, where $p = \lceil \log_2 n \rceil$, and S_i is a subset of $\{0, 1, \ldots, 2p-1\}$ of size exactly p (note that there are $\binom{2p}{p} \geq n$ possible choices for S_i). Indeed, there are $2^p = 2^{\lceil \log_2 n \rceil} \geq n$ ways to choose which elements of $\{0, 1, \ldots, p-1\}$ are in S_i; each of these choices can be completed to a set of size p by choosing the remaining elements from the set $\{p, p+1, \ldots, 2p-1\}$. Therefore there are at least n available labels, and we can define the X_i's so that they are all distinct. Then we define Y_i to be the complement of X_i for every i. That is, $X_i \cap Y_i = \varnothing$ and $X_i \cup Y_i = \mathbb{N}$.

By construction, it follows that $|X_i \cap \{0, 1, \ldots, 2p-1\}| = p$, and $|Y_i \cap \{0, 1, \ldots, 2p-1\}| = p$, $\forall i$. As a consequence, if $i \neq j$ and $r\mathbb{N}$, then X_i and Y_j have a non-empty intersection in $\{r, r+1, \ldots, r+2p+1\}$. Furthermore, in this labeling, each X_i contains all integers of the form $2p + r \cdot (2p+2)$, and each Y_i's contains all integers of the form $2p + 1 + r \cdot (2p+2)$. An example of sets for a ring of size 8 is reported in Fig. 8.

The following property is immediate by construction:

Observation 1. *Let $r \in \mathbb{N}$ and let $I = \{r, r+1, \ldots, r+2p+1\}$. Then, X_i and Y_j have a non-empty intersection in I if and only if $i \neq j$, X_i and X_j have a non-empty intersection in I, and Y_i and Y_j have a non-empty intersection in I.*

From the previous observation, it follows that two agents moving following the *Logic Ring* in opposite directions will never cross each other on an edge of the actual ring.

As a consequence of this fact, we can derive a bound on the number of rounds that guarantee two groups of agents moving in opposite direction, to "merge". In the following lemma, we consider the execution of the algorithm proceeding in *periods*, where each period is composed by $2p + 2$ rounds. We have:

Lemma 8. *Let us consider two groups of agents, G and G', moving in opposite directions following the* Logic Ring. *After at most n periods, that is at most $\mathcal{O}(n \log n)$ rounds, the groups will be at a distance $d \leq 1$ (in the direction of their movements).*

Proof. Without loss of generality, let us assume that G and G' are initially positioned on two nodes, respectively v and v', trying to traverse two edges incident to v and v'. If the two edges have labels that are the complement of each other in the *Logic Ring* then, by construction, they are trying to traverse the same edge in the actual ring in opposite directions, and the lemma follows.

Now, let us then assume that the two groups are trying to traverse edges whose labels in the *Logic Ring* are not the complement of each other. Since these sets of labels have a non empty intersection (Observation 1), it follows that, in each period of $2p + 2$ rounds, the adversary can block at most one of the two groups. Thus, there exists a round r in which both groups try to cross two different edges, and at least one of them will succeed, hence moving of one step in the direction of the other group. Therefore, after at most $(n-1)(2p+2)$ rounds the two groups will be at a distance at most one in the directions of their movements. Since each period has $\mathcal{O}(\log n)$ rounds, the lemma follows. □

We are now ready to describe the second Phase of the algorithm.

Phase 2 In the following, when the agents are moving according to the meta-rule in the *Logic Ring*, we will use variable $BPeriods$, instead of $Btime$, indicating the number of consecutive periods in which the agent failed to traverse the current edge. As in the case of $Btime$, the new variable $BPeriods$ is reset each time the agent traverses the edge, changes direction, or encounters new agents in its moving direction.

In the first $3n$ rounds, each agent moves towards the elected node using the minimum distance path. After round $3n$, the agents move on the *Logic Ring* ring: the group in state ReachedElected starts moving in clockwise direction, the group in state ReachingElected in counterclockwise. One of the two groups terminates if $BPeriods \geq n$ rounds or if $Agents = k$. This replaces the terminating condition $Btime = 2n$ that was used in case of Cross detection. Phase 2 of the Algorithm is shown in Fig. 9.

```
States: {ReachedElected, ReachingElected, ChangeDir, ChangeState, DirCommR, DirCommS, Term}.
In state Phase 2:
    if $C \in \mathcal{P}$ then
        unsolvable()
        Go to State Term
    resetAllVariables except $TotalAgents$
    $dir$ = leaderMinimumPath()
    Explore ($dir$ | $seeElected$: ReachedElected; $Ttime = 3n$: ReachingElected)
In state ReachedElected:
    if $Ttime \geq 3n$ then
        $dir$ = clockwiseDirection()
        Explore ($dir$ | ($BPeriods \geq 4n + 8 \vee Agents = TotalAgents$): Term;)
In state ReachingElected:
    if $Ttime = 3n$ then
        $dir$ = counterclockwiseDirection()
    Explore (dir | ($BPeriods \geq 4n + 8 \vee Agents = TotalAgents$): Term;)
```

Fig. 9. Phase 2 of Algorithm Gather(⊄ross,Chir).

Lemma 9. Phase 2 *of Algorithm* Gather(⊄ross,Chir) *terminates in at most* $\mathcal{O}(n \log n)$ *rounds, solving the* Gathering *problem.*

Proof. Let us first prove that the algorithms terminates in $\mathcal{O}(n \log n)$ rounds. At the end of round $3n$ of Phase 2, we have at most one group of agents in state ReachedElected and one group in state ReachingElected (Lemma 5 derived in the case with cross-detection still holds). If there is only one of these groups, termination is immediate from condition $Agents = k$. If both groups are present (moving in opposite direction by construction) we have that, by Lemma 8 the two groups will be at distance 1 by at most round $3n + n(2p + 2)$, where p is a quantity bounded by $\mathcal{O}(\log n)$. At this point, they either meet in one node because only one of the two group will be allowed to cross the edge, and therefore they terminate by condition $Agents = k$, or they are blocked by the adversary on two endpoints of the same edge. In this case, however, they will terminate e by condition $BPeriods \geq n$. Notice that, if a group G terminates by $BPeriods \geq n$ gathering will be achieved, because by Lemma 8, we have that the other group G' is at the other endpoint of the edge where G has been blocked. Therefore, G' either terminates by condition $BPeriods \geq n$, or it reaches the node where G is and it terminates by condition $Agents = k$. □

From the previous Lemma, and the correctness of Phase 1 already discussed in Section 4.3, the next theorem immediately follows.

Theorem 4. *With chirality and knowledge of n or k,* Gathering *is solvable from any configuration* $C \in \mathcal{C} \setminus \mathcal{P}$*. Moreover, there exists an algorithm solving* Gathering *that terminates in* $\mathcal{O}(n \log n)$ *rounds for any* $C \in \mathcal{C} \setminus \mathcal{P}$*, if* $C \in \mathcal{P}$ *the algorithm either solves* Gathering *or it detects that the configuration is in* $\mathcal{P}$*.* □

5.2. Without cross detection: without chirality

In this section, we consider the most difficult setting when neither cross detection nor chirality are available. We show that in this case Gathering is impossible if $C \in \mathcal{E}$. On the other hand, we provide a solution for rings of known size from any initial configuration $C \in \mathcal{C} \setminus (\mathcal{P} \cup \mathcal{E})$, which works in $\mathcal{O}(n^2)$ rounds. We start this Section with the impossibility result.

5.2.1. Impossibility for $C \in \mathcal{E}$

Theorem 5. *Without chirality and without cross detection,* Gathering *is impossible when starting from a configuration* $C \in \mathcal{E}$*. This holds even if the agents know C (which implies knowledge of n and k).*

Proof. By contradiction. Consider an initial configuration C with two agents a_l, a_r, a unique axis of symmetry passing through edges e_u, e_d, and where the two homebases h_l, h_r are at distance at least 4 from e_u and 5 from e_d (see an example in Fig. 10). Let A be an algorithm that solves gathering starting from configuration C in an execution E where the adversary does not remove any edge. Note that, because of symmetry, without edge removals the two agents can cross each other only over e_u or e_d never meeting in the same node at the same time, so gathering could be achieved only on the two endpoint of one of these edges. Let us suppose, w.l.o.g., that A terminates when the two agents are on the endpoints of edge $e_u = (v_{u_1}, v_{u_2})$. Let v'_{u_2} be the neighbor of v_{u_2} different from v_{u_1} (resp. v'_{u_1} the neighbor of v_{u_1} different from v_{u_2}). Let r_f be the round in which a_l reaches v_{u_1} and terminates (note that a_l could have passed by v_{u_1} several times before, without terminating; let r_1, possibly equal to r_f, be the first round when a_l reaches v_{u_1}). Agent a_l may reach v_{u_1} at round

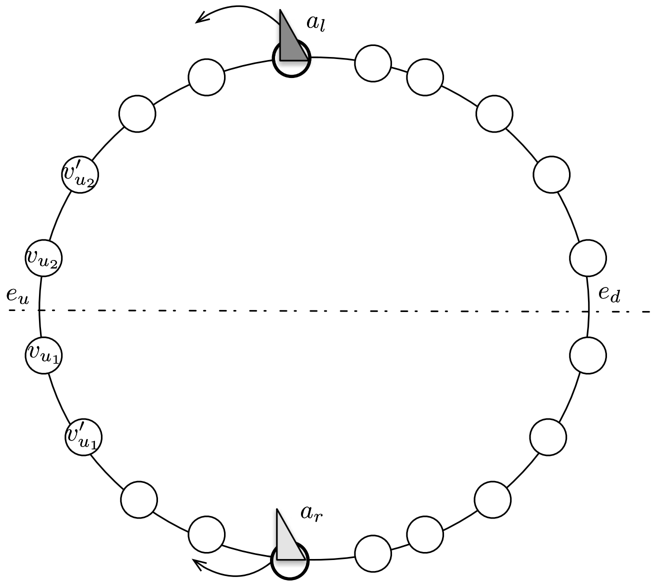


Fig. 10. Configuration used to prove the impossibility of Gathering when the configuration is in $\mathcal{E}$ and there is no cross detection.

r_f in two ways: *Case 1)* after performing a loop of the ring starting from v_{u_1} (note that, during the loop, the agents may go back and forth over some nodes several times). *Case 2)* after moving in a certain direction for X step and then back for other X step, possibly moving back and forth over some nodes several times. In either case, agent a_r does exactly the symmetric moves of a_l with respect to the symmetry axis.

Let us now consider an execution E' starting from C where the agents behave like in execution E until they possibly find themselves blocked by an edge removal. We will show that the edge removal schedule chosen by the adversary does not influence agent a_l, which behaves exactly as in execution E terminating in node v_{u_1} at round r_f, but gathering is not achieved.

No edge removal is done on the way of agent a_l until it terminates in node v_{u_1}. If a_l does so by looping around the ring (*Case 1*), also a_r is performing an opposite loop and the adversary blocks a_r, on an endpoint of e_d, after the agents cross each other, for the last time, on e_d during their loop. Regardless of the decision taken by a_r at this point, when a_l terminates, a_r is at least two edges apart. If a_l is reaching v_{u_1} after moving for X steps and coming back (*Case 2*), a_r is performing the symmetric moves and the adversary behaves differently depending on various sub-cases. *Case 2.1)* Assume first that a_l (resp. a_r) leaves the set of nodes $\{v_{u_1}, v_{u_2}, v'_{u_2}\}$ (resp. $\{v_{u_2}, v_{u_1}, v'_{u_1}\}$) at least once after round r_1. In this case, if the agents do not traverse e_d or e_u, then the adversary blocks a_r the last time it leaves node v_{u_2}; if instead they traverse e_d, then a_l, a_r cross on e_d and the adversary blocks a_r on an endpoint of e_d after their last cross. Finally, if the agents cross each other on e_u, then the adversary blocks a_r as soon as it moves from v'_{u_1}. In all these situations, when a_l reaches v_{u_1}, a_r is at least two edges apart. *Case 2.2)* Assume now that a_l never leaves the set of nodes $\{v_{u_1}, v_{u_2}, v'_{u_2}\}$ after round r_1. The adversary blocks agent a_r right before it is entering for the first time in the set of nodes $\{v_{u_2}, v_{u_1}, v'_{u_1}\}$, this would be undetectable by a_l, and, by construction, a_r would be at distance at least 2 from v_{u_1}, when a_l terminates.

Being run E' undistinguishable for a_l from the execution E, we have that, in E', a_l terminates on v_{u_1}, while agent a_r is not on a neighbor node of v_{u_1}. At this point the adversary blocks a_r from any further move and gathering will never be achieved. A contradiction. □

5.2.2. *Algorithm* Gather($\not\subset$ross, $\not\subset$hir)*:* Phase 1

As we know, the lack of cross detection is not a problem when there is a common chirality. However, the combined lack of both cross detection and chirality significantly complicates Phase 1, and new mechanisms have to be devised to insure that all agents finish the ring exploration and correctly switch to Phase 2. The new algorithm for Phase 1 is in Fig. 11.

In the following we will denote by $Btime'$ the value of $Btime$ at the previous round, that is at round $Ttime - 1$.

Each agent attempts to move along the ring in its own left direction. An agent terminates in the Init state if it has been blocked long enough ($Btime \geq 2n + 2$), or if it was blocked for an appropriate amount of time and it is now meeting a new agent ($Btime' \geq n + 1 \wedge meeting$). It is easy to see that if an agent is blocked on an edge for $2n + 2$ rounds, then any other agent, going in either directions, will be also blocked on the same edge. Moreover, any agent has been blocked for at least $n + 1$ rounds. If an agent does not terminate by round $(3n + 1)(n + 3)$, it enters the *sync sub-phase*, that lasts $2n$ rounds; this synchronization step is used to ensure that, if a group of agents terminates in the Init state by condition $Btime \geq 2n + 2$, all the remaining active agents will correctly terminate in this sub-phase.

An agent with $Btime > n$ starts the *sync sub-phase* in the KeepState state; in this case (and only in this case) it does not reset $Btime$ (see line $Btime = Btime' + 1$ in the pseudocode). Otherwise, an agent starts in state SyncL. In the following

```
States: {Init, KeepState,SyncR, SyncL, Term}.
In state Init:
    EXPLORE ($left$ | $(Ttime \geq (3n+1)(n+3)) \wedge (Btime > n)$: KeepState; $Ttime \geq (3n+1)(n+3)$: SyncL; $Btime \geq (2n+2) \vee (Btime' \geq n+1 \wedge meeting)$: Term)
In state KeepState:
    $Btime = Btime' + 1$
    EXPLORE ($left$ | $(Ttime \geq (3n+1)(n+3)+2n+1) \vee Agents = TotalAgents$: Term; $Esteps = 1$: SyncL)
In state SyncL:
    EXPLORE ($left$ | $Agents = TotalAgents$: Term; $Ttime \geq (3n+1)(n+3)+2n+1$: Phase 2; $Btime = 1$: SyncR)
In state SyncR:
    EXPLORE ($right$ | $Agents = TotalAgents$: Term; $Ttime \geq (3n+1)(n+3)+2n+1$: Phase 2; $Btime = 1$: SyncL)
```

Fig. 11. Phase 1 of Algorithm GATHER($\not\subset$ROSS, $\not\subset$HIR).

rounds, when an agent becomes blocked ($Btime = 1$) in state SyncR (resp. SyncL), if it ever occurs, it switches direction, changing state to SyncL (resp. SyncR). The agent terminates if it either detects k agents at its current node, or if it never moved ($Esteps < 1$) until round $(3n+1)(n+3)+2n+1$ while being in state KeepState; otherwise, at round $(3n+1)(n+3)+2n+1$, it starts Phase 2.

Observation 2. *If an edge is missing for $3n+1$ consecutive rounds, between rounds 0 and $(3n+1)(n+3)$, then all agents terminate. Therefore, if an agent has not terminated by round $(3n+1)(n+3)$, then it has done a complete tour of the ring.*

Lemma 10. *If an agent terminates during* Phase 1, *then all agents terminate and* GATHERING *is correctly solved.*

Proof. The proof proceeds by considering all possible cases when an agent a^* can terminate during Phase 1.

If a^* terminates at a round $r \leq (3n+1)(n+3)$, then it is blocked on a missing edge, say at node v. Also, by definition of state Init, either condition $Btime \geq 2n+2$ or $Btime' \geq n+1 \wedge meeting$ is satisfied by a^* at round r.

- If $Btime \geq 2n+2$ is satisfied at round r, then all agents with the same direction of movement of a^* are terminated as well at v, and for all of them $Btime \geq 2n+2$ is satisfied. Let us consider the agents with direction of movement opposite to that of a^*. If there is no such agent, then the lemma clearly follows. Otherwise, they form a group, call it G, on the other endpoint of the missing edge. Note that, at round r, the agents in G have been blocked for at least $2n+2-(n+1) = n+3$ rounds, hence, at round r, for the agents in G, (**) $Btime \geq n+3$ is satisfied.
  If the agents in G are terminated at round r, then the lemma follows. Otherwise, at round r, for the agents in G, $Btime' \geq n+1$ is satisfied (see (**)). If the agents in G do not change state, and if the edge is missing for the next $n+1$ rounds, then the agents in G terminate on condition $Btime \geq 2n+2$. Otherwise (i.e., if the edge comes alive within the next $n+1$ rounds) the agents in G will cross it, meet the (terminated) agents in v, and terminate as well. Thus, gathering is correctly achieved, and the lemma follows.
  On the other hand, if the agents in G switch to the KeepState state, two cases can occur: (a) the edge is missing for the next $2n$ rounds: in this case, the agents in G terminate on condition $Ttime \geq (3n)(n+3)+2n+1$ by remaining in state KeepState; (b) the edge comes alive within the next $2n$ rounds: in this case, the agents in G cross it and meet all the other (terminated) agents in v. In both cases, the gathering is correctly achieved, and the lemma follows.
- If $Btime' \geq n+1 \wedge meeting$ is satisfied at round r, then all agents with the same direction of movement of a^* are terminated as well at v, and for all of them $Btime' \geq n+1 \wedge meeting$ is satisfied. Let us consider the agents with direction of movement opposite to that of a^*. They form a group, call it G, on the other endpoint of the missing edge.
  First note that, since $meeting$ is satisfied at round r, at the previous round $r-1$, a^* was at v's previous node (according to the chirality of a^*), say v_{i-1}. Moreover, since $Btime' \geq n+1$ is satisfied, the edge between v_{i-1} and v is missing at round $r-1$, and the agents in G must have entered in a terminal state by round $r-1$ (notice that the agents in G had enough time to reach the other endpoint and enter the port, therefore if they are not in terminal state at round r, then a cross would have occurred at round r, hence $meeting$ not satisfied); also, the agents in G terminated on condition $Btime \geq 2n+2$. Therefore, at round r, gathering is correctly achieved at v, and the lemma follows.

It remains to prove the correctness of the termination of a^*, say at node v, in a round $r > (3n+1)(n+3)$, that is during the sync sub-phase. In this case, by Observation 2, all agents know k. The correctness of the termination when condition $Agents = k$ holds is trivial. Thus, let us consider the case when termination occurs because $Ttime \geq (3n+1)(n+3)+2n+1$ holds and the agent is in state KeepState.

Let G be the group of agents that terminates because of this condition. Notice that no agent in G can leave the KeepState state. In fact, any agent that enters state SyncL (or SyncR) at any time during the sub-phase will never have $Btime > n$ for the rest of the sub-phase. This implies that the edge on which G terminates has been missing for the whole execution of the sub-phase.

At round $(3n)(n+3)+n$ all agents with direction of movements opposite to the one of the agents in G are in another group G' on the other endpoint of the missing edge. If the agents in G' are already terminated, the termination of the agents in G correctly solves gathering, and the lemma follows.

Otherwise, if the agents in G' are not terminated, but they are also in the KeepState state, then they will also terminate because of condition $Ttime \geq (3n+1)(n+3)+2n+1$; therefore gathering is correctly achieved, and the lemma follows.

Otherwise, the agents are either in state SyncL or SyncR: in this case, they will change direction at round $(3n+1)(n+3)+n+1$, and in the following n rounds they will move towards G, thus reaching group G. When this occurs, gathering will be correctly achieved (because of condition $Agents = k$), and the lemma follows. □

5.2.3. *Algorithm* GATHER(⊄ROSS, ⊄HIR)*:* Phase 2

By Lemma 10, at the end of Phase 1 each agent knows the current configuration. Since we know that the problem is not solvable for initial configurations $\mathcal{C} \in \mathcal{E}$ (Theorem 5), the initial configuration must be non-symmetric (i.e., without any axis of symmetry) or symmetric but with the unique axis of symmetry going through a node. In both cases, the agents can agree on a common chirality. In fact, if $\mathcal{C}$ does not have any symmetry axes, the agents can agree, for example, on the direction of the lexicographically smallest sequence of homebases inter distances. If instead there is an axis of symmetry going through a node v_L, they can agree on the direction of the port of v_L with the smallest label. We can then use as Phase 2 the one of Algorithm GATHER(⊄ROSS,CHIR) presented in Section 5.1.2.

Theorem 6. *Without chirality,* GATHERING *is solvable in rings of known size without cross detection from all* $C \in \mathcal{C} \setminus (\mathcal{P} \cup \mathcal{E})$*. Moreover, there exists an algorithm that, for any* $C \in \mathcal{C}$*, terminates in* $\mathcal{O}(n^2)$ *rounds. If* $C \notin \mathcal{P} \cup \mathcal{E}$*, it solves* GATHERING*; otherwise it either solves* GATHERING *or it detects that the configuration is in* $\mathcal{P} \cup \mathcal{E}$*.*

Proof. The correctness and the $\mathcal{O}(n^2)$ bound of Phase 1 follow by Lemma 10. The correctness and complexity of Phase 2 follow by Lemma 9 (Section 5.1.2). The last statement of the theorem is obvious by Lemma 10: if at the end of Phase 1 GATHERING is not solved, then the agents know $\mathcal{C}$; therefore they can detect if the configuration is in $\mathcal{P} \cup \mathcal{E}$, and gather otherwise. □

6. Concluding remarks and open questions

In this paper we started the investigation of gathering in dynamic graphs, and studied its feasibility in a dynamic ring of anonymous nodes; the class of dynamics we considered is the classic 1-interval-connectivity.

We provided a complete characterization of the impact that chirality and cross detection have on the solvability of the problem establishing several results. The feasibility results of the characterization are all constructive. The algorithms for gathering with cross detection are time optimal. The ones without cross detection allow the agents to gather within low polynomial time; an interesting open question is whether the current bounds $O(n \log n)$ (with chirality) and $O(n^2)$ (without chirality) might be improved.

This work poses several major open questions. In particular: Are there more factors, other than chirality and cross-detection, that might have a computational impact on the feasibility of gathering? What happens to gathering in 1-interval connected graphs with a more complex topology (e.g., torus, hypercube, etc.)? What happens, even in rings, in presence of different assumptions on the dynamics?

Acknowledgements

This research has been supported in part by NSERC under the Discovery Grant program, and by Dr. Flocchini's University Research Chair.

References

[1] S. Alpern, S. Gal, The Theory of Search Games and Rendezvous, Kluwer, 2003.
[2] L. Barrière, P. Flocchini, P. Fraigniaud, N. Santoro, Rendezvous and election of mobile agents: impact of sense of direction, Theory Comput. Syst. 44 (3) (2007) 143–162.
[3] V. Baston, S. Gal, Rendezvous search when marks are left at the starting points, Naval Res. Logist. 38 (1991) 469–494.
[4] M. Biely, P. Robinson, U. Schmid, M. Schwarz, K. Winkler, Gracefully degrading consensus and k-set agreement in directed dynamic networks, in: Proc. 2nd International Conference on Networked Systems, NETSYS, 2015, pp. 109–124.
[5] S. Bouchard, Y. Dieudonne, B. Ducourthial, Byzantine gathering in networks, Distrib. Comput. 29 (6) (2016) 435–457.
[6] M. Bournat, A. Datta, S. Dubois, Self-stabilizing robots in highly dynamic environments, in: Proc. 18th International Symposium on Stabilization, Safety, and Security of Distributed System, SSS, 2016, pp. 54–69.
[7] A. Casteigts, P. Flocchini, B. Mans, N. Santoro, Measuring temporal lags in delay-tolerant networks, IEEE Trans. Comput. 63 (2) (2014) 397–410.
[8] A. Casteigts, P. Flocchini, W. Quattrociocchi, N. Santoro, Time-varying graphs and dynamic networks, Int. J. Parallel Emergent Distrib. Syst. 27 (5) (2012) 387–408.
[9] J. Chalopin, S. Das, N. Santoro, Rendezvous of mobile agents in unknown graphs with faulty links, in: Proc. 21st International Symposium on Distributed Computing, DISC, 2007, pp. 108–122.
[10] M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro, Distributed computing by mobile robots: gathering, SIAM J. Comput. 41 (4) (2012) 829–879.

[11] R. Cohen, D. Peleg, Convergence properties of the gravitational algorithm in asynchronous robot systems, SIAM J. Comput. 34 (2005) 1516–1528.
[12] J. Czyzowicz, S. Dobrev, E. Kranakis, D. Krizanc, The power of tokens: rendezvous and symmetry detection for two mobile agents in a ring, in: Proc. 34th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM, 2008, pp. 234–246.
[13] J. Czyzowicz, A. Labourel, A. Pelc, How to meet asynchronously (almost) everywhere, ACM Trans. Algorithms 8 (4) (2012) 37:1–37:14.
[14] S. Das, F.L. Luccio, R. Focardi, E. Markou, D. Moro, M. Squarcina, Gathering of robots in a ring with mobile faults, in: Proc. 17th Italian Conference on Theoretical Computer Science, ICTCS, 2016, pp. 122–135.
[15] S. Das, F.L. Luccio, E. Markou, Mobile agents rendezvous in spite of a malicious agent, in: Proc. 11th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS, 2015, pp. 211–224.
[16] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, U. Vaccaro, Asynchronous deterministic rendezvous in graphs, Theoret. Comput. Sci. 355 (2006) 315–326.
[17] B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, R. Wattenhofer, A tight runtime bound for synchronous gathering of autonomous robots with limited visibility, in: Proc. 23rd ACM Symposium on Parallelism in Algorithms and Architectures, SPAA, 2011, pp. 139–148.
[18] G.A. Di Luna, R. Baldoni, Brief announcement: investigating the cost of anonymity on dynamic networks, in: Proc of the 34th Symposium on Principles of Distributed Computing, PODC, 2015, pp. 339–341.
[19] G.A. Di Luna, R. Baldoni, S. Bonomi, I. Chatzigiannakis, Counting in anonymous dynamic networks under worst-case adversary, in: Proc. of the IEEE 34th International Conference on Distributed Computing Systems, ICDCS, 2014, pp. 338–347.
[20] G.A. Di Luna, S. Dobrev, P. Flocchini, N. Santoro, Live exploration of dynamic rings, in: Proc. 36th IEEE International Conference on Distributed Computing Systems, ICDCS, 2016, pp. 570–579.
[21] G.A. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, G. Viglietta, Gathering in dynamic rings, in: Proc. of the 24th International Colloquium on Structural Information and Communication Complexity, SIROCCO, 2017, pp. 339–355.
[22] S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro, Multiple agents rendezvous in a ring in spite of a black hole, in: Proc. 7th International Conference on Principles of Distributed Systems, OPODIS, 2003, pp. 34–46.
[23] P. Flocchini, E. Kranakis, D. Krizanc, F.L. Luccio, N. Santoro, Sorting multisets in anonymous asynchronous rings, J. Parallel Distrib. Comput. 64 (2) (2004) 254–265.
[24] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk, Multiple mobile agent rendezvous in the ring, in: Proc. 6th Latin American Conference on Theoretical Informatics, LATIN, 2004, pp. 599–608.
[25] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility, Theoret. Comput. Sci. 337 (1–3) (2005) 147–168.
[26] P. Flocchini, N. Santoro, G. Viglietta, M. Yamashita, Rendezvous with constant memory, Theoret. Comput. Sci. 621 (2016) 57–72.
[27] B. Haeupler, F. Kuhn, Lower bounds on information dissemination in dynamic networks, in: Proc. 26th International Symposium on Distributed Computing, DISC, 2012, pp. 166–180.
[28] D. Ilcinkas, R. Klasing, A. Wade, Exploration of constantly connected dynamic graphs based on cactuses, in: Proc. 21st Int. Coll. Structural Inform. and Comm. Complexity, SIROCCO, 2014, pp. 250–262.
[29] D. Ilcinkas, A. Wade, Exploration of the t-interval-connected dynamic graphs: the case of the ring, in: Proc. 20th Int. Coll. on Structural Inform. and Comm. Complexity, SIROCCO, 2013, pp. 13–23.
[30] R. Klasing, E. Markou, A. Pelc, Gathering asynchronous oblivious mobile robots in a ring, Theoret. Comput. Sci. 390 (1) (2008) 27–39.
[31] E. Kranakis, D. Krizanc, E. Markou, Mobile agent rendezvous in a synchronous torus, in: 7th Latin American Conference on Theoretical Informatics, LATIN, 2006, pp. 653–664.
[32] E. Kranakis, D. Krizanc, E. Markou, The Mobile Agent Rendezvous Problem in the Ring, Morgan & Claypool, 2010.
[33] E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk, Mobile agent rendezvous problem in the ring, in: Proc. 23rd International Conference on Distributed Computing Systems, ICDCS, 2003, pp. 592–599.
[34] F. Kuhn, N. Lynch, R. Oshman, Distributed computation in dynamic networks, in: Proc. 42nd Symposium on Theory of Computing, STOC, 2010, pp. 513–522.
[35] F. Kuhn, Y. Moses, R. Oshman, Coordinated consensus in dynamic networks, in: Proc. 30th Symposium on Principles of Distributed Computing, PODC, 2011, pp. 1–10.
[36] J. Lin, A. Morse, B. Anderson, The multi-agent rendezvous problem. Parts 1 and 2, SIAM J. Control Optim. 46 (6) (2007) 2096–2147.
[37] L. Pagli, G. Prencipe, G. Viglietta, Getting close without touching: near-gathering for autonomous mobile robots, Distrib. Comput. 28 (5) (2015) 333–349.
[38] A. Pelc, Deterministic rendezvous in networks: a comprehensive survey, Networks 59 (3) (2012) 331–347.
[39] C. Sawchuk, Mobile Agent Rendezvous in the Ring, Ph.D. Thesis, Carleton University, January 2004.
[40] A. Ta-Shma, U. Zwick, Deterministic rendezvous, treasure hunts, and strongly universal exploration sequences, ACM Trans. Algorithms 10 (3) (2014).
[41] X. Yu, M. Yung, Agent rendezvous: a dynamic symmetry-breaking problem, in: Proc. International Colloquium on Automata, Languages, and Programming, ICALP, 1996, pp. 610–621.