Tight Bounds for Black Hole Search in Dynamic Rings

Giuseppe Antonio Di Luna[†], Paola Flocchini[*], Giuseppe Prencipe[‡], Nicola Santoro[&]

†:Sapienza University, ∗: University of Ottawa, ‡: University of Pisa, &: University of Carleton.

Abstract

In this paper, we start the investigation of distributed computing by mobile agents in dangerous dynamic networks. The danger is posed by the presence in the network of a *black hole* (Bh), a harmful site that destroys all incoming agents without leaving any trace. The problem of determining the location of the black hole in a network, known as *black hole search* (Bhs), has been extensively studied in the literature, but always and only assuming that the network is static. At the same time, the existing results on mobile agents computing in dynamic networks never consider the presence of harmful sites. In this paper we start filling this research gap by studying black hole search in temporal rings, specifically focusing on *1-interval connectivity* adversarial dynamics.

The main complexity parameter of Bhs is the number of agents (called *size*) needed to solve the problem; other parameters are the number of moves (called *cost*) performed by the agents, and the *time* until termination.

Feasibility and complexity depend on many factors; the size n of the ring, whether or not n is known, and the type of inter-agent communication (whiteboards, tokens, face-to-face, visual). In this paper, we provide a *complete* feasibility characterization presenting size optimal algorithms. Furthermore, we establish lower bounds on the cost and time of size-optimal solutions and show that our algorithms achieve those bounds.

keywords: Mobile agents, black hole search, dynamic ring.

1 Introduction

1.1 Background

When computing in networked environments, *mobile agents* are used both as a theoretical computational paradigm and as a system-supported programming platform. These distributed mobile computing environments are subject to several security threats, including those posed by *host attacks*; that is, the presence in a site of processes that harm incoming agents. A particularly dangerous host, called *black hole* (Bh), is a network site hosting a stationary process that disposes of visiting agents upon their arrival, leaving no observable trace of such a destruction. Notice that the existence of a black hole is not uncommon in networked systems supporting code mobility; for example, both the presence of a virus that trashes any incoming message (e.g., by classifying it as spam) and the undetectable crash failure of a site render that site a black hole. Clearly, in presence of such a harmful host, the first step must be to determine its location. Black Hole Search (Bhs) is the distributed problem of determining the location of the black hole by a team of system agents. The problem, also called dangerous graph exploration, is solved if, within finite time, at least one agent survives knowing the location of the black hole [22].

The task to identify the Bh is clearly dangerous for the searching agents and might be impossible to perform. The research concern has been to determine under what conditions a team of mobile agents can successfully accomplish this task. Bhs has been extensively investigated in a variety of settings, depending on the types of communication mechanisms employed by the agents, their level of synchronicity, the topology of the network, etc., and under a variety of assumptions on the agents' knowledge and capabilities (e.g., [10–12, 21, 22, 27, 37, 38]; for a recent survey, see [44]). All these investigations share a common trait: the dangerous network on which the agents operate is *static* i.e., the link structure does not change in time.

Recently, research within distributed computing has started to focus on mobile-agent computing in *time-varying graphs* (a.k.a., highly dynamic graphs), i.e., graphs where the topological changes are not limited to sporadic and disruptive events (such as process failures, links congestion, etc), but are rather inherent in the nature of the network (e.g., see [9]). A large body of literature exists on time-varying graphs as they model a wide range of modern networked systems whose dynamic nature is the natural product of innovations in communication technology (e.g., wireless networks), in software layer (e.g., a controller in a software defined network), and in society (e.g., the pervasive nature of smart mobile devices). The vast majority of the research on dynamic networks has considered time-varying graphs with *discrete* temporal dynamics; that is, the network is seen as an infinite sequence of static graphs with the same vertex set, and it is usually called *evolving* or *temporal* graph. The study of mobile agents in temporal graphs includes both centralized and distributed investigations (e.g., [1,25,34,35,45]). Notice that, in such temporal graphs, the distributed computation is by definition *synchronous*; extensive investigations have been carried out under specific assumptions on the discrete temporal dynamics, including the minimal assumption of *temporal connectivity* (e.g., [5, 8, 32]), the popular assumption of *1-interval connectivity* (and its generalization of *T-interval connectivity*) (e.g., [2,3,17,19,33,39,40,46]), and *periodicity* (e.g., [1, 8, 26, 30, 36]). While the study of mobile agents on static networks is really mature, and generated a copious literature (e.g., see [31] and chapters therein), the research on mobile agents in temporal graphs is still in its infancy, especially from a purely distributed perspective. Its focus has been on classical problems, such as graph exploration [6,18,32], gathering [7,19], scattering [3], and grouping [14] under a variety of settings, depending on the types of communication mechanisms employed by the agents, the topology of the network, etc., and under a variety of assumptions on the agents' knowledge and capabilities (for a recent detailed survey see [16]). In spite of the different settings, these investigations share a common trait: they assume that the dynamic network on which the agents operate is *safe*; i.e., there is no Bh.

Summarizing, practically nothing is known on distributed computing by mobile agents in *dangerous dynamic networks*. In this paper, we start filling this research gap.

1.2 Problem

In this paper, we study Bhs in a *temporal ring* under the *1-interval connectivity* adversarial dynamics by a team of *colocated* agents. In other words, the network is a synchronous ring where one of the nodes is a Bh and, at any time unit, one edge (chosen by an adversary) is possibly missing. The problem to be solved is to identify the location of the Bh. The problem is solved by a team of mobile agents, executing the same protocol and initially deployed at a safe node of the network, if within finite time at least one agent survives and unambiguosly knows the location of the Bh.

The main research questions is to determine the minimum number of agents needed to solve Bhs; this parameter is called *team size* or simply *size*. Another important complexity measures is the number of moves, called *cost*, performed by the agents; in synchronous systems, such as temporal rings, an additional complexity measure is the amount of *time* until termination occurs.

Feasibility and complexity depend on many parameters; on the size n of the ring, on whether or not n is known, and whether the agents have distinct ids or are anonymous. A factor that is particularly important is the mechanism provided to the agents to communicate and interact. In the literature on distributed computing by mobile agents different models of interaction and communication with different capabilities have been considered. Listed in decreasing computational power, these models are: *Whiteboard*, whereas each node provides all visiting nodes with a shared memory, called whiteboard, that can be accessed, in fair mutual exclusion, to exchange information; *Pebble* (or *Token*), whereas each agent has available a pebble that can be carried and, when at a node, can be placed there or taken from there, the last two operations performed in fair mutual exclusion; *FaceToFace* (F2F), where the agents can exchange information only when they are in the same node at the same time; and *Vision*, where an agent can only sense the other agents in the same node at the same time but cannot explicitly communicate with them. These models can be conveniently grouped into two classes: *endogenous* (or internal), where the agents rely only on their own internal capabilities to communicate and can do so only when present on the same node (F2F and Vision models); and *exogenous* (or external), where the agents make use of external tools (pebbles and whiteboards) that allow them to leave traces or messages in the nodes of the network. Not surprisingly, neither the solutions for exploration of safe synchronous rings nor the ones for Bhs devised for static synchronous rings can be applied for the exploration of a dangerous temporal ring. Furthermore, the existing trivial lower-bound for Bhs in static synchronous rings, that more than one agent is needed, does not provide any insight into the complexity of the problem under

	Exogenous		*Endogenous*	
	Anonymous	IDs	Anonymous	IDs
Colocated	$\Theta(n^{1.5})$		impossible	$\Theta(n^2)$

Table 1: Map of the results.

investigation nor of the computational impact of the different parameters.

In other words, prior to this work, the feasibility and complexity of exploring a dangerous 1-interval connected ring is an unexplored problem.

1.3 Contributions

In this paper, we investigate the black hole search problem in an oriented 1-interval connected rings of size n by a team of k colocated agents. We consider the communication mechanism that they employ, whether or not n is known, and whether or not the agents are anonymous. For each possible setting, we provide a complete feasibility characterization. Furthermore, whenever the problem is solvable, we establish tight bounds on cost and time of a size-optimal solution.

We start by showing that knowledge of n is necessary for teams of any size k and irrespectively of the other parameters; and, that $k < 3$ agents cannot solve the problem even in the strongest possible model (colocated agents, and nodes equipped with whiteboards). We show that any optimal-size algorithm that uses endogenous communication requires $\Omega(n^2)$ cost and time even in the strongest of the endogenous mechanisms, FaceToFace; and we constructively prove that this bound is tight by designing a solution for the weakest of the endogenous models, Vision, that has a move and time complexity of $\Theta(n^2)$. With the more powerful exogenous mechanisms, we show a tight bound of $\Theta(n^{1.5})$: the lower bound holding for the strongest Whiteboard model, the matching upper bound for the weakest one, Pebble.

A summary of the results is shown in Table 1.

1.4 Related Work

The existing literature related to our research can be divided between that considering Bhs in static networks and that investigating distributed computing by mobile agents in safe dynamic graphs.

Agents in dangerous static graphs. The black hole search problem has been introduced by Dobrev *et al.* in their seminal paper [21]. A panoply of papers followed [4, 15, 28, 42, 47] solving the problem in different classes of graphs (trees [13], rings and tori [10, 23, 43], and in graphs of arbitrary and possibly unknow topology [12, 20, 21]), under several assumptions (see the recent survey [44]).

The most relevant papers for our work are the ones investigating the Bhs in static ring networks. In the *asynchronous* setting, optimal size and cost bounds have been established, solving the problem with two colocated agents and $\Theta(n \log n)$ moves, in the whiteboard model [22], and subsequently in the pebble model [27]. In [11], it has been shown that two scattered agents with pebbles are sufficient to find a black hole on oriented rings with $\mathcal{O}(n \log n)$ moves [24] and with $\mathcal{O}(n^2)$ in unoriented rings in [23]. In the *synchronous* setting, on the other hand, it is well known that two colocated non-anonymous agents with FaceToFace communication can solve the problem in arbitrary known graphs (and therefore on the ring) [12]. Finally, Bhs by scattered agents with constant memory has been studied in [11]: in unoriented rings 3 agents are necessary and sufficient when equipped with movable tokens, while more agents are needed when tokens are not movable. All papers on black hole search assume a static topology; the only exception is the study of *carrier graphs*, a particular class of periodic graphs defined by circular intersecting routes of public carriers, where the stops are the nodes of the graph and the agents can board and disembark from a carrier at any stop [29].

Agents in safe dynamic graphs. The study of mobile agents in temporal graphs is rather recent and includes both centralized and distributed investigations (e.g., [1, 25, 34, 35, 45]). From a distributed perspective, the research has so-far considered three types of temporal dynamics: *periodic* [30], *temporal connectivity* [5, 6, 32], and *1-interval connectivity* [3, 14, 18, 19, 32]; for a recent detailed survey see [16]. Several of these works considered ring networks.

Specifically, in *1-interval connected rings*: the *gathering* problem has been investigated in [19]; The *exploration* problem by a set of anonymous agents has been studied in [18] under several assumptions (handedness agreement, synchrony vs semi-synchrony, knowledge of n vs landmark, etc...); a recent preprint [41] has closed some questions left-open by [18] regarding the terminating exploration by a team of 3 agents. Always in 1-interval connected ring, recent papers investigated the problems of *grouping* [14] and *scattering* [3]. Exploration of a *temporally connected ring* was examined in [5, 6].

2 Model and Preliminaries

2.1 The Model and the Problem

The system is modeled as a time-varying graph $\mathcal{G} = (V, E, \mathbb{T}, \rho)$, where V is a set of nodes, E is a set of edges, $\mathbb{T}$ is the temporal domain, and $\rho : E \times \mathbb{T} \to \{0, 1\}$, called *presence function*, indicates whether a given edge is available at a given time [9].

The graph $G = (V, E)$ is called *underlying* graph (or *footprint*) of $\mathcal{G}$. In this paper we consider *discrete* time; that is, $\mathbb{T} = \mathbb{Z}^+$. Since time is discrete, the dynamics of the system can be viewed as a sequence of static graphs: $\mathcal{G} = G_0, G_1, \ldots, G_r, \ldots$, where $G_r = (V_r, E_r)$ is the graph of the edges present at round r (also called *snapshot* at time r). The time-varying graph in this case is called *temporal graph* (or *evolving graph*). We use the term *adversary* to refer to the scheduler that decides the sequence of dynamic graphs.

A temporal graphs where connectivity is guaranteed at every round is called *1-interval connected*; that is, a temporal graph $\mathcal{G}$ is 1-interval connected (or always connected) if $\forall G_i \in \mathcal{G}$, G_i is connected. In this paper we focus on *dynamic rings*, defined as 1-interval connected temporal graphs whose footprint is a ring. Let $\mathcal{R} = (v_0, v_1, \ldots v_{n-1})$ be a dynamic oriented ring, i.e., where each node v_i has two ports, consistently labelled left and right connecting it to v_{i-1} and v_{i+1}, respectively (all operations on the indices are modulo n). In plain speaking, an interval connected ring is a ring where at each round at most one edge is missing, please note that there is no fairness on edge removal and the same edge may be removed forever.

A set $A = \{a_0, a_1, \ldots, a_{k-1}\}$ of mobile agents operate in $\mathcal{R}$. Agents are initially in the same node (called *home-base*), we say that they are *colocated*. When the agents are identical (i.e., do not have distinct identifiers), we say that they are *anonymous*. In case agents are not anonymous we assume that their identities are visibile (e.g., if several agents are on the same node they can see who is who). The agents can move from node to neighbouring node, have bounded storage ($\mathcal{O}(\log n)$ bits of internal memory suffice for our algorithms), have computing capabilities and obey the same set of rules (i.e., execute the same algorithm). The agents operate in synchronous rounds, and they are all activated in each round. Upon activation, an agent on node v at round r takes a local snapshot of v that contains the set $E_r(v)$ of edges incident on v at this round, and the set of agents present in v. The agent also interacts with the other agents either explicitly or implicitly (the method of interaction depends on the communication mechanism employed and will be discussed later). On the basis of the snapshot, the local interaction, and the content of its local memory, an agent then decides what action to take. The action consists of a *communication step* (defined below) and a *move step*. In the move step the agent may decide to stay still or to move on an edge $e = (v, v') \in E_r(v)$. In the latter case, the agent will reach v' in round $r + 1$.

The interaction among the agents is regulated by different *communication mechanisms* depending on the model. We consider two classes of communication mechanisms (*endogenous* and *exogenous*) which give rise to four models.

Endogenous Mechanisms rely only on the robots' capabilities without requiring any external object. Among those we distinguish:
- *Vision:* the agents have no explicit means of communication; they can only see each other when they reside on the same node.
- *FaceToFace (F2F):* the agents can explicitly communicate among themselves only when they reside on the same node.

Exogenous Mechanisms do require external objects for the robots to exchange information. Among those we distinguish:
- *Pebble:* each agent is endowed with a single pebble that can be placed on or taken from a node. On each node, the concurrent actions of placing or taking pebbles are done in fair mutual exclusion.

- *Whiteboard:* each node contains a local shared memory, called whiteboard, of size $O(\log n)$ where agents can write on and read from. Access to the whiteboard is done in adversarial but fair mutual exclusion.

Notice that the mutual exclusion nature of the Pebble and Whiteboard models allows anonymous colocated agents to break the symmetry and assume different Ids.

The temporal graph $\mathcal{G}$ contains a *black hole* (BH), a node that destroys any incoming agent without leaving any detectable trace of that destruction. The goal of a *black hole search* algorithm $\mathcal{A}$ is to identify the location of the black hole, that is:

Definition 1. (BHS) *Given a dynamic ring $\mathcal{R}$, and an algorithm $\mathcal{A}$ for a set of agents we say that $\mathcal{A}$ solves the* BHS *if at least one agent survives and terminates. Each agent that terminates hast to know the footprint of $\mathcal{R}$ with the indication of the location of the backhole.*

The main measure of complexity is the number of agents, called *size*, used by the protocol. The other important cost measures are the total number of moves performed by the agents, which we shall call *cost*, and *time* it takes to complete the task.

In Figure 1 are shown (a) four rounds of an execution in a dangerous dynamic ring, and (b) the space diagram representation that we will use in this paper. The agent is represented as the black quadrilateral and it is moving clockwise; the BH is the black node. At round $r = 2$ and $r = 3$ the agent is blocked by the missing edge. In the diagram, the movement of the agent is represented as a solid line.

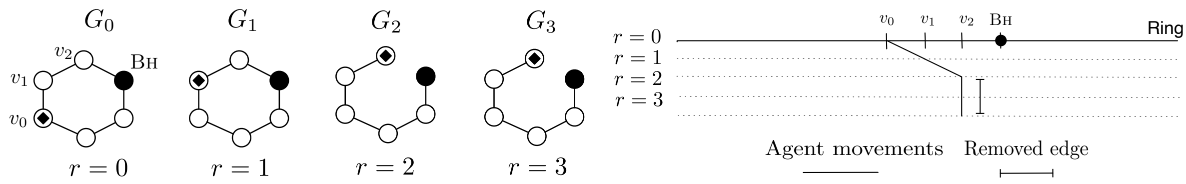


Figure 1: (a) Execution in a dangerous dynamic ring, and (b) its space diagram representation.

3 Impossibilities and Basic Limitations

In this section we show some general impossibility results that hold in the whiteboard model and hence under all communication mechanisms considered in this paper. More precisely, we establish that three agents are necessary to locate the BH and that, irrespectively of the number of agents available, the size of the ring must be known.

Lemma 1. *Let $\mathcal{R}$ be a dynamic ring of size $n > 3$. Let the agents know that the black hole is located in one of three consecutive nodes $H = \{v_1, v_2, v_3\}$ (different from the home-base). It is impossible for two colocated agents to locate the black hole and terminate. The impossibility holds even if the nodes are equipped with whiteboards, the agents have distinct IDs, and the ring is oriented.*

Proof. Let a and b be the two agents. By contradiction, let $\mathcal{A}$ be an algorithm that correctly locates the black hole regardless of the pattern of edge disappearance in the ring. Note that the two agents cannot visit for the first time a node in H travelling on the same edge at the same round, otherwise the adversary would place the black hole in that node killing both. At least one of the agents must move to visit H. Let us assume, w.l.o.g., that a is the first to reach H (to visit v_1) at some round r or that both agents reach H at round r (a visiting v_1 and b visiting v_3 from the other side). At this point the adversary, regardless of the position of b, removes edge $e = (v_0, v_1)$. Note that, while the edge is missing, the two agents cannot communicate because they are disconnected on one side by the missing link and on the other by the black hole. Should agent a survive, it has no choice but visiting v_2 (from v_1) to determine whether the black hole is in v_2 or in v_3. Agent b cannot wait for ever in v_0 because e might be permanently missing, and it has to reach H from the other side. Once b reaches v_3, if it does not die it cannot avoid visiting v_2 to determine whether the black hole is in v_1 or in v_2. Hence, within finite time they would both enter v_2, albeit at different moments. By choosing the black hole to be in v_2, both agents die, contradicting the correctness of the algorithm.

□

From the above technical lemma is immediate that:

Theorem 2. *In a dynamic ring of size $n > 3$, two colocated agents cannot solve the* Bhs. *The impossibility holds even if the agents have unique IDs, and are equipped with the strongest (*Whiteboard*) communication model.*

Interestingly, we can show that there is no algorithm solving Bhs if n is unknown. Such result does not depend on the number of agents.

Theorem 3. *There exists no algorithm that solves the* Bhs *in a dynamic ring $\mathcal{R}$ whose size is unknown to the agents. The result holds even if the nodes have whiteboards, the agents have IDs, and irrespectively of the number of agents.*

Proof. The proof is by contradiction. Let $\mathcal{A}$ be a correct algorithm, and let the adversary remove an edge e at round 0. For the algorithm to be correct, there must exist a round r when the the portion of the ring delimited by the black hole and by edge e is fully explored (with at least one agent dead in the black hole and the corresponding link marked as dangerous). Not knowing the size of the ring, as long as e is missing, the remaining agents cannot decide whether they have explored all the nodes of the ring and can terminate, or whether the ring is larger, the missing edge e is not incident to the black hole, and there is still a portion to explore. If they decide to terminate, the adversary will make e re-appear revealing the unexplored part of a larger ring and $\mathcal{A}$ would be incorrect; if instead they decide to wait for e to re-appear, the missing edge will be permanently missing, and the agents will never terminate. □

Next recall the following obvious fact:

Observation 1. *Anonymous colocated agents with an endogenous communication mechanism (i.e., F2F or Vision model) cannot solve* Bhs *in a static ring, regardless of their number.*

We now introduce a technical lemma that will be used to establish lower bounds in the rest of the paper. The lemma is based on the following observation

Observation 2 ([25]). *Given a dynamic ring $\mathcal{R}$, and a cut U (with $|U| > 1$) of its footprint connected by edges e_c and e_{cc} to nodes in $V \setminus U$. Assume that at round r all the agents are in U. If at round r there are not two agents, one that tries to traverse e_c and one that tires to traverse e_{cc}, then the adversary may prevent agents to visit a node outside U.*

We say that a node v is explored if it is has been visited at least one time by an agent. Let us assume that agents are colocated and let us use U_r to denote the set of explored nodes at round r. Note that U_r must be a cut of the ring, and that $U_{r-1} \subseteq U_r$. We will also say that a round r is an *expansion round* if $U_r \subset U_{r+1}$, and that agent a communicates with another agent b after round r, if either a and b meet at a round $r' > r$, or at a round $r' > r$ agent b visits a node on which a wrote something in a round r'' with $r < r'' \leq r'$.

Lemma 4. *If $\mathcal{A}$ solves the* Bhs *with $\mathcal{O}(n \cdot f(n))$ moves using three agents, then there must exist an agent a that explores a sequence seq of at least $\Omega(\frac{n}{f(n)})$ nodes such that:*
- a does not communicate with any other agent while exploring nodes in seq.
- a visits at most $o(n)$ nodes outside seq while exploring nodes in seq.
The lemma holds even if the agents are colocated, they have distinct IDs, and the nodes are equipped with whiteboards.

Proof. Let us have three agents starting from node v_0. It is easy to see that, if $\mathcal{A}$ is correct, then there is a round r in which $|U_r| = \Theta(n)$ and $|V \setminus U_r| = \Theta(n)$: as long as the black hole Bh $\notin U_r$ then $\mathcal{A}$ cannot correctly terminate, and Bh can be at a clockwise and counter-clockwise distance that is $\Theta(n)$ from v_0.

Let round $r' \geq r$ be an *expansion round* in which a new node v_{x_1} is explored. By Observation 2 we have that at each expansion round there are two agents trying to cross the edges of the cut, we name them the pushing agents. Note that the adversary may always allow only one the pushing agent to explore a new node, and block the other.

Without loss of generality, we use the term exploring agent, shortened in a, to refer to the pushing agent that explores a new node, and as b and c the others. Note that it may be possible that the three agents alternate their roles of pushing agents and so that exploring agent is not always the same agent to explore. However in this case we just rename the agents, as we are using the name a just fo convenience since the proof does not depend on a being always the same agent.

By Observation 2 and from the fact that the adversary can decide which agent will explore a node outside $U_{r'}$ by orchestrating the removal of the edges, we can assume that in each expansion round the distance between a and the other agents is $\Theta(n)$.

Therefore, in order for a to communicate with b and c after each expansion round $r' > r$, at least $\Theta(n)$ moves are needed.

Now let $v_{x_1}, v_{x_2}, \ldots, v_{x_t}$ be a sequence of t consecutive nodes that a explores from round r' to round r'', such that (i) a never communicates with other agents from round r' to round r'', and (ii) a visits at most $o(n)$ nodes after each exploration of a node in the sequence. Let us call such a sequence a *solitary sequence* of length t. Note that when a explores a single node and immediately communicates with others, we have a solitary sequence of length 1. Therefore, after round r, each time we explore a new node, we create a solitary sequence of length at least 1.

We now argue that after round r at most $s = \mathcal{O}(f(n))$ solitary sequences are generated. Suppose the contrary, by definition when a solitary sequence ends $\Omega(n)$ moves are executed, and, by hypothesis $\mathcal{A}$ runs in $\mathcal{O}(n \cdot f(n))$ moves. If we had $\omega(f(n))$ solitary sequences we would have $\omega(n \cdot f(n))$ moves (recall that $f(n) = \omega(g(n))$ if $\lim_{n \to +\infty} \frac{f(n)}{g(n)} = +\infty$). However, there are still $k = \Theta(n)$ nodes to explore after round r, and this has to be done with $s = \mathcal{O}(f(n))$ solitary sequences. Therefore, there must exists at least one solitary sequence of length at least $\frac{k}{s} = \Omega(\frac{n}{f(n)})$. Such a solitary sequence proves our claim. $\square$

Intuitively, Lemma 4 says that, in any Bhs algorithm that has cost (i.e., number of moves) $\mathcal{O}(n \cdot f(n))$, there exists at least one agent that explores a sequence of nodes of length $\Omega(\frac{n}{f(n)})$; during the exploration of such sequence the agent does not communicate with others (either by a direct meeting, or by writing on a whiteboard or leaving a pebble on node that is visited by others), and it visits at most $o(n)$ nodes outside the ones in the sequence.

4 Preliminaries

Before presenting and analyzing our solution protocols, we briefly describe a well known idea employed for Bhs in static graphs that will be adapted in our algorithms, as well as the conventions and symbols used in our protocols.

4.1 Cautious Walk

Cautious Walk is a mechanism introduced in [21] for agents to move on dangerous graphs in such a way that two (or more) agents never enter the black hole from the same edge. The general idea of cautious walk in static graphs is that when an agent a moves from u to v through an unexplored (thus dangerous) edge (u, v), a must leave the information that the edge is under exploration at u. The information can be provided through some form of mark in case of exogenous communication mechanisms, or implicit in case of endogenous mechanisms (e.g., by employing a second agent as a "witness"). In our algorithms we will make use of variants of the general idea of cautious walk, adapting it to the dynamic case and to the particular model under discussion.

4.2 Pseudocode Convention

We use the pseudocode convention introduced in [18]. In particular, our algorithms use as a building block procedure Explore ($dir \mid p_1 : s_1; p_2 : s_2; \ldots; p_k : s_k$), where $dir \in \{left, right, nil\}$, p_i is a predicate, and s_i is a state. In Procedure Explore, the agent takes a snapshot, then evaluates predicates $p_1, \ldots, p_k$ in order; as soon as a predicate is satisfied, say p_i, the procedure exits, and the agent transitions to the state s_i specified by p_i. If no predicate is satisfied, the agent tries to move in the specified direction dir (or it stays still if $dir = nil$), and the procedure is executed again in the next round. The following are the main predicates used in our Algorithms:

- *meeting*[ID]: the agent sees another agent with identifier ID arriving at the node where it resides, or the agent arrives in a node, and it sees another agent with identifier ID.

- *sees*[ID]: the agent sees another agent with identified ID in the node where it resides.

Furthermore, the following variables are maintained by the algorithms:

- $Ttime$, $Tnodes$: the total number of rounds and distinct visited nodes, respectively, since the beginning of the execution of the algorithm.
- $Etime$, $Enodes$: the total number of rounds and distinct visited nodes, respectively, since the last call of procedure Explore.
- $EMtime$ [C/ (CC)]: the number of rounds during which the clockwise/ (resp. counter-clockwise) edge is missing since the last call of procedure Explore.
- $\#Meets[ID]$: the number of times the agent has met with agent ID.
- $RLastMet[ID]$ records the number of rounds elapsed since the agent has seen (or meet) an agent with id ID

Observe that, in a fully synchronous system, when predicate *meeting*[y] holds for an agent a with id x, then predicate *meeting*[x] holds for the agent with id y. In the algorithm agents are divided in three roles: Avanguard, Leader and Retroguard. Based on the total order of IDs agents assign to themselves the appropriate role depending on if they are lowest, middle, or highest.

5 Endogenous Communication Mechanisms

First recall that, with endogenous communication mechanisms, IDs are necessary for Bhs (see Observation 1). Hence, we assume the agents have unique IDs.

5.0.1 Lower bound on Cost and Time

In this section we present a quadratic lower bound on the number of moves and on the number of rounds needed by any algorithm of optimal size to solve Bhs.

Theorem 5. *Given a dynamic ring $\mathcal{R}$, any algorithm $\mathcal{A}$ that solves the* Bhs *with three agents and an endogenous communication mechanism has a cost of at least $\Omega(n^2)$ moves and needs $\Omega(n^2)$ rounds. The result holds even if the agents are colocated, have distinct IDs, and the model is F2F.*

Proof. Let a, b, c be the three agents. We first show the bound on the number of moves. The proof proceeds by contradiction: let $\mathcal{A}$ be a solution algorithm performing $o(n^2)$ moves. By Lemma 4, when n is large enough, in $\mathcal{A}$ there exists a round r such that, by the end of round r, agent a explored at least three nodes, say v_1, v_2 and v_3, without communicating with b and c. Let the black hole be one of these three nodes; hence, by round r, agent a is eliminated. At this point, the two remaining agents, b and c, even if aware of a's demise, are unaware of which of v_1, v_2 and v_3 is the Bh. By Lemma 1, the agents cannot determine the exact location and, hence, $\mathcal{A}$ cannot correctly solve the problem; a contradiction.

For the bound on the rounds, as just shown, $\mathcal{A}$ must perform $\Omega(n^2)$ moves. Even if these moves were equally divided among the three agents and performed in parallel, we would have at most three moves at each round. Therefore, the number of rounds is quadratic, and the claim follows. □

5.0.2 An Optimal Solution: CautiousPendulum

From Theorem 2 we know that the minimum size of any solution algorithm is three. From Theorems 5, we know that cost and time are $\Omega(n^2)$. In this Section, we show that the bounds are tight describing a size optimal solution with $O(n^2)$ moves and rounds. Our solution works in the weakest communication model (Vision).

Our CautiousPendulum algorithm works using three agents: the Avanguard, the Retroguard, and the Leader (refer to Figure 2, which shows two examples of possible runs of the algorithm). Initially, all three agents are on the same node v_0, the home-base.

Avanguard and Leader move clockwise "cautiously": If the edge e in the clockwise direction is not present, both Leader and Avanguard wait until it reappears.

Algorithm 1 Algorithm CautiousPendulum for AVANGUARD

States: {Init, NewNode, Return, Move}.
In state Init, NewNode:
  EXPLORE($right$ | $Enodes > 0$: Return)
In state Return:
  EXPLORE($left$ | $Enodes > 0$: Move)
In state Move:
  EXPLORE($right$ | $Enodes > 0$: NewNode)

Algorithm 2 Algorithm CautiousPendulum for RETROGUARD

States: {Init, Bounce, Return}.
In state Init:
  $nextTarget \leftarrow 1$
  EXPLORE($left$ | $Enodes \geq nextTarget$: Return)
In state Return:
  EXPLORE($right$ | $sees$[LEADER]: Bounce)
In state Bounce:
  $nextTarget \leftarrow Enodes + 1$
  EXPLORE($left$ | $Enodes \geq nextTarget$: Return)

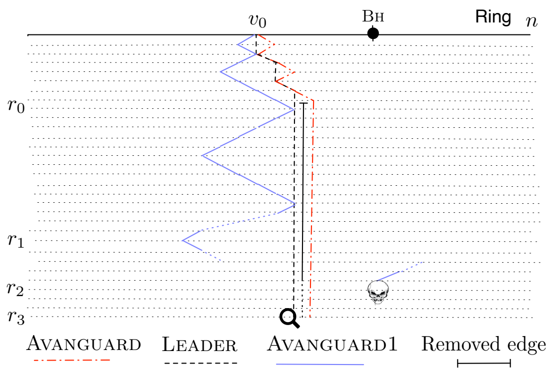
If edge e is present, AVANGUARD moves to the unexplored node using edge e. Then, if in a successive round the edge e is present, AVANGUARD goes back to LEADER, signalling that the recently visited node is safe; at this point, both LEADER and AVANGUARD safely move clockwise to the recently explored node If AVANGUARD does not return when e is present, LEADER knows the position of the black hole (the node just visited by AVANGUARD) and terminates; in this case, we say that AVANGUARD *fails to report.*

While AVANGUARD and LEADER are performing this special exploration, RETROGUARD moves as follows: it goes counter-clockwise until it visits the first unexplored node; then, it goes back clockwise until it meets again LEADER. Once RETROGUARD meets LEADER, it reverts back its movement direction to counter-clockwise, iterating the same kind of move: in other words, it swings similarly to a pendulum that increases its counter-clockwise amplitude of one node at each oscillation.

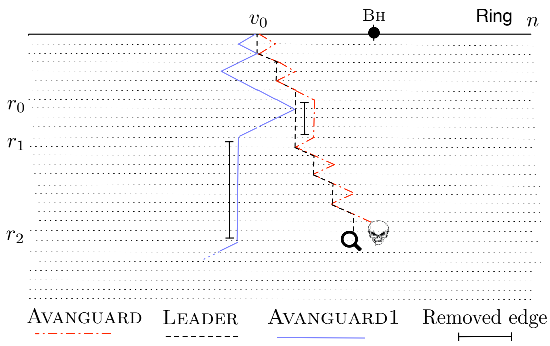
In case RETROGUARD finds a missing edge on its path, it waits until the edge re-appears; and then it keeps performing the oscillating movement. We say that RETROGUARD *fails to report* to LEADER if the LEADER sees a missing edge in its clockwise direction and, despite waiting for this edge to appear for a time long enough for RETROGUARD to explore a new node and go back, it does not meet RETROGUARD. Intuitively, since at most one edge is missing at each round, the fail to report of RETROGUARD implies that RETROGUARD entered the black hole. Note that, in this case, LEADER can compute exactly the position of the black hole.

Algorithm 3 Algorithm CautiousPendulum for LEADER

Predicates Shorthands: $FailedReport[Avanguard] = Etime > EMtime[C]$.
$FailedReport[Retroguard] = EMtime[C] > 2((\#Meets[Retroguard] + 1) + Tnodes)$.
States: {Init, Cautious, Move, TerminateA, TerminateR}.
In state Init,Cautious:
  EXPLORE(nil | $meeting[Avanguard]$: Move; $FailedReport[Avanguard]$: TerminateA;
$FailedReport[Retroguard]$: TerminateR)
In state Move:
  EXPLORE($right$ | $Enodes > 0$: Cautios; $FailedReport[Retroguard]$: TerminateR)
In state TerminateA:
  Terminate, BH is in the next node in clockwise direction.
In state TerminateR:
  Terminate, BH is in the node at counter-clockwise distance $\#Meets[Retroguard] + 1$ from v_0.



(a) Example of a run where the termination is due to Retroguard. At round r_0 Avanguard and Leader are blocked by a missing edge. At round r_2 Retroguard enters in the black hole. At round r_3 Retroguard fails to report.



(b) Example of a run where the termination is due to Avanguard. At round r_1 Avanguard and Leader are free to go while Retroguard is blocked (at most one edge can be removed at each round). At round $r_2 - 1$ Avanguard enters in the black hole. At round r_2 Avanguard fails to report.

Figure 2: Example runs of algorithm CautiousPendulum. The black hole is the node Bh and it is indicated by a black dot. When no edge is missing agents move as follows. The Retroguard moves in the counter-clockwise direction, while Avanguard and Leader move in clockwise direction. The Avanguard and the Leader move in a coordinated way: the Avanguard goes forward, then back and at this point both do a step forward (see the zig-zag line of Avanguard). Agent Retroguard moves as a pendulum exploring a new node left at each swing. In Figure 2a is depicted a termination due to the failure to report by Retroguard. In Figure 2b the termination is due to the failure to report by Avanguard.

Correctness In this section we prove the correctness of algorithm CautiousPendulum described in the previous section (refer to the pseudocode of Algorithms 1, 2, and 3).

Lemma 6. *Consider three agents executing* CautiousPendulum*. Let r be the first round in which one agent enters the black hole. We have $r = \mathcal{O}(n^2)$.*

Proof. Recall that Retroguard moves counter-clockwise of $nextTarget$ steps, and then it moves clockwise until it meets Leader: the distance it travels during this movement is upper bounded by $2n$. This implies that if Retroguard is not blocked by a missing edge, it explores a new node every $2n$ rounds at most. Since at most one edge is missing at each round, and since by construction Avanguard moves clockwise and Retroguard moves counter-clockwise exploring disjoint portion of the rings, they can never be both blocked at the same round. Moreover, if one of them is blocked for $2n$ rounds (even not consecutively), the other explores at least a node. It follows that at least one new node is explored by one of these two agents every (at most) $\mathcal{O}(n)$ rounds. The above implies that, in at most $\mathcal{O}(n^2)$ rounds, Retroguard or Avanguard reaches the black hole. □

Observation 3. *Consider three agents executing* CautiousPendulum*. The leader never enters in the* Bh*.*

Proof. The Leader moves into a node only after Avanguard explored it and went back to signal the node as safe: therefore, Leader can never enter the black hole. □

Lemma 7. *Consider three agents executing* CautiousPendulum*. Let r be the round in which one agent enters* Bh*. Then* Leader *terminates by a round $r_f = r + \mathcal{O}(n^2)$.*

Proof. By Observation 3, at round r Avanguard or Retroguard entered the black hole. First observe that for the Leader we always have $\#Meets[Retroguard] \leq n$ and $Tnodes \leq n$. The bound $Tnodes \leq n$ follows from the fact that the Leader cannot do more than n steps otherwise it would enter the black hole. The bound $\#Meets[Retroguard] \leq n$ derives from the fact that Retroguard explores a new node (in counter-clockwise direction) for each meeting with Leader; thus, after at most n meetings, Retroguard entered the black hole. We distinguish the two possible cases:
1) Avanguard reaches the black hole at round r. By construction, at round r the Leader is in state Cautious in the counter-clockwise neighbor of the black hole. If edge e between the Leader and the

black hole is present at a round $r' > r$, then LEADER will not see AVANGUARD returning at round $r'+1$; hence, it terminates by predicate $FailedReport[Avanguard]$. Therefore, let us assume that edge e is missing from round r on. In this case, RETROGUARD cannot be blocked and will explore a new node every at most $2n$ rounds (recall that RETROGUARD oscillates). Now, the only scenario that can occur is that also RETROGUARD enters the black hole (from the counter-clockwise neighbor of the black hole): again, no more than one edge is missing and it has to be e, therefore, RETROGUARD is never blocked. Consequently, in $\mathcal{O}(n^2)$ rounds RETROGUARD enters in the black hole. Recall, that at this moment on the LEADER we have $\#Meets[Retroguard] \leq n$ and $Tnodes \leq n$. Therefore, after $\mathcal{O}(n)$ additional rounds RETROGUARD will fail to report to LEADER, and the predicate $FailedReport[Retroguard] = EMtime[C] > 2n+2 > 2*((\#Meets[Retroguard]+1)+Tnodes)$ will be verified.
2) RETROGUARD reaches the black hole at round r. Let us suppose the LEADER is at a node v whose clockwise edge is missing; if this edge is missing for more than $2 \cdot n$ rounds, then the failure to report of RETROGUARD triggers, and the LEADER terminates. Therefore, let us assume that no edge in the clockwise direction of movement of LEADER is ever missing for more than $2 \cdot n$ rounds. In this case, LEADER and AVANGUARD will eventually reach the black hole from the clockwise direction, in at most in $\mathcal{O}(n^2)$ rounds. Once AVANGUARD enters the black hole, LEADER terminates in at most $2 \cdot n$ rounds, either by the failure to report check by RETROGUARD (see predicate $FailedReport[Retroguard]$) or by AVANGUARD not returning to LEADER (predicate $FailedReport[Avanguard]$). □

Lemma 8. *Consider three agents executing* CautiousPendulum. *If the* LEADER *terminates then it correctly locates the* BH.

Proof. We first discuss the relationship between variables $\#Meets[Retroguard]$ and $Tnodes$ on LEADER and the behaviour of RETROGUARD. RETROGUARD moves counter-clockwise of $nextTarget$ steps, and then it moves clockwise until it meets LEADER. When RETROGUARD meets with the leader its variable $Enodes$ contains the number of edges traversed from the last explored node, in counter-clockwise direction, and the LEADER, at this point RETROGUARD updates its $nextTarget$ as $nextTarget = Enodes+1$, and the LEADER updates $\#Meets[Retroguard] = \#Meets[Retroguard]+1$. It is thus clear that the next node RETROGUARD will explore is at distance $\#Meets[Retroguard]+1$ from v_0 in the clockwise orientation ($nextTarget$ starts from 1). It is also clear that, considering the variables $\#Meets[Retroguard]$ and $Tnodes$ on LEADER, the quantity $2*((\#Meets[Retroguard]+1)+Tnodes)$ is an upper bound on the number of edges that RETROGUARD traverses to explore a new node and go back to LEADER.
We can now prove our claim by a cases analysis on the possible termination cases:
- LEADER terminates in state TerminateA on node v at round r: in this case,
$FailedReport[Avanguard]= Etime > EMtime[C]$ is verified when LEADER is in state Cautious. That is, there has been at least one round in which the clockwise edge incident to v was present. Let the round r'. By construction, in a round $r'' < r'$ AVANGUARD moved in the clockwise direction (recall that LEADER is in state Cautious); but at round r' the edge is not missing and agent AVANGUARD did not return. It follows that AVANGUARD moved into the black hole, hence LEADER can correctly compute its position.
- LEADER terminates in state TerminateR on node v at round r: in this case,
$FailedReport[Retroguard]= vMtime[C] > 2*((\#Meets[Retroguard]+1)+Tnodes)$ is verified. That is, the clockwise edge incident to v has been missing for at least $2*((\#Meets[Retroguard]+1)+Tnodes)$ rounds. Since at most one edge can be missing at each round, this interval of time is sufficient for RETROGUARD to reach an unexplored node at distance $\#Meets[Retroguard]+1$ from v_0 in counter-clockwise orientation, and then go back to LEADER (recall that LEADER is at distance $Tnodes$ from v_0 in the clockwise orientation). This implies that RETROGUARD entered the black hole, at distance $\#Meets[Retroguard]+1$ from v_0 in the counter-clockwise direction. Hence, also in this case, LEADER can correctly compute the position of the black hole.

□

Theorem 9. *Consider a dynamic ring $\mathcal{R}$, with three colocated agents with distinct IDs in the Vision model. Algorithm* CautiousPendulum *solves* BHS *with $\mathcal{O}(n^2)$ moves and in $\mathcal{O}(n^2)$ rounds.*

Proof. By Lemma 6 and Observation 3 we have that at a round $r = \mathcal{O}(n^2)$ RETROGUARD or AVANGUARD entered the black hole while the LEADER remains alive. By Lemma 7 we have that at a round $r_f = r + \mathcal{O}(n^2)$ the LEADER terminates, and by Lemma 8 it correctly locates the BH. The fact that no other agent terminates incorrectly it is immediate from the observation that Retroguard and Avanguard have no terminating state. □

Summarizing, by Th. 5 and Th. 9 we have:

Theorem 10. *Algorithm* CautiousPendulum *is size-optimal with optimal cost and time.*

5.1 Exogenous Communication Mechanisms

Note that, with the exogenous communication mechanisms, anonymous agents on the same node can easily break the symmetry and assume different Identifiers by exploiting the mutual exclusion nature of pebbles and whiteboards.

As for the ability of agents to interact, we can observe that, even in the simpler pebble model, any communication between agents located at the same node is easy to achieve (e.g., two agents may exchange messages of any size using a communication protocol in which they send one bit every constant number of rounds). Therefore, in this section we can assume that the agents are able to communicate. Specifically, the communication of constant size messages is assumed to be instantaneous, since it can implemented trivially by a multiplexing mechanism (the logical rounds are divided in a constant number of physical round, the first of which is used to execute the actual algorithm and the others to communicate).

5.1.1 Lower bound on Cost and Time

The lower bound of Theorem 5 does not hold when employing one of the proposed exogenous communication mechanisms. We now show lower bounds of $\Omega(n^{1.5})$ on cost and time complexity; the lower bounds hold even employing the strongest of the exogenous mechanisms: whiteboards.

Theorem 11. *Given a dynamic ring $\mathcal{R}$ in the Whiteboard model, any algorithm $\mathcal{A}$ solving* Bhs *with three agents requires $\Omega(n^{1.5})$ moves and $\Omega(n^{1.5})$ rounds even if the agents are colocated and have distinct IDs.*

Proof. Let a, b and c be the tree agents. By Lemma 4, if $\mathcal{A}$ terminates in $o(n \cdot \sqrt{n})$ moves, then there is an agent, say a, that explores a sequence $S = v_1, v_2, \ldots, v_k$ of $\Omega(\sqrt{n})$ consecutive nodes while it does not communicate with any other agent. Let us suppose that the black hole is in S. Recall that a communicated with the other agents only before starting the exploration of S. Hence, when a reaches Bh, neither b nor c know the exact location of the node a was visiting when it entered the black hole. In the best case, when a disappears, agents b and c know that Bh is in sector S, but they do not know the exact location of Bh in S (refer to Figure 3).

Since, by hypothesis, nodes are equipped with whiteboards, it is possible that a wrote on a set of nodes Q the exact location of Bh. However, by Lemma 4, when a reaches the black hole Bh, neither b nor c visited one of the nodes in Q. Without loss of generality, let us assume that Q is a set of contiguous nodes placed at the counter-clockwise side of the Bh (see Figure 3). By Lemma 4, $|Q| = o(n)$.

Also, let C be the cut of the ring that includes both b and c (i.e., $C \cap Q = \emptyset$). By Observation 2, if b and c want to visit a node outside C, then one of them has to try to traverse the clockwise edge connecting C with other nodes in V, while the other agent has to do the same in the counter-clockwise direction. Algorithm $\mathcal{A}$ may identify the black hole only in two possible ways: either one agent visits a node in Q; or an agent, say b, reaches Bh while a knows the next node b is just about to visit.

Let e_q be the edge connecting a node in Q to a node in $V \setminus S$, and let e_{s_0} be the edge connecting a node in S to a node in $V \setminus Q$ (see Figure 3). We now establish the following strategy for an adversarial scheduler that decides which edge is missing: the adversary removes the edges such that no agent will ever traverse e_q (this can be achieved by always removing e_q when an agent is present on a node with e_q incident). Moreover, the adversary never let agents continue their exploration on S as long as one of them is not trying to traverse e_q. As an example as long as v_1 is never explored the adversary always remove e_{s_0} but when an agent may traverse e_q, when v_1 is explored the same behaviour will be applied to $e_{s_1} = (v_1, v_2)$, and so on. Thanks to this strategy, no agent will ever learn the location of Bh by visiting a node in Q.

Therefore, an agent is forced to reach the black hole by using a counter-clockwise edge (e_{s_2} in Figure 3). Let us assume that b is the agent that traverses for the first time an edge $e_{s_j} = (v_j, v_{j+1})$ with $v_j \in S$.

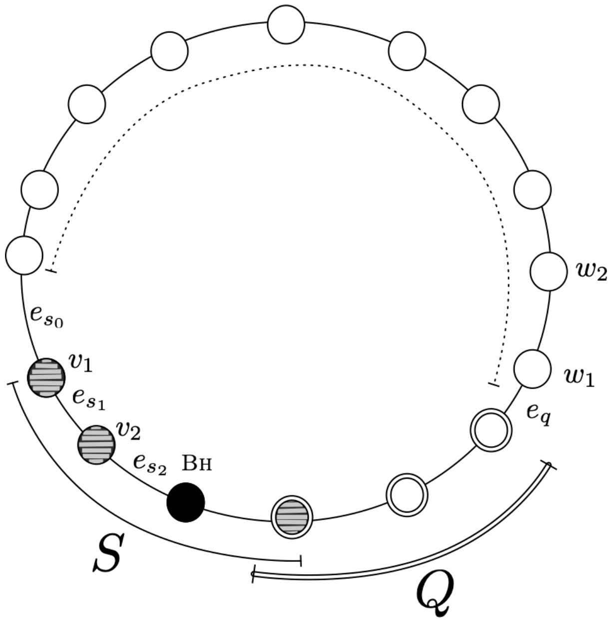


Figure 3: Pictorial representation for the lower bound of $n\sqrt{n}$: the zone S is a set of contiguous nodes where the black hole could be located, the zone Q represents all nodes where a wrote information about the location of s.

Note that it might be possible that is not always b to do that, but each time agent cross each other we may simply assume they swap also identity.

The strategy of the scheduler forces a to try to traverse e_q each time b traverses a new e_{s_j}. Also, b needs to communicate with a after every newly explored node in S: otherwise, if b explores two nodes without communicating with a and one is the black hole, then a cannot disambiguate which one is BH (the adversary can perpetually block a on two neighbor nodes by a careful removal of edges).

Finally, $m = \Theta(n)$ moves are necessary for b to be able to communicate with a: in fact, a is trying to traverse e_q when b traverses an e_{s_j}. Therefore, $m|S|$ moves are required to identify a black hole in S. The bound now follows immediately from the fact that $|S| = \Omega(\sqrt{n})$. The bound on the number of rounds follow immediately from the fact that a constant number of agents needs $\Omega(\sqrt{n})$ rounds to perform $\Omega(\sqrt{n})$ moves.

□

5.1.2 An Optimal Solution: CautiousDoubleOscillation

We now describe the CautiousDoubleOscillation, that is an optimal solution in the weakest of the two exogenous communication mechanisms (i.e., the Pebble model).

Primitives and Pseudocode Conventions in the Pebble model In our algorithm we use the term *mark* to and *unmark* to indicate that an agent is leaving/removing the pebble from a node. Besides the Explore procedure described in Section 4, we also use procedure CautiousExplore. CautiousExplore makes an agent perform a cautious walk using the pebbles: the agent marks a node, moves in the *dir* direction, then it goes back, unmarks the node, and move again in the *dir* direction. Moreover, to verify predicate *meeting*$[x]$ in this context, the concerned agents should not be trying to unmark a node (otherwise they are not considered to have met).

Note that this can be implemented by sending messages as discussed at the beginning of Section 5.1. Finally, we define predicate *marked* to be verified when the agent resides in a marked node.

High level description The algorithm structure is reminiscent of a pendulum that oscillates with two different amplitudes. At the start Retroguard oscillates by increasing its counter-clockwise amplitude of $\sqrt{n}$ nodes at times: in other words, Retroguard explores at each oscillation a sector of $\sqrt{n}$

contiguous nodes before reporting to LEADER. While exploring a sector, RETROGUARD uses the pebble to perform a cautious walk. As long as RETROGUARD is alive, the LEADER and the AVANGUARD act as in CautiousPendulum.

The LEADER detects when, and if, RETROGUARD reached the black hole by using a timeout strategy. Such a timeout strategy is designed in order to not trigger in case RETROGUARD is just blocked by an edge (see the detailed description).

In case RETROGUARD entered the black hole, the LEADER knows the sector that RETROGUARD was exploring (LEADER keeps track of the sector by counting the number of times it met RETROGUARD). Since this sector contains the black hole, it will be denoted as *dangerous sector*.

When (and if) RETROGUARD fails to report, the LEADER starts moving counter-clockwise looking for the last node marked by RETROGUARD, while AVANGUARD starts exploring the dangerous sector in the clockwise direction.

The exploration of the sector by AVANGUARD is done in a cautious way, by reporting back to LEADER for each newly explored node in this sector (at each swing, the amplitude of AVANGUARD increases of only 1 node); thanks to this strategy, LEADER knows which node AVANGUARD is exploring. Note that, if AVANGUARD is not blocked by a missing edge it enters in the black hole in $\mathcal{O}(n\sqrt{n})$ rounds.

At the end, either LEADER reaches the pebble left by RETROGUARD (and thus, it knows where the black hole is), or AVANGUARD will enter the black hole and it fails to report to LEADER (also in this case LEADER can correctly compute the position of the black hole).

Detailed description The pseudocode for CautiousDoubleOscillation is in Algorithms 4, 5, and 6. Also, in Figure 4, two executions are reported.

As long as RETROGUARD is not detected dead, the behavior of LEADER and AVANGUARD is the same as in CautiousPendulum. RETROGUARD explores counter-clockwise, in a cautious way, sectors of size $\sqrt{n}$ (states Init and Bounce of Algorithm 4).

Once RETROGUARD explored a sector, it reaches LEADER to report, by moving clockwise (state Return of Algorithm 4). Once the report is over, RETROGUARD moves counter-clockwise until it reaches the end of the next unexplored sector (see the update of $Esteps$ in state Bounce; see also the example reported in Figure 4a).

If RETROGUARD fails to report back to LEADER, then LEADER goes into the Detection state by using a timeout strategy: the transition occurs when the number of rounds from the last meet with RETROGUARD is $7((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$.

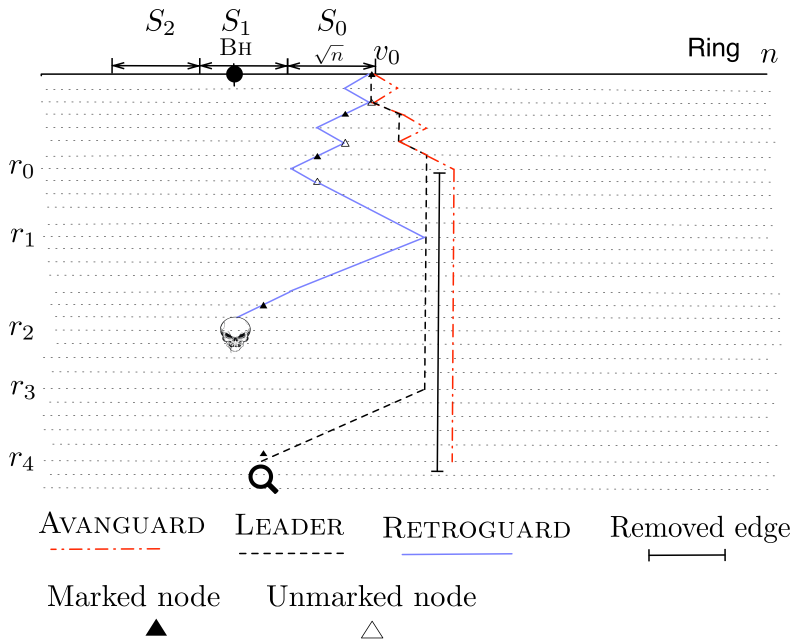
Note that the quantity $(\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes$ is an upper bound on the maximum distance from LEADER and RETROGUARD: the first component $(\#Meets[Retroguard] + 1)\sqrt{n}$ is the counter-clockwise distance between the last node of the sector under exploration and v_0, the second quantity $Tnodes$ is the clockwise distance between the actual position of LEADER and v_0 (in the proof will become apparent why we need to multiply this quantity by the constant 7). We remark that the factor $Tnodes$ avoid that the LEADER timeous if RETROGUARD is blocked on an edge, as long as RETROGUARD is blocked the LEADER moves increasing $Tnodes$ and delaying the timeout.

Once in state Detection, the LEADER walks counter-clockwise trying to reach the last node marked by RETROGUARD (see state Detection of Algorithm 6). In state Detection the LEADER also resets $\#Meets[Avanguard]$ to 0, since it is interested in counting the number of times it meets AVANGUARD from the state switch. A LEADER in Detection state terminates if either it finds the marked node, or $3n$ rounds passed without meeting AVANGUARD (refer again to the example of Figure 4a).

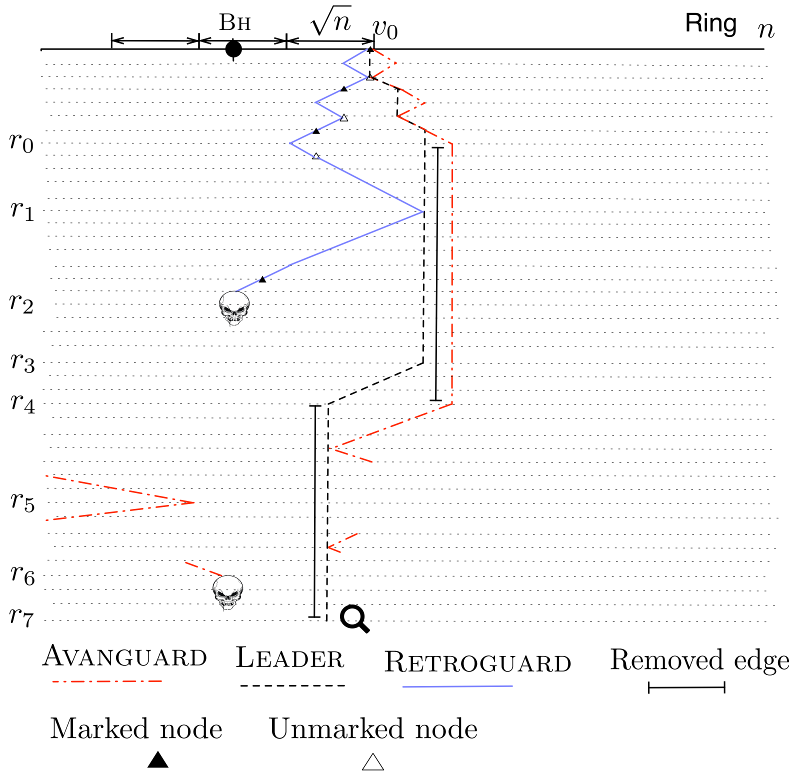
The AVANGUARD detects that RETROGUARD disappeared by recognizing that the LEADER moved in a way that is not compatible with the simulated cautious walk. More specifically, AVANGUARD goes into the SearchLeader state if in state NewNode or state Move it does not see the LEADER.

While in this state, AVANGUARD moves counter-clockwise until it meets LEADER; when (and if) this occurs, they both start a communication protocol in which AVANGUARD reads the variable $\#Meets[Retroguard]$ from LEADER's memory[1], and it starts oscillating between state Detection2 and Return1. The behavior of AVANGUARD in these two states is as follows: AVANGUARD goes clockwise until it reaches the first unexplored node in the dangerous sector (the sector that RETROGUARD was exploring when failed to

[1]Recall that at the beginning of Section 5.1 we discussed how pebbles can be used to communicate messages of non-constant size.



(a) Example of a run where the termination is due to LEADER finding the marked node. At round r_0 AVANGUARD and LEADER are blocked by a missing edge. In the meanwhile agent RETROGUARD is exploring a sector of size $\sqrt{n}$: note the marking and unmarking of nodes. At round r_0 the RETROGUARD completes the exploration of the first sector S_0, and goes back to the LEADER. After meeting with LEADER, it starts exploring sector S_1 (we are not showing the cautious walk until the beginning of S_1). At round r_2 RETROGUARD enters in the black hole leaving the counter-clockwise neigbhour marked. At round r_3 the LEADER detects the failure to report by RETROGUARD and it changes direction of movement. At round r_4 the LEADER finds the marked node and terminates.



(b) Example of a run where the termination is due to AVANGUARD. Until round r_3 the execution is as the one in Figure 4a. At round r_4 the LEADER is blocked. At the same round AVANGUARD goes back and it does not find the LEADER, thus it understands that the leader is looking for the node marked by RETROGUARD. At this point AVANGUARD goes counter-clockwise until it meets the LEADER, and it learns the sector that RETROGUARD was exploring. Once this is done, AVANGUARD starts exploring the dangerous sector in the clockwise direction: at round r_5 it explores the first node of such sector and goes back to LEADER. At round r_6 it explores the second node entering in BH. At round r_7 the LEADER detects the failure to report by AVANGUARD and terminates.

Figure 4: Example executions of Algorithm CautiousDoubleOscillation.

report); the dangerous sector's position can be computed using $\#Meets[Retroguard]$: in fact, its position is $\#Meets[Retroguard]\sqrt{n}$ nodes away, according to the counter-clockwise direction from v_0. When a new node in the dangerous sector has been explored, AVANGUARD moves counter-clockwise until it meets LEADER. Thanks to this mechanism, LEADER always knows the precise location of the next node AVANGUARD is just about to explore (refer to the example in Figure 4b).

Algorithm 4 Algorithm CautiousDoubleOscillation for RETROGUARD

States: {Init, Bounce, Return}.
In state Init:
  CAUTIOUSEXPLORE($left$ | $Enodes \geq \sqrt{n}$: Return)
In state Return:
  EXPLORE($right$ | $sees$[LEADER]: Bounce)
In state Bounce:
  $steps \leftarrow Enodes + \sqrt{n}$
  CAUTIOUSEXPLORE($left$ | $Enodes \geq steps$: Return)

Algorithm 5 Algorithm CautiousDoubleOscillation for AVANGUARD

States: {Init, NewNode, Return, Move, SearchLeader, Detection1, Return1, Detection2.}
In state Init, NewNode:
  EXPLORE($right$ | $\neg sees[Leader]$:SearchLeader; $Enodes > 0$: Return)
In state Return:
  EXPLORE($left$ | $Enodes > 0$: Move)
In state Move:
  EXPLORE($right$ | $\neg sees[Leader]$:SearchLeader; $Enodes > 0$: NewNode)
In state SearchLeader:
  EXPLORE($left$ | $meeting[Leader]$: Detection1)
In state Detection1:
  Communicate with the LEADER to compute $nextTarget$ that is the distance to first unexplored node in the dangerous sector from clockwise direction.
  EXPLORE($right$ | $Enodes \geq nextTarget$: Return1)
In state Return1:
  EXPLORE($left$ | $meeting$[LEADER]: Detection2)
In state Detection2:
  $nextTarget \leftarrow Enodes + 1$
  EXPLORE($right$ | $Enodes \geq nextTarget$: Return1)

Correctness.

Definition 2. *A* sector *is a sequence of* $\sqrt{n}$ *nodes that* RETROGUARD *explores in state* Bounce*. The sequence of sectors that* RETROGUARD *explores is denoted by* $S_0, S_1, \ldots$*, where* S_i*,* $i \geq 0$*, is the i-th sector explored by* RETROGUARD*.*

Lemma 12. *If* LEADER *enters in state* Detection *at round* r*, then* RETROGUARD *reached the black hole in a round* $r_x < r$*.*

Proof. The proof is by contradiction. Suppose the LEADER enters in state Detection at round r, while RETROGUARD is still alive. It follows that at round r the predicate $FailedReport[Retroguard]$ is verified by LEADER. Let r_{last} be the last round, before r, in which RETROGUARD and LEADER met; thus, $RLastMet[Retroguard] = r - r_{last} > 7((\#Meets[Retroguard]+1)\sqrt{n} + Tnodes)$. Where $Tnodes$ is the value of the corresponding variable stored by LEADER at round r.

Let T be the interval between r_{last} and r, that is $T = [r_{last}, r-1]$. We now prove the fact (1): in interval T the leader has done less than $\frac{|T|}{7}$ steps clockwise. Suppose the contrary, for each step clockwise

Algorithm 6 Algorithm CautiousDoubleOscillation for LEADER

Variables: $RLastMet[X]$ = number of round since the last meeting of LEADER and agent X.
Predicates Shorthands:
$FailedReport[Avanguard] = Etime > EMtime[C]$.
$FailedReport[Retroguard] = RLastMet[Retroguard] > 7((\#Meets[Retroguard]+1)\sqrt{n}+Tnodes)$.
$FailedReportD = RLastMet[Avanguard] > 3n$.

States: {Init, Cautious, Move, Detection, TerminateA, TerminateR, TerminateAD }
In state Init, Cautious:
    EXPLORE(*nil* | *meeting*[*Avanguard*]: Move; *FailedReport*[*Avanguard*]: TerminateA; *FailedReport*[*Retroguard*]: Detection)
In state Move:
    EXPLORE(*right* | *Enodes* > 0: Cautios; *FailedReport*[*Retroguard*]: Detection)
In state Detection:
    $\#Meets[Avanguard] \leftarrow 0$
    $D \leftarrow$ number of nodes explored in the dangerous sector by *Avanguard*
    EXPLORE(*left* | *marked*: TerminateR; *FailedReportD*: TerminateAD)
In state TerminateA:
    Terminate, BH is in the next node in clockwise direction.
In state TerminateR:
    Terminate, BH is in the next node in counter-clockwise direction.
In state TerminateAD:
    Terminate, BH is the last node AVANGUARD was exploring. This is computed using $\#Meets[Avanguard]$ and $\#Meets[Retroguard]$. Specifically, BH is the node that is $(\#Meets[Avanguard] + D)$ nodes way from the clockwise end of the dangeours sector, where D is the number of nodes AVANGUARD explored in the dangerous sector during the simulation of the cautious walk.

of the leader the quantity $7((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$ increases of 7 units, this implies that after $\frac{|T|}{7}$ steps the quantity would be greater or equal than $|T|$, that is in contradiction with the triggering of $FailedReport[Retroguard] = |T| > 7((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$.

Now we will show that fact (1) is in contradiction with the hypothesis that RETROGUARD is alive at round r. By construction, at round r_{last}, RETROGUARD starts moving counter-clockwise until it explores all nodes in sector $S_{\#Meets[Retroguard]}$. If RETROGUARD is not blocked by a missing edge, it will complete this exploration in at most $3(\#Meets[Retroguard] + 1)\sqrt{n} + 3Tnodes$ rounds; then, RETROGUARD will move clockwise toward LEADER, reaching it in at most $(\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$ rounds. The overall sum of the above rounds is $4((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$, that is clearly less than $|T|$.

Therefore, the only possible scenario left to analyze is when RETROGUARD has been blocked during its movement (by hypothesis, RETROGUARD cannot reach the black hole before round r). The number of rounds in which RETROGUARD has been blocked during interval T is at least $3(\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$: if RETROGUARD moves for $4((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$ rounds, it meets the leader in interval T preventing $FailedReport[Retroguard]$ to trigger.

However, for every three rounds in which RETROGUARD is blocked the LEADER moves one step. Therefore, in T the leader did (at least) $(\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$ steps clockwise, this quantity is precisely $\frac{|T|}{7}$ and it is in contradiction with fact (1).

From the above we have that RETROGUARD cannot be alive at round r, and this prove our claim. □

Lemma 13. *The* LEADER *does not enter the black hole.*

Proof. By Lemma 12, if LEADER goes into the Detection state, then RETROGUARD reached BH; also, RETROGUARD marked with a pebble the last safe visited node. Thus, LEADER will find the marked node and terminate before entering BH (predicate *marked* in state Detection). If LEADER does not go in state Detection, then it moves clockwise simulating a cautious walk with AVANGUARD. Since LEADER waits for AVANGUARD to return before visiting a new node, it follows that LEADER will never enter the black hole. ☐

Lemma 14. *Let r be the first round in which one among* AVANGUARD *or* RETROGUARD *enters the black hole. We have* $r = \mathcal{O}(n^{1.5})$.

Proof. First note that, as long as RETROGUARD does not reach the black hole, the set of nodes that are visited by RETROGUARD and AVANGUARD are disjoint (not considering the black hole BH): in fact, AVANGUARD switches direction of movement only when LEADER goes in state Detection; however, by Lemma 12, this transitions cannot occur as long as RETROGUARD does not reach the black hole. Therefore, since at most one edge at the time might be missing, it follows that at most one among RETROGUARD and AVANGUARD may be blocked at each round.

If RETROGUARD is not blocked, it first explores sector S_0 of $\sqrt{n}$ nodes, then it goes back to LEADER (that is at most n hops away), then it explores the new sector S_1 of $\sqrt{n}$, and so on. Therefore, its exploration costs a number of rounds that is upper bounded by:

$$\sum_{i=0}^{\sqrt{n}} 3(i\sqrt{n} + n).$$

By immediate algebraic manipulation, we have that $3\sum_{i=0}^{\sqrt{n}}(i\sqrt{n} + n) \leq 6n\sqrt{n}$.

Now we observer that if AVANGUARD is free to move for $3n$ rounds, then it explores n nodes. Thus, if AVANGUARD is not blocked for at least $3n$ rounds over an interval of $12n\sqrt{n}$ rounds, it will necessary reach the black hole BH. However, if AVANGUARD is blocked for $3n$ rounds, then RETROGUARD is free to move for $12n\sqrt{n} - 3n \geq 6n\sqrt{n}$ rounds, and it reaches BH. From the above in the first $12n\sqrt{n}$ RETROGUARD or AVANGUARD reaches the black hole, and thus the lemma follows. ☐

Lemma 15. *Let r be the first round when an agent enters the black hole; then,* LEADER *terminates by round* $r_t = r + \mathcal{O}(n^{1.5})$.

Proof. Let r be the first round when someone enters in the black hole, note that by Lemma 13 we can exclude that is the LEADER to enter in the black hole. Therefore, at round r AVANGUARD or RETROGUARD reached the black hole.

We will now show that LEADER terminates by round $r' = r + \mathcal{O}(n^{1.5})$. We distinguish two cases:

- AVANGUARD reaches the black hole BH at round r. By construction, at round r LEADER is at node v neighbour of BH; let e be the edge between v and BH. LEADER waits on v until either: (1) the edge e is not missing, or (2) RETROGUARD fails to report. In case (1), predicate $FailedReport[Avanguard]$ is verified by LEADER, hence LEADER terminates.

  If (1) does not apply, we first show then there is a round $r_f > r$ in which LEADER detects that RETROGUARD has failed to report, with $r_f \in \mathcal{O}(n \cdot \sqrt{n})$. Since e is missing, RETROGUARD cannot be blocked; hence, it reaches BH by round $r + 6 \cdot n \cdot \sqrt{n}$ (refer to the argument for the bound on r used in Lemma 14). Also, RETROGUARD marks the node before the black hole with a pebble.

  By predicate $FailedReport[Retroguard]$, LEADER detects this event (RETROGUARD in BH) by round $r_f = r + 6 \cdot n \cdot \sqrt{n} + 7((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$. We argue that $r_f = r + 6 \cdot n \cdot \sqrt{n} + 7((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes)$ is $\mathcal{O}(n \cdot \sqrt{n})$: we have that $Tnodes \leq n$, otherwise the LEADER would have entered the black hole that is impossible, see Lemma 13); and $\#Meets[Retroguard] \leq n$, each times RETROGUARD meets the leader it explores $\sqrt{n}$ nodes thus RETROGUARD would enter in the black hole after at most $\sqrt{n}$ meetings.

  At round r_f, LEADER changes state, it reverts direction of movement, and starts moving towards the node that RETROGUARD marked. If LEADER does not reach the marked node within $3n$ rounds, then AVANGUARD fails to report to LEADER (by predicate $FailedReportD$), and LEADER terminates. Otherwise, if LEADER reaches the marked node, it terminates as well. In both cases, the leader within $\mathcal{O}(n^{1.5})$ rounds, and the lemma follows.

- RETROGUARD reaches the black hole at round r, while exploring sector S_J. By construction, RETROGUARD marks the node v before BH with a pebble; also, $J = \#Meets[Retroguard]$.

Consider now round $r' = r + 7((\#Meets[Retroguard] + 1)\sqrt{n} + Tnodes$: if the LEADER does not terminate by round r', then it detects that RETROGUARD failed to report (by predicate $FailedReport[Retroguard]$).

Since $\#Meets[Retroguard] \leq n$ and $Tnodes \leq n$, it follows that $r' = \mathcal{O}(n \cdot \sqrt{n})$. At round r' LEADER goes into Detection state, and it switches direction of movement. Now, if AVANGUARD reaches BH by round r', then the proof is similar to the previous case. Otherwise, if LEADER is never blocked by a missing edge, it reaches v at most n rounds after switching direction, it hence terminates, and the lemma follows again.

Thus, let us consider the case when LEADER is blocked by a missing edge before n rounds after r'. We assume that AVANGUARD does not enter in the black hole by round r', otherwise the proof is equal to previous case.

While the LEADER is blocked, AVANGUARD has enough time to trigger the predicate $\neg sees[Leader]$, and to go SearchLeader state; in this state, AVANGUARD moves toward the LEADER. When AVANGUARD and LEADER meet, AVANGUARD uses the value of $\#Meets[Retroguard]$ to identity the node of sector $S_{\#Meets[Retroguard]}$ that has to be explored (this is done by updating variable $nextTarget$). At this point AVANGUARD, in state Detection1, moves until it explores, from clockwise direction, the first node in the dangerous sector. If such node is not the black hole AVANGUARD switches to state Return1, it goes back to LEADER, and upon meeting it goest to Detection2 updating its target node in the dangerous sector. This oscillation between states Return1 and Detection2 is iterated until AVANGUARD enters in the black hole.

We claim that, if AVANGUARD is not blocked, then AVANGUARD enters in the black hole in at most $2n\sqrt{n}$ rounds: in state Detection2 AVANGUARD explores a new node in $S_{\#Meets[Retroguard]}$, then it switches to state Return1 and meets the LEADER; this process lasts at most $2n$ rounds. Since $S_{\#Meets[Retroguard]}$ contains $\sqrt{n}$ rounds the claim follows. Once AVANGUARD entered in the black hole, the LEADER will terminate after additional $3n$ rounds (see predicate $FailedReportD$).

□

Lemma 16. *If the* LEADER *terminates, it can correctly locate the position of the black hole.*

Proof. The proof proceeds by case analysis on the terminating conditions of Algorithm 6.

- LEADER terminates in state TerminateA. This case occurs when $FailedReport[Avanguard]$ is triggered and LEADER is in state either Init or Cautious. In both states, LEADER is waiting for AVANGUARD to return; also, $FailedReport[Avanguard]$ triggers if the clockwise edge e is present for two rounds (not necessarily consecutive), and $meeting[Avanguard]$ has not been verified. The first round AVANGUARD moves to a neighbor node using e, and it goes to Return state. Thus, as soon as edge e appears again the second time, an alive AVANGUARD would get back to the node where the LEADER is, preventing the triggering of $FailedReport[Avanguard]$. Therefore, the only possibility left, is for AVANGUARD to be lost in the black hole. Hence, the LEADER correctly terminates and correctly detects the position of BH.

- LEADER terminates in state TerminateR. First, LEADER can go in the TerminateR state only from state Detection. By Lemma 12, LEADER goes in the Detection state only after RETROGUARD reaches the black hole. By construction, when RETROGUARD reaches the black hole, it leaves a pebble in the clockwise neighbour of BH (RETROGUARD performs a cautious exploration in both states Init and Bounce); also, RETROGUARD is the only agent that leaves a pebble. Therefore, when LEADER finds a marked node, it necessarily is the node marked by RETROGUARD. Hence, BH is the counter-clockwise neighbour of the marked node, and thus LEADER terminates correctly.

- LEADER terminates in state TerminateAD. First, this state is reachable only from the Detection state; also, by Lemma 12, LEADER goes in the Detection state only after RETROGUARD reaches the black hole. Let r be the round when the LEADER goes to the Detection state. Note that the LEADER resets variable $\#Meets[Avanguard]$ to 0.

  If LEADER reaches the node marked with a pebble by RETROGUARD, then it would terminate in state TerminateR. (see previous case). Therefore, let us assume that LEADER never reaches the node marked by RETROGUARD; that is, LEADER is blocked by a missing edge starting from round $r_b = r + \delta$ on, with $\delta < n$.

First, we will show that $FailedReportD$ cannot trigger as long as AVANGUARD is alive. If AVANGUARD is alive at round r_b (i.e., it did not reach BH), it follows that AVANGUARD cannot be blocked from round r_b on; hence, at most by round $r_b + 2$, it goes to the searchLeader state. Now, in at most n rounds, AVANGUARD meets LEADER, it computes the position of the dangerous sector, and it starts oscillating with a period of $2n$ rounds. Therefore, predicate $FailedReportD$ cannot trigger as long as AVANGUARD is alive, and when AVANGUARD enters in the black hole after round r_b the LEADER knows the position of BH (Recall that, LEADER is always aware of the node that AVANGUARD is exploring in the dangerous sector, by accessing the value of $\#Meets[Avanguard]$).

Let us now consider the case when AVANGUARD is not alive at round r_b: this case occurs only if AVANGUARD reaches BH, and BH is the clockwise neighbour of the node where LEADER is when it goes to the Detection state. By construction of the algorithm before round $r + 1$, AVANGUARD and LEADER do not occupy the same node only when AVANGUARD explores a new node (state Return). Hence, since AVANGUARD is not alive, predicate $FailedReportD$ triggers at round $r + 3n$, and $\#Meets[Avanguard] = 0$. Thus, LEADER correctly locate the position of the black hole: it is in the clockwise neighbour of the node visited by the LEADER at round r (that is the node where the LEADER switches to Detection state).

Therefore, in all cases, the lemma follows. ☐

Theorem 17. *Consider a dynamic ring $\mathcal{R}$ with three colocated agents in the Pebble model. Algorithm* CautiousDoubleOscillation *solves the* BHS *with $\mathcal{O}(n^{1.5})$ moves and $\mathcal{O}(n^{1.5})$ rounds.*

Proof. By Lemmas 14 and 13, it follows that in at most $\mathcal{O}(n^{1.5})$ rounds, either RETROGUARD, or AVANGUARD, or both, reach the black hole. At this time, by Lemma 15, LEADER terminates in $\mathcal{O}(n^{1.5})$ rounds; also, by Lemma 16, LEADER correctly identifies the position of the black hole, thus solving BHS. It is also clear that agents in state AVANGUARD and RETROGUARD cannot terminate by algorithm design, they have no terminating state in their algorithms. Thus, they cannot terminate incorrectly.

Finally the bound on the number of moves derives directly from the bound on the number of rounds needed to terminate, and from the fact that the number of agents is constant. ☐

We can conclude that:

Theorem 18. *Algorithm* CautiousDoubleOscillation *is size-optimal with optimal cost and time.*

References

[1] E. Aaron, D. Krizanc, and E. Meyerson. DMVP: Foremost waypoint coverage of time–varying graphs. In *40th International Workshop on Graph–Theoretic Concepts in Computer Science*, pages 29–41, 2014.

[2] S. Abshoff and F. Meyer auf der Heide. Continuous aggregation in dynamic ad-hoc networks. In *21st Int. Coll. on Structural Inf. and Comm. Compl.*, pages 194–209, 2014.

[3] A. Agarwalla, J. Augustine, W. Moses, S. Madhav, and A.K. Sridhar. Deterministic dispersion of mobile robots in dynamic rings. In *19th International Conference on Distributed Computing and Networking*, pages 19:1–19:4, 2018.

[4] B. Balamohan, S. Dobrev, P. Flocchini, and N. Santoro. Exploring an unknown dangerous graph with a constant number of tokens. *Theoretical Computer Science*, 2014.

[5] M. Bournat, A.K. Datta, and S. Dubois. Self–stabilizing robots in highly dynamic environments. In *18th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 54–69, 2016.

[6] M. Bournat, S. Dubois, and F. Petit. Computability of perpetual exploration in highly dynamic rings. In *37th IEEE International Conference on Distributed Computing Systems*, pages 794–804, 2017.

[7] M. Bournat, S. Dubois, and F. Petit. Gracefully degrading gathering in dynamic rings. In *20th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 349–364, 2018.

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

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

[10] J. Chalopin, S. Das, A. Labourel, and E. Markou. Black hole search with finite automata scattered in a synchronous torus. In *Proc of 25th International Symposium on Distributed Computing*, pages 432–446, 2011.

[11] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for black hole search with scattered agents in a synchronous ring. *Theoretical Computer Science*, 509:70–85, 2013.

[12] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71:229–242, 2006.

[13] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in synchronous tree networks. *Combinatorial Probabilistic Computing*, 16(4):595–619, 2007.

[14] S. Das, G.A. Di Luna, L. Pagli, and G. Prencipe. Compacting and grouping mobile agents on dynamic rings. In *15th Annual Conference on Theory and Applications of Models of Computation*, pages 114–133, 2019.

[15] M. D'Emidio, D. Frigioni, and A. Navarra. Exploring and making safe dangerous networks using mobile entities. In *Ad–hoc, Mobile, and Wireless Network*, pages 136–147. Springer, 2013.

[16] G.A. Di Luna. *Mobile Agents on Dynamic Graphs,* Chapter 20 of [31]. Springer, 2019.

[17] G.A. Di Luna and R. Baldoni. Brief announcement: Investigating the cost of anonymity on dynamic networks. In *34th Symposium on Principles of Distributed Computing*, pages 339–341, 2015.

[18] G.A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Distributed exploration of dynamic rings. *Distributed Computing*, 33:41–67, 2020.

[19] G.A. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, and G. Viglietta. Gathering in dynamic rings. *Theoretical Computer Science*, 811:79–98, 2020.

[20] S. Dobrev, P. Flocchini, R. Královič, and N. Santoro. Exploring an unknown dangerous graph using tokens. *Theoretical Computer Science*, 472:28–45, 2013.

[21] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocols. *Distributed Computing*, 19(1):1–35, 2006.

[22] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48(1):67–90, 2007.

[23] S. Dobrev, N. Santoro, and W. Shi. Locating a black hole in an un–oriented ring using tokens: The case of scattered agents. In *13th International Euro–Par Conference European Conference on Parallel and Distributed Computing*, pages 608–617. Springer, 2007.

[24] S. Dobrev, N. Santoro, and W. Shi. Scattered black hole search in an oriented ring using tokens. In *25th IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2007.

[25] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *42nd International Colloquium on Automata, Languages, and Programming*, pages 444–455, 2015.

[26] T. Erlebach and J.T. Spooner. A game of cops and robbers on graphs with periodic edge-connectivity. In *46th International Conference on Current Trends in Theory and Practice of Informatics*, pages 64–75, 2020.

[27] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: optimal black hole search with pebbles. *Algorithmica*, 62(3-4):1006–1033, 2012.

[28] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *28th IEEE International Parallel and Distributed Processing Symposium*, pages 1–10, 2009.

[29] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, 2012.

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

[31] P. Flocchini, G. Prencipe, and N. Santoro *(Eds.)*. *Distributed Computing by Mobile Entities.* Springer, 2019.

[32] T. Gotoh, P. Flocchini, T. Masuzawa, and N. Santoro. Tight bounds on distributed exploration of temporal graphs. In *23rd International Conference on Principles of Distributed Systems*, pages 22:1–22:16, 2019.

[33] B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In *26th Int. Symp. on Distributed Computing*, pages 166–180, 2012.

[34] D. Ilcinkas, R. Klasing, and A.M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *21st International Colloquium Structural Information and Communication Complexity*, pages 250–262, 2014.

[35] D. Ilcinkas and A.M. Wade. Exploration of the T–interval–connected dynamic graphs: the case of the ring. *Theory of Computing Systems*, 62(5):1144–1160, 2018.

[36] R. Jathar, V. Yadav, and A. Gupta. Using periodic contacts for efficient routing in delay tolerant networks. *Ad Hoc & Sensor Wireless Networks*, 22(1,2):283–308, 2014.

[37] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and approximation results for black hole search in arbitrary networks. *Theor. Comput. Sci.*, 384(2-3):201–221, 2007.

[38] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Approximation bounds for black hole search problems. *Networks*, 52(4):216–226, 2008.

[39] F. Kuhn, T. Locher, and R. Oshman. Gradient clock synchronization in dynamic networks. *Theory of Computing Systems*, 49(4):781–816, 2011.

[40] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *42nd ACM Symposium on Theory of Computing*, pages 513–522, 2010.

[41] S. Mandal, A. Rahaman Molla, and W.K. Moses Jr. Live exploration with mobile robots in a dynamic ring, revisited, 2020. `arXiv:2001.04525`.

[42] E. Markou. Identifying hostile nodes in networks using mobile agents. *Bulletin of the European Association for Theoretical Computer Science*, 108:93–129, 2012.

[43] E. Markou and M. Paquette. Black hole search and exploration in unoriented tori with synchronous scattered finite automata. In *14th International Conference on Principles of Distributed Systems*, pages 239–253, 2012.

[44] E. Markou and W. Shi. *Dangerous Graphs,* Chapter 18 of [31]. Springer, 2019.

[45] O. Michail and P.G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

[46] R. O'Dell and R.Wattenhofer. Information dissemination in highly dynamic graphs. In *Joint Workshop on Foundations of Mobile Computing*, pages 104–110, 2005.

[47] M. Peng, W. Shi, and J. Corriveau. Repairing faulty nodes and locating a dynamically spawned black hole search using tokens. In *6th IEEE Conference on Communications and Network Security*, pages 136–145. IEEE, 2018.