Gathering in Dynamic Rings

Giuseppe Antonio Di Luna* Paola Flocchini* Linda Pagli† Giuseppe Prencipe†
Nicola Santoro‡ Giovanni Viglietta*

Abstract

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*. 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 protocols for gathering with cross detection are time optimal.

We also show that cross detection is a powerful computational element. We prove that, without chirality, knowledge of the ring size is strictly more powerful than knowledge of the number 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.

*School Electrical Engineering and Computer Science, University of Ottawa, Canada.
†Dipartimento di Informatica, University of Pisa, Italy
‡School of Computer Science, Carleton University, Canada.

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, known also as *multi-agent rendezvous*, has been intesively and extensively studied in a variety of fields, including operations research (e.g., [1]) and control (e.g., [41]), 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 *continuous* domains, both gathering and rendevous 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 [11, 12, 17, 27, 28, 43]). In *discrete* domains, 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, 18, 19, 24, 25, 32, 35–37, 46, 47]). 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 nodes are uniquely numbered, solving gathering is trivial. On the other hand, when the nodes are anonymous, the network is highly symmetric, the 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*; this enables different agents to execute different deterministic algorithms (e.g., see [13, 18, 19, 47]). 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, 24]). 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., [14]). An assumption much less demanding than agents having identities or explicit communication is that of having the homebases (i.e., the nodes where the agents are initially located) identifiable by an identical mark visible to any agent passing by it; originally suggested in [3], it has been used and studied e.g., in [25, 37, 45]. Summarizing, the existing literature on gathering and rendezvous is extensive and the variety of assumptions and results is aboundant (for surveys see [36, 44]). Regardless of their differences, all these investigations, including those that consider faulty nodes (e.g., see [6, 9, 24]), share the same fundamental assumption that the topological structure does not change during the rendezvous or the gathering. 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; rather, they occur continuously and at unpredictable locations, and are integral part of the nature of the system [8, 40]. The study of distributed computations in dynamic graphs has concentrated on problems of information diffusion, agreement, and exploration (e.g., [4, 5, 7, 29–31, 38, 39]).

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 intesively studied in the static case (e.g., see the monograph on the subject [36]). The presence, in the static case, of a mobile faulty agent that can block other agents, considered in [15, 16], 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

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., [22, 29, 31, 38, 39]); that is, the system is fully synchronous and under a (possibly unfair) adversarial schedule that, at each round, chooses which edge (if any) will be missing. 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, 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 protocols 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.
Due to space limitations, the proofs are omitted; for the full text see the Appendix or [23].

2 Model and Basic Limitations

2.1 Model and Terminology

Let $\mathcal{R} = (v_0, \ldots v_{n-1})$ be a synchronous dynamic ring where, at any round $t \in N$, 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 (e.g., [23, 31]). Each node v_i is connected to its two neighbours 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 labels of the ports are arbitrary elements of a totally ordered set, 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). Operating in $\mathcal{R}$ is

a set $\mathcal{A} = \{a_0, \ldots, a_{k-1}\}$ 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; homebases are marked so that an 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 has its own *left/right* orientation of the ring, but the orientations 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. They 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. 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 in v executes its protocol (the same for all agents). Based on the number of agents at v and in its buffers, and on the content of its local memory and its state, the agent determines 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 destination node in round $r+1$; otherwise it does not leave the outgoing buffer. As a consequence, an agent can be in an outgoing buffer at the beginning of a round only when the corresponding link was not present in the previous round. In the following, when an agents is in an outgoing buffer that leads to the missing edge, we will say that the agent is *blocked*. When multiple agents are at the same node, all of them have the same direction of movement, and are in the same state, we say that they form a *group*. Let $(\mathcal{R}, \mathcal{A})$ denote a system so defined. In $(\mathcal{R}, \mathcal{A})$, gathering is achieved in round r if all agents in A are on the same node or on two neighbouring nodes in 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. Let $h_0, \ldots, h_{k-1}$ denote the nodes corresponding to the marked homebases (in a clockwise order) in $C \in \mathcal{C}$. We shall indicate by d_i $(0 \leq i \leq k-1)$ the distance (i.e., number of edges) between h_i and h_{i+1} (all operations are modulo k). Let δ^{+j} denote the *inter-distance* sequence clockwise $\delta^{+j} =< d_j, d_{j+1} \ \ldots d_{j+k-1} >$, and let δ^{-j} denote the couter-clockwise sequence $\delta^{-j} =< d_{j-1} \ \ldots \ d_{j-(k-1)} >$. The unordered pair of inter-distance sequences δ^{+j} and δ^{-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 $i = 0, \ldots k-1$. Let $\mathcal{P}$ denote the set of periodic configurations. Let $\Delta^+ = \{\delta^{+j} : 0 \leq j < k-1\}$ and $\Delta^- = \{\delta^{-j} : 0 \leq j < k-1\}$. We will denote by δ_{min} the ascending lexicographically minimum sequence in $\Delta^+ \cup \Delta^-$. Among the non-periodic configurations, particular ones are the *double-palindrome* configurations, where $\delta_{min} = \delta^{+i} = \delta^{-j}$ with $i \neq j$, where it is easy to see that 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.

A characterization of the configurations where a leader can be elected depending on chirality is well known in static rings.

Property 1. *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 $C \in \mathcal{C} \setminus \mathcal{P}$.

2.3 Basic Limitations and Properties

The simple properties below motivate the necessity of the following assumptions: identical but distinguishable homebases, knowledge of either n or k, gathering on a node or on an edge.

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

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

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

Finally, the following obvious but important limitation 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.*

3 General Solution Structure

Our algorithms have the same general structure, and use the same building block and variables.

General Structure. All our algorithms are divided in 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 the ring is dynamic. Different strategies are devised, depending on the setting, to guarantee that in finite time the problem is solved in spite of the choice of schedule of missing links decided by the adversary. For each setting, we will describe the two phases depending on the availability or lack of cross detection, as well as on the presence or not of chirality. 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. Predicates and variables used by procedure Explore are indicated in Tables 1 and 2.

Table 1. Variables	Description
r_{ms}	It stores the last round when the agent meets someone (at a node) that is moving in the same direction (initially set to 0); this value 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 set.
Agents	The number of agents at the node of the executing agent. This value is set at each round.

Table 2. Predicate	Description
meeting	Satisfied when the agent (either in a port or at a node) detects an increase in the numbers of agents it sees at each round.
meetingSameDir	Satisfied when the agent detects, in the current round, new agents moving in its same direction. This is done by seeing new agents in an incoming or outgoing buffer corresponding to a direction that is equal to the current direction of the agent.
meetingOppositeDir	Satisfied when the agent detects, in the current round, new agents moving in its opposite direction. This is done by seeing new agents in an incoming or outgoing buffer corresponding to a direction that is opposite to the current direction of the agent.
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	Let us assume there is either an elected node or an elected edge. This predicate is satisfied when the agent has reached the elected node or an endpoint of the elected edge.

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 sovable 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 sovable 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,$\not\exists$CHIR), that solves GATHERING in rings of known size with cross setection but without chirality.

Algorithm GATHER(CROSS,$\not\exists$CHIR)**: Phase 1.** The overall idea of this phase, shown in Figure 1, 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. 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 the predicate $Pred \equiv (r_{ms} < 3n \wedge Esteps < n)$ is verified. If *Pred* is not verified, then (as we show) the agent has explored the entire ring and thus knows the total number k of agents (local variable $TotalAgents$); in this case, the agent switches direction, and enters state SwitchDir. Otherwise, if after $6n$ rounds *Pred* is satisfied, then k is not known yet: in this case, the agent keeps the same direction, and enters state KeepDir. In state SwitchDir, the agent attempts to move in the chosen direction until round $12n$. At round $12n$, the agent terminates if the predicate $[r_{ms} < 9n \wedge Esteps < n]$ holds, predicate *meetingOppositeDir* does not hold, and in its current node there are k agents; otherwise, it starts Phase 2. In state KeepDir, if at round $6n + 1$ predicate

crossed or predicate *meetingOppositeDir* hold, the agent terminates; otherwise, it attempts to move to its left until round $12n$. At round $12n$, if the predicate $[r_{ms} < 9n \wedge Esteps < n]$ holds, the agent terminates; otherwise, it switches to Phase 2.

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* | $Ttime = 12n \wedge r_{ms} < 9n \ \wedge \ Esteps < n \ \wedge \ Agents = TotalAgents \wedge \neg meetingOppositeDir$: Term; $Ttime = 12n$: Phase 2)
In state KeepDir:
EXPLORE (*left* | $crossed \vee meetingOppositeDir$: Term; $Ttime = 12n \wedge r_{ms} < 9n \wedge Esteps < n$: Term; $Ttime = 12n$: Phase 2)

Figure 1: Phase 1 of Algorithm GATHER(CROSS,$\not\exists$CHIR)

We now prove some important properties of Phase 1.

Lemma 1. *Let agent a^* move less than n steps in the first $3n$ rounds. Then, by round $3n$, all agents moving in the same direction as a^* belong to the same group.*

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 verified for an agent $a^* \in A$, then all agents in A are in the same group at round $6n$. Moreover,* Pred *is verified for all agents in A.*

Lemma 3. *Let $A \in \{A_r, A_l\}$. If* Pred *is not verified 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 verified for all agents in A.*

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

Algorithm GATHER(CROSS,$\not\exists$CHIR)**: Phase 2.** When the agents execute Phase 2, by Lemma 4, 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 homebases v_L (Property 1). For simplicity of exposition and w.l.g., in the following we assume that Phase 2 of the algorithm 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 same state (either ReachedElected or ReachingElected), then they are in the same group, and terminate ($Agents = TotalAgents$). If they do not terminate, all agents start moving: each ReachingElected agent in the same direction it chose at the beginning of Phase 2, while the ReachedElected agents reverse direction. From this moment, each agent, regardless of its state, terminates as soon as it perceives k agents in the same node or if it is blocked on a missing edge for $2n$ rounds. In other situations, the behaviour of agent a^* depends on its state, as described below.
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^* is able to reach the elected node/edge (*seeElected* is verified), it enters the ReachedElected state, and switches direction. 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 verified). Finally, 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 $2n$ it terminates.

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; *meetingSameDir*: ReachedElected; *meetingOppositeDir* $\vee$ *seeElected*: ReverseDir; *crossed*: Waiting)
In state Waiting:
  EXPLORE (nil | $Etime > 2n$: Term; *meeting*: ReverseDir)
In state ReverseDir:
  dir =opposite(dir)
  Go to State ReachedElected

Figure 2: Phase 2 of Algorithm GATHER(CROSS,$\not C$HIR)

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.*

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

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

Theorem 1. *Without chirality,* GATHERING *is solvable in rings of known size with cross detection, starting from any $C \in \mathcal{C} \setminus \mathcal{P}$. This can be done in $\mathcal{O}(n)$ rounds fby an effective algorithm.*

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). 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.*

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. If n is known, the problem is already optimally solved by Algorithm GATHER(CROSS,$\not$CHIR). So, we need only to consider the case when k is known but n is not.

Phase 1 of the solution, Algorithm GATHER(CROSS, CHIR), consists of the following modification of Phase 1 of Algorithm GATHER(CROSS,$\not$CHIR), to work with knowledge of k. 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 attempt to move in the same direction switching to Phase 2 at round $3n+1$ (unless gathering occurs before). After $3n$ rounds, if the agents have not terminated, they have however certainly performed a loop of the ring, know n (having seen $k+1$ home bases) and they start Phase 2. Since n is known, the agents can use as Phase 2 the one of Algorithm GATHER(CROSS,$\not$CHIR), which is time optimal. 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 $C \in \mathcal{C} \setminus \mathcal{P}$ with an effective algorithm..*

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 configuration in $O(n^2)$ time.

5.1 Without Cross Detection: With Chirality

The structure of the algorithm, GATHER($\not$CROSS,CHIR), still follows the two Phases. Since Phase 1 of Algorithm GATHER(CROSS,CHIR) does not use cross detection, it can be used as Phase 1 of GATHER($\not$CROSS,CHIR).

Let thus focus on 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 the *Logic Ring*. 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, $0 \leq i \leq n-1$, 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. Intuitively, we want to construct these sets of labels in such a way that X_i and Y_i have an empty intersection, and allow an agent to traverse an edge at round r only if r is contained in the set of labels 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. The following property is immediate by construction:

Observation 1. *Let $m \in \mathbb{N}$ and let $I = \{m, m+1, \ldots, m+2p+1\}$. 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 robots 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).*

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;)

Figure 3: Phase 2 of Algorithm GATHER(ℂROSS,CHIR)

We are now ready to describe the actual Phase 2. In the following, when the agents are moving following 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 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 Figure 3.

Lemma 9. *Phase 2 of Algorithm* GATHER(ℂROSS,CHIR) *terminates in at most $\mathcal{O}(n \log n)$ rounds, solving the* GATHERING *problem.*

Theorem 4. *With chirality and knowledge of n or k,* GATHERING *is solvable from any configuration $C \in \mathcal{C} \setminus \mathcal{P}$. This can be done in $\mathcal{O}(n \log n)$ rounds with an effective algorithm.*

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.

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 (hence n and k).*

Algorithm GATHER($\not C$ROSS,$\not C$HIR)**: Phase 1.** The lack of cross detection is not a problem when there is a common chirality. However, the combination of 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. 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 is now meeting a new agent ($Btime' \geq n+1 \wedge meeting$). If an agent does not terminate by round $(3n)(n + 3)$ it enters a *sync sub-phase*, whose purpose is to perform a synchronization 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 also terminate correctly.

States: {Init, SyncR, SyncL, Term}.
In state Init:
EXPLORE (*left* | $Ttime \geq (3n)(n+3)$: SyncL; $Btime \geq (2n+2) \vee (Btime' \geq n+1 \wedge meeting)$: Term)
In state SyncL:
EXPLORE (*left* | $(Ttime \geq (3n)(n+3)+2n+1 \wedge Btime > n) \vee Agents = TotalAgents$: Term; $Ttime \geq (3n)(n+3)+2n+1$: Phase 2; $0 < Btime \leq n$: SyncR)
In state SyncR:
EXPLORE (*right* | $Agents = TotalAgents$: Term; $Ttime \geq (3n)(n+3)+2n+1$: Phase 2; $Btime = 1$: SyncL)

Figure 4: Phase 1 of Algorithm GATHER($\not C$ROSS,$\not C$HIR)

Lemma 10. *If an agent does not terminate at the end of* Phase 1*, then no agent terminates and all of them have done at least one complete loop of the ring. If an agent terminates during* Phase 1*, then all agents terminate and* GATHERING *is correctly solved.*

Algorithm GATHER($\not C$ROSS,$\not C$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($\not C$ROSS,CHIR) presented in Section 5.1.

Theorem 6. *Without chirality,* GATHERING *is solvable in rings of known size without cross detection from all $C \in \mathcal{C} \setminus (\mathcal{P} \cup \mathcal{E})$. This can be done in $\mathcal{O}(n^2)$ rounds by an effective algorithm.*

References

[1] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous.* Kluwer, 2003.

[2] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Rendezvous and election of mobile agents: Impact of sense of direction. *Theory of Computing Systems* 40(2): 143–162, 2007.

[3] V. Baston and S. Gal. Rendezvous search when marks are left at the starting points. *Naval Research Logistics* 38: 469–494, 1991.

[4] M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *2nd International Conference on Networked Systems* (NETSYS), 109-124, 2015.

[5] M. Bournat, A. Datta, and S. Dubois. Self-stabilizing robots in highly dynamic environments. *18th International Symposium on Stabilization, Safety, and Security of Distributed System* (SSS), 54-69, 2016.

[6] S. Bouchard, Y. Dieudonne, and B. Ducourthial. Byzantine gathering in networks. *Distributed Computing* 29(6): 435–457, 2016.

[7] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. *IEEE Transactions on Computers* 63 (2): 397–410, 2014.

[8] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *Int. Journal of Parallel, Emergent and Distributed Systems* 27 (5): 387–408, 2012.

[9] J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. *21st International Symposium on Distributed Computing* (DISC), 108–122, 2007.

[10] J. Chalopin, Y. Dieudonne, A. Labourel, and A. Pelc. Fault-tolerant rendezvous in networks. *41st International Colloquium on Automata, Languages, and Programming* (ICALP), 411–422, 2014.

[11] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing* 41(4): 829–879, 2012.

[12] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing* 34: 1516–1528, 2005.

[13] J. Czyzowicz, A. Labourel, and A. Pelc. How to meet asynchronously (almost) everywhere. *ACM Transactions on Algorithms* 8(4): 37:1–37:14, 2012.

[14] J. Czyzowicz, S. Dobrev, E. Kranakis, and D. Krizanc. The power of tokens: Rendezvous and symmetry detection for two mobile agents in a ring. *34th Conference on Current Trends in Theory and Practice of Computer Science* (SOFSEM), 234–246, 2008.

[15] S. Das, F.L. Luccio, R. Focardi, E. Markou, D. Moro, and M. Squarcina. Gathering of robots in a ring with mobile faults. *17th Italian Conference on Theoretical Computer Science* (ICTCS), 122–135, 2016.

[16] S. Das, F.L. Luccio, and E. Markou. Mobile agents rendezvous in spite of a malicious agent. *11th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics* (ALGOSENSORS), 211–224, 2015.

[17] B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. *23rd ACM Symposium on Parallelism in Algorithms and Architectures* (SPAA), 139–148, 2011.

[18] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theoretical Computer Science* 355: 315–326, 2006.

[19] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. *European Symposium on Algorithms* (ESA), 184–195, 2003.

[20] Y. Dieudonné and A. Pelc. Anonymous meeting in networks. *Algorithmica* 74(2), 908–946, 2016.

[21] Y. Dieudonné, A. Pelc, and V. Villain. How to meet asynchronously at polynomial cost. *SIAM Journal on Computing* 44(3):844–867, 2015.

[22] G.A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Live exploration of dynamic rings. *36th IEEE International Conference on Distributed Computing Systems*, (ICDCS), 570-579, 2016.

[23] G.A. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, and G. Viglietta. Gathering in dynamic rings. *Arxiv* ??? , 2017.

[24] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Multiple agents rendezvous in a ring in spite of a black hole. *7th International Conference on Principles of Distributed Systems* (OPODIS), 34–46, 2003.

[25] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Multiple mobile agent rendezvous in the ring. *6th Latin American Conference on Theoretical Informatics* (LATIN), 599–608, 2004.

[26] P. Flocchini, B. Mans, N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science* 469: 53–68, 2013.

[27] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science* 337(1-3):147–168, 2005.

[28] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous with constant memory. *Theoretical Computer Science* 621:57–72, 2016.

[29] B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. *26th International Symposium on Distributed Computing* (DISC), 166–180, 2012.

[30] D. Ilcinkas, R. Klasing, and A.M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *Proc. 21st Int. Coll. Structural Inform. and Comm. Complexity (SIROCCO)*, 250–262, 2014.

[31] D. Ilcinkas and A.M. 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)*, 13-23, 2013.

[32] R Klasing, E Markou, and A Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science* 390 (1): 27–39, 2008.

[33] D.R. Kowalski and A. Malinowski. How to meet in anonymous network. *Theoretical Computer Science* 399(12): 141–156, 2008.

[34] D.R. Kowalski and A. Pelc. Polynomial deterministic rendezvous in arbitrary graphs. *15th Annual Symposium on Algorithms and Computation* (ISAAC), 644–656, 2004.

[35] E. Kranakis, D. Krizanc, and E. Markou. Mobile agent rendezvous in a synchronous torus. *7th Latin American Conference on Theoretical Informatics* (LATIN), 653–664, 2006.

[36] E. Kranakis, D. Krizanc, and E. Markou. *The Mobile Agent Rendezvous Problem in the Ring.* Morgan & Claypool, 2010.

[37] E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Mobile agent rendezvous problem in the ring. *23rd International Conference on Distributed Computing Systems* (ICDCS), 592-599, 2003.

[38] F. Kuhn, N. Lynch, R. Oshman. Distributed computation in dynamic networks. *42th Symposium on Theory of Computing* (STOC), 513–522, 2010.

[39] F. Kuhn, Y. Moses, R. Oshman. Coordinated consensus in dynamic networks. *30th Symposium on Principles of Distributed Computing* (PODC), 1–10, 2011.

[40] F. Kuhn and R. Oshman. Dynamic networks: Models and algorithms. *SIGACT News* 42(1): 82–96, 2011.

[41] J. Lin, A.S. Morse, and B.D.O. Anderson. The multi-agent rendezvous problem. Parts 1 and 2. *SIAM Journal on Control and Optimization*, 46(6): 2096–2147, 2007.

[42] A. Miller and A. Pelc. Time versus cost tradeoffs for deterministic rendezvous in networks. *Distributed Computing* 29(1): 5164, 2016.

[43] L. Pagli, G. Prencipe, and G. Viglietta. Getting close without touching: Near-gathering for autonomous mobile robots. *Distributed Computing* 28(5):333–349, 2015.

[44] A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks* 59(3): 331-347, 2012.

[45] C. Sawchuk. *Mobile Agent Rendezvous in the Ring.* Ph.D Thesis, Carleton University, January 2004.

[46] A. Ta-Shma and U. Zwick. Deterministic rendezvous, treasure hunts, and strongly universal exploration sequences. *ACM Transactions on Algorithms* 10(3), 2014.

[47] X. Yu and M. Yung. Agent rendezvous: A dynamic symmetry-breaking problem. *International Colloquium on Automata, Languages, and Programming* (ICALP), 610-621, 1996.