Black Hole Search in Dynamic Rings: The Scattered Case

Giuseppe A. Di Luna
DIAG, Sapienza University of Rome, Italy

Paola Flocchini
School of Electrical Engineering and Computer Science, University of Ottawa, Canada

Giuseppe Prencipe
Department of Computer Science, University of Pisa, Italy

Nicola Santoro
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

In this paper we investigate the problem of searching for a black hole in a dynamic graph by a set of scattered agents (i.e., the agents start from arbitrary locations of the graph). The black hole is a node that silently destroys any agent visiting it. This kind of malicious node nicely models network failures such as a crashed host or a virus that erases the visiting agents. The black hole search problem is solved when at least one agent survives, and it has the entire map of the graph with the location of the black hole. We consider the case in which the underlining graph is a dynamic 1-interval connected ring: a ring graph in which at each round at most one edge can be missing. We first show that the problem cannot be solved if the agents can only communicate by using a face-to-face mechanism: this holds for any set of agents of constant size, with respect to the size n of the ring.

To circumvent this impossibility we consider agents equipped with movable pebbles that can be left on nodes as a form of communication with other agents. When pebbles are available, three agents can localize the black hole in $O(n^2)$ moves. We show that such a number of agents is optimal. We also show that the complexity is tight, that is $\Omega(n^2)$ moves are required for any algorithm solving the problem with three agents, even with stronger communication mechanisms (e.g., a whiteboard on each node on which agents can write messages of unlimited size). To the best of our knowledge this is the first paper examining the problem of searching a black hole in a dynamic environment with scattered agents.

2012 ACM Subject Classification Computing methodologies → Distributed algorithms; Theory of computation → Self-organization

Keywords and phrases Black hole search, mobile agents, dynamic graph

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2023.33

Funding The work was partially supported by the Natural Sciences and Engineering Research Council of Canada under Discovery Grants and by "PRA – Progetti di Ricerca di Ateneo" (Institutional Research Grants) – Project no. PRA_2022_81 of the Pisa University and PRA RM1221816C1760BF from Sapienza and by (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union.

1 Introduction

1.1 Exploration of Dynamic Networks

In the distributed computing community a large set of works (see [20]) studies the computational paradigm of *mobile agents*. A *mobile agent* is a software component that is able to move on a network visiting nodes. When a node is visited, the agent executes some

27th International Conference on Principles of Distributed Systems (OPODIS 2023).
Editors: Alysson Bessani, Xavier Défago, Junya Nakamura, Koichi Wada, and Yukiko Yamauchi; Article No. 33; pp. 33:1–33:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

computation interacting with the local environment, the memory and resources of the visited computational node, and then it moves on a next computational node by transmitting itself on the network. That is the agent can be seen as an intelligent message with computational capabilities and that is able to decide its next destination.

In the mobile agent paradigm a plethora of problems have been studied. The most famous are: *exploration*, the agents have to collectively visit the entire network; *gathering*, the agents have to reach the same node; *patrolling*, the agents have to periodically patrol the network minimising the time between two visits of the same node.

One problem that has been thoroughly investigated is the Black Hole Search (BHS) [25]. In this problem there exists a dangerous stationary node, called *black hole* (BH), that silently erases from the network all the agents that visit it. A BH node could model several kind of common failures: for instance, consider a crashed host, and any agent trying to visit it will be lost and removed from the network; or a node infected by a virus that cancels the incoming agents. The many works that investigated BH have given to us a complete knowledge of the computational properties of the problem under several assumptions (examples are communication mechanisms employed by the agents, level of synchronicity, topology of the network, and agents' knowledge and capabilities). However, almost all of them examine the case in which the network is *static*: the set of computational nodes and the links connecting them are static and never change.

Recently, research within distributed computing has started to focus on mobile-agents in *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). Highly dynamic graphs 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) (e.g., see [5]).

We consider *evolving graphs*, that are dynamic graphs that can be seen as an infinite sequence of static graphs. In this case the model of computation is by definition *synchronous*, and at each round corresponds a static graph of the aforementioned sequence. A popular assumption in this model is the *1-interval connectivity*: this assumption dictates that at each round the dynamic graph is connected (e.g., [1, 10, 21–23, 26]).

We focus on the case of 1-interval connected rings, where the topology is a ring graph in which at each round at most one edge is missing.

In the last years the body of works studying agents on highly dynamic graphs has been growing with a sustained velocity (for a recent survey see [9]). In particular, a large number of papers is focussing on *1-interval connected rings*: the *gathering* problem has been investigated in [12], the *exploration* problem in [3,4,11] and the BHS for colocated[1] agents in the recent [13]. Despite this large interest, a lot of questions are open. One of them is answered in this paper: how does the computational landscape of finding a BH change when agents are *scattered*? In the scattered case agents start from arbitrary nodes of the graph. We will show that this setting has many differences with respect to the case of colocated agents (studied in [13]) in terms of both solvability (solving BHS in some settings becomes impossible) and complexity.

[1] Agents are colocated when they all start from the same node.

1.2 Related Works

The Black Hole Search (BHS) problem has ben introduced in [15]. The problem has been studied in graph of restricted topologies (e.g., trees [8], rings and tori [6, 17, 24]) and in arbitrary and possibly unknown topology (e.g., [7, 14, 15]). For a recent survey see [25]. The most relevant papers are the ones investigating BHS in static ring networks. In the *asynchronous* setting, it is possible to solve the problem with two colocated agents and $\Theta(n \log n)$ moves, in the whiteboard model [16], and in the pebble model [18]. It has been shown that $\mathcal{O}(n \log n)$ moves also suffices for the scattered case and oriented rings [6]. Others [2] investigated time-optimal algorithms when considering unitary delay.

In spite of all the differences in settings and assumptions, all these investigations share a common trait: the agents operate on a *static* network.

The only works studying BHS in dynamic graph are [13] and [19]. [19] is on the black hole search in *carrier graphs*, a particular class of periodic temporal 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.

[13] studies the BHS in the same setting studied in this paper: 1-interval connected ring with a single BH. [13] shows that three agents are necessary to find the BH (in the static case two agents are enough to explore arbitrary known graphs [7]), and it presents optimal algorithms to find the BH with three agents. Moreover, if agents can communicate only when they are on the same node, the authors show that BHS can be solved in $\Theta(n^2)$ moves and rounds. Finally, if agents can use pebbles, they show an improved algorithm that finds the BH in $\Theta(n^{1.5})$ moves and rounds.

In both studies, the agents are assumed to be initially *colocated*, i.e. to start from the same safe node. To the best of our knowledge, no study considers the case of scattered agents.

1.3 Contributions

We study the problem of finding a BH in an oriented dynamic ring by a set of scattered agents with visible identities under different communication capabilities. We study two main families of communication mechanisms. The *endogenous* family, where the agents can communicate without using any external *facilities.* In this case, they can either see each other only when on the same node (*Vision model*), or they can also talk with each other (*Face2Face model*). In contrast, in the *exogenous* family, the agents communicate using external tools. In this case, we have the *Pebbles model*, in which each agent carries a pebble that can leave on a node to mark it for other agents, or that it can remove from a node in case this marking is not needed anymore; or the *Whiteboard model*, in which each node has a public whiteboard on which agents can write messages of unlimited size.

Our first result (Obs. 6) is that in the endogenous family the BHS is unsolvable using three agents. This is in contrast with the colocated case where BHS is solvable in the same setting by using three agents (see [13]).

To circumvent such impossibility we then rely on *exogenous* communication. We focus on algorithms that use an optimal number of agents. In particular, in [13], it has been shown that BHS in dynamic rings is unsolvable if only two agents are available. This results clearly extends also to the scattered case. Therefore, we will consider algorithms for set of three agents (size optimal algorithms). In Th. 7, we show that any optimal size algorithm solving BHS requires $\Omega(n^2)$ moves and $\Omega(n^2)$ rounds in the whiteboard model. We note that on a static and synchronous rings two agents can find the blackhole in $O(n)$ moves and rounds.

This observation is interesting as it shows that dynamicity does not only increase the number of required agents but it also increases, significatively, the time and moves required. This also highlights the price to pay for having scattered agents: in fact, for dynamic rings, $O(n^{1.5})$ moves and rounds are sufficient in the colocated case [13]. Finally, our lower bound is tight: we provide an algorithm that solves BHS in the pebble model in $O(n^2)$ rounds and moves using three agents (Th. 17).

2 Model and Preliminaries

2.1 The Model and the Problem

The system is a temporal graph where a set of nodes V is connected by a set of edges E. The system is synchronous, and the dynamic networks is an evolving graph $\mathcal{G}$. The time is divided in fictional unites called rounds. The evolving graph can be seen 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. The footprint of the dynamic graph is a static graph containing all the edges that will be present in the system, alternatively, is the union of all the graphs in the aforementioned sequence. An evolving graph where connectivity is guaranteed at every round is called *1-interval connected* (i.e., $\forall G_i \in \mathcal{G}$, G_i is connected). In this paper we focus on *dynamic rings*: 1-interval connected 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} (operations on indices are modulo n). Nodes are anonymous, that is they do not have IDs. A set $A = \{a_0, a_1, \ldots, a_{k-1}\}$ of mobile agents inhabits $\mathcal{R}$. The agents start from distinct arbitrary locations: they are *scattered*. Agents have distinct visible identifiers in $\{0, \ldots, k-1\}$ and they know the total size of the ring n. The agents can move from node to neighbouring node and they have bounded storage ($\mathcal{O}(\log n)$ bits of internal memory suffice for our algorithms). In each round all agents are activated. 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 communicates with the others (the communication mechanism employed will be discussed later). On the basis of the snapshot, the communication, 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, if the edge is present, the agent will reach v' in round $r+1$.

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. In the *FaceToFace (F2F)* model the agent can explicitly communicate among themselves only when they reside on the same node. In the *Vision* model an agents can see all the other agents that reside on the same node (hence count their number); however, they cannot communicate.

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.

The temporal graph $\mathcal{G}$ contains a *black hole* (BH), a node that destroys any visiting agent without leaving any detectable trace of that destruction.

We say that an agent knows the footprint of $\mathcal{R}$ when it knows its left and right distance from the blackhole, that is the agent is able to build in its memory a graph isomorphic to the footprint of $\mathcal{R}$ and it knows its relative position with respect to the blackhole.

▶ **Definition 1** (BHS [13]). *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 knowing the footprint of $\mathcal{R}$. Each agent that terminates has to know the footprint of $\mathcal{R}$.*

Since n is known it is enough that the agent knows its right (or left) distance from the blackhole in order to know the footprint, this means that is not necessary for the agent to visit all nodes of $\mathcal{R}$.

We call *size* the number of agents used by the protocol. Other measures of complexity are the total number of moves performed by the agents, which we shall call *cost*, and *time* it takes to complete the task.

Figure 1 shows (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.

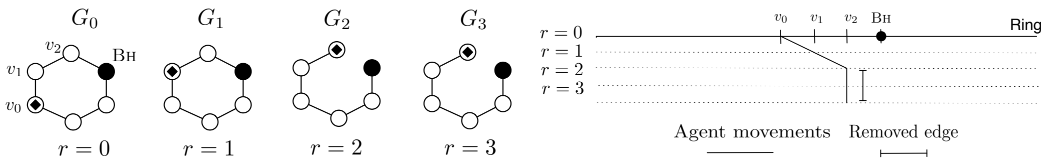


Figure 1 (a) Execution in a dangerous dynamic ring, and (b) its space diagram representation.

3 Preliminaries

Before presenting and analyzing our solution protocols, we report some known impossibility results and a technical lemma that we will use in our paper. Then 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.

3.1 Known Impossibilities

In this section we report known impossibility results.

▶ **Theorem 2** ([13]). *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.*

▶ **Theorem 3** ([13]). *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.*

▶ **Observation 4** ([13]). *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 $V \setminus U$, agents in U explore a node outside the cut at the end of round r if and only if, in round r, there are two agents in U, one that tries to traverse e_c and one trying to traverse e_{cc}, respectively.*

3.2 Cautious Walk

Cautious Walk is a mechanism introduced in [15] 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 scenario.

3.3 Pseudocode Convention and Communication

We use the pseudocode convention introduced in [11]. In particular, our algorithms use as a building block procedure Explore (dir | $p_1 : s_1$; $p_2 : s_2$; …; $p_k : s_k$), 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/Role]: the agent sees another agent with identifier ID (or role *Role*) arriving at the node where it resides, or the agent arrives in a node, and it sees another agent with identifier ID (or role *Role*).

Furthermore, the following variables are maintained by the algorithms:

- $Ttime$, $Tnodes$: the total number of rounds and distinct visited nodes[2], respectively, since the beginning of the execution.
- $Etime$, $Enodes$: 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 pebble model we will also use the CautiousExplore procedure: it is analogous to Explore, with the main difference that the agent uses the pebble to perform a cautious walk. That is, the agent leaves a pebble on the current node, it moves to the node in the moving direction, then it goes back to remove the left pebble, and finally it returns to the recently explored node.

[2] An agent is able to identify if the node where it resides has been previously visited or not by counting the number of steps it has performed in certain direction.

Communication and pebble removal. As for the ability of agents to interact, we 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, we can assume that, in the exogenous models, 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). During a cautious walk using CautiousExplore procedure, agent x might return to a node to retrieve its pebble, even while another agent is present on the same node. In such instances, the two agents meet, but any state transitions initiated by this meeting will only take effect after agent x has successfully reclaimed its pebble.

3.4 CautiousPendulum: An algorithm for colocated agents

In this section we describe the Bhs algorithm CautiousPendulum presented in [13]. The algorithm solves the Bhs when three agents with visible IDs start from the same node. We will use CautiousPendulum as subprocedure in our algorithm for the scattered case (Section 6).

The CautiousPendulum algorithm uses three agents: the Avanguard, the Retroguard, and the Leader. The three agents start on the home-base node v_0.

Agents Avanguard and Leader move clockwise simulating a cautious walk. In particular, If the edge e in the clockwise direction is not present, both agents wait until it reappears. If edge e is present, Avanguard moves to the unexplored node using edge e. Then, if in a successive round the edge e is still present, Avanguard goes back to Leader, signalling that the visited node is safe; at this point, both Leader and Avanguard move clockwise to the recently explored node. If Avanguard does not return when e is present, then the Leader knows that the node visited by Avanguard is the blackhole.

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 moves again counter-clockwise, iterating the same pattern. In case Retroguard finds a missing edge on its path, it waits until the edge re-appears; then it resumes its movement.

If the Leader sees a missing edge e in its clockwise direction and, while waiting for e to appear, does not meet Retroguard after enough time for Retroguard to explore a node and go back, then we say that Agent Retroguard *fails to report* to Leader. In this case, Retroguard entered the black hole, hence the Leader can correctly compute its location.

▶ **Theorem 5** ([13]). *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.*

4 Impossibility with Scattered Agents and Endogenous Communication

When the agents are scattered, three of them, even equipped with the stronger endogenous mechanism (i.e., F2F model), cannot solve Bhs on rings of arbitrary size, as shown by the following:

▶ **Observation 6.** *Three scattered agents in the F2F model cannot solve* Bhs *on a static ring of arbitrary size n, even if they have distinct IDs.*

Proof. Let $\mathcal{A}$ be an algorithm that correctly solves the problem. The proof is by contradiction: we will show that there exists an initial configuration C of the 3 agents on a ring of a proper size $n > 10$ that makes $\mathcal{A}$ fail. We will construct the configuration C in such a way that no two agents meet. Let id_1, id_2, id_3 be the IDs of agents. We consider the behaviour of agent id_i until round r in a run where it executes the algorithm $\mathcal{A}$, and it does not meet any agent. Let $D(a_i, n, r)$ the maximum distance in any direction travelled by an agent until round r. Let r_m be the minimum round at which $D(a_i, n, r) > 0$ for some a_i. Without loss of generality, let id_1 be this agent. We can position the agents so that id_1 is adjacent to the blackhole and enters it at round r_m. Every other agent can be positioned such that they have not met any other agent by round r_m. At this stage, we are left with two agents, and we can invoke Th. 2 to conclude that it is impossible to solve the problem. Note that the premise of Th. 2 is based on co-located agents. However, this scenario is simpler than the dispersed case, so the result is applicable to our context as well. ◀

Fortunately, any exogenous mechanism circumvents the impossibility of Obs. 6. In the following we focus on such mechanisms.

5 Exogenous Communications: Lower Bound for Size-optimal Algorithms

We now consider the Exogenous Communications. Interestingly, we can show a quadratic lower bound on the number of moves and rounds of any size-optimal algorithm that solve that BHS-Problem with scattered agents; the bound holds even if agents have IDs and whiteboards are present.

▶ **Theorem 7.** *In a dynamic ring $\mathcal{R}$ with whiteboards, any algorithm $\mathcal{A}$ for* Bhs *with three scattered agents with unique IDs requires $\Omega(n^2)$ moves and $\Omega(n^2)$ rounds.*

Proof. The proof is by contradiction. Let $\mathcal{A}$ be a sub-quadratic algorithm that solves the BHS-Problem; and let a, b, and c be the three agents. Suppose the agents have unique IDs, and, without loss of generality, let c be the first agent that moves.

Let us assume an initial configuration where c is initially placed on node v_c, neighbour of the Bh, and where a and b are on two neighbours nodes, v_a and v_b, at distance $n/2$ from v_c. Furthermore, w.l.o.g, let us assume that c is placed in such a way that when it moves, at round $r = 0$, it immediately enters Bh; also, let v_c be the counter-clockwise neighbor of the Bh.

Note that, (N1) since c enters the blackhole at round 0 and it can write information on node v_c (whiteboard model), agents a and b can compute the position of the Bh only in two cases: either (N1.1) one of them enters Bh or (N1.2) one of them visits node v_c. We now prove that case (N1.2) might never happen. Let U_r be the partition of nodes explored by agents a and b at the end of round r. By Observation 4, agents a and b may explore a node outside U_r only if they try to traverse at the same round both the edges crossing the cut U_r and $V \setminus U_r$. Let e_c be the clockwise edge incident in U_r, and e_{cc} be the counter-clockwise one. Edge e_c might always be missing, thus preventing the agents from crossing it. Therefore, Bh might only be reached by its clockwise neighbor, and node v_c will never be explored.

Since the blackhole is at distance $\frac{n}{2}$ from both v_a and v_b, by (N1.1) there must exists a set of rounds $r_1, r_2, \ldots, r_{n/8}$ where $U_{r_j} \geq n/8 + j$. We now prove that if $\mathcal{A}$ is sub-quadratic, then in one of these rounds, there must exist an agent, say b, that explores at least two nodes, say v_1 at round r_1 and v_2 at round r_2, such that (1) it does not communicate with a between the two explorations and (2) both a and b visit $o(n)$ disjoint nodes between r_1 and r_2. Assume by contradiction then neither (1) or (2) apply. Then, in each U_{r_j} agent b explores only one node and the agents collectively performs at least $n/8$ moves. Since there are $n/8$ such rounds we have at least $\mathcal{O}(n^2)$ moves. Thus having a contradiction.

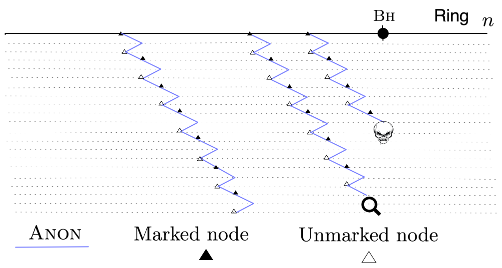
Note that, since the initial configuration is arbitrary and a and b never received any information communicated from c, the positions of v_1 and v_2 do not depend on the positions of Bh and v_c. Therefore, there can exist two initial configurations C_1 and C_2, such that $v_1 = Bh$ in C_1 and $v_2 = Bh$ in C_2. Since b reaches the Bh by round r_2, a is the only agent that can disambiguate between the two configurations. However, a might be blocked indefinitely on a set of nodes that was never visited by b after round r_1 (see Observation 4 – at round r_2 agent a is trying to traverse edge e_c). Consequently, a is not able to distinguish between C_1 and C_2; thus, $\mathcal{A}$ cannot be correct, having a contradiction. Finally, the bound on the number of rounds derives immediately from the bound on moves and from having a constant number of agents. ◀

The above theorem shows the cost optimality of the size-optimal algorithm Gather&Locate described in the following Section 6.

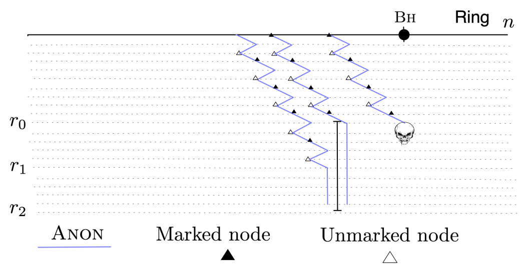
6 An Optimal Exogenous Algorithm: Gather&Locate

In this section, we describe an algorithm to solve the problem with $k = 3$ agents using pebbles. We name the algorithm Gather&Locate. Gather&Locate works in two phases:

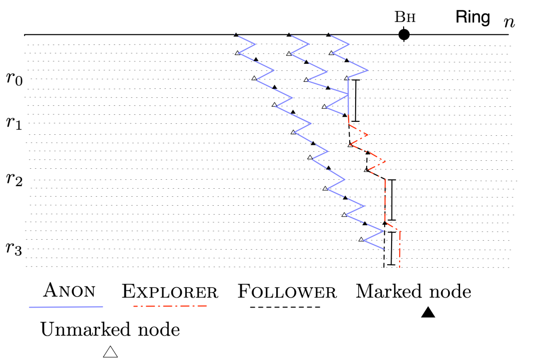
- **Phase 1:** In the first phase agents move clockwise using pebbles to implement a cautious walk. If they meet they synchronise their movements such that at most one of them enters in the black hole. This phase lasts until all three agents meet or $9n$ rounds have passed. We will show that at the end of this phase we have either:
  - (1) three agents are on the same node or on the two endpoint nodes of the same edge. In this case we say that agents gathered;
  - (2) the counter-clockwise neighbour of the black hole has been marked, at most one agent is lost, and the two remaining agents are gathered (that is they are on the same node, or on two endpoints of an edge);
  - (3) one agent is lost in the black hole and the counter-clockwise neighbour of the black hole has been marked. The remaining two agents either both terminated, locating Bh, or only one terminated, with the other still looking for the Bh. In Phase 2 this last agent will either terminate or it will be blocked forever (in both cases the problem is solved).
- **Phase 2:** The second phase starts after the previous one, and relies on the properties enforced by the first phase. In particular, if at the beginning of this phase three agents are on the same node, they start algorithm CautiousPendulum. Otherwise, if two agents are on the same node, they act similarly to Retroguard and Leader in CautiousPendulum. If none of the above applies, then two agents are on the two endpoints of a missing edge, or only a single agent is still looking for the Bh. This scenario is detected by a timeout strategy: upon a timeout, an agent starts moving clockwise looking for the node marked during Phase 1. If two agents meet during this process, they act similarly to Retroguard and Leader in CautiousPendulum. Otherwise, in case a single agent is still active, it will either reach the marked node (and terminate correctly) or it will be blocked forever on a missing edge. We remark that in this last case there has been an agent correctly terminating in Phase 1, and thus BHS-Problem is still correctly solved.



(a) Phase 1: run where BHS-PROBLEM is solved.



(b) Phase 1: run where two agents gather. At round r_0 the rightmost agent enters in the black hole, while the middle agent is blocked. At round r_1 the two remaining agents gathers.



(c) Phase 1: run where three agents gather. At round r_0 the rightmost agent is blocked. At round r_1 two agents meet creating a pair EXPLORER-FOLLOWER. At round r_2 the pair is blocked and the leftmost agent is able to catch up. At round r_3 the tree agents gathered. Note that the meeting predicate of the START with the FOLLOWER triggers at round r_3 and $r_3 - 2$: when an agent is cautious exploring it cannot meet other agents if it has to unmark a node.

Figure 2 Example of runs for Phase 1 of Gather&Locate.

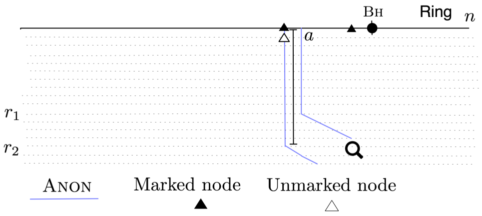
6.1 Detailed Description

The pseudocode of Phase 1 is reported in Algorithms 1, 3, and 2; and Phase 2 in Algorithms 4 and 5. In the pseudocode, we use the predicate $\#A = x$ that is verified when on the current node there are x agents. Initially, all agents have role START.

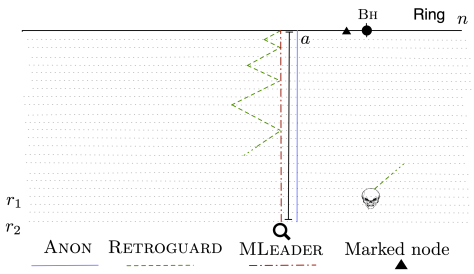
Phase 1. The first phase lasts for at most $9n$ rounds (refer also to the examples in Figure 2). The agents start in state Init of Algorithm 1 and role START: each agent walks cautiously clockwise for $9n$ rounds. If an agent reaches a marked node (predicate *marked*), then it waits until the next node can be deemed as safe or unsafe (see state Wait).

If in the marked node the incident clockwise edge is present and the agent that marked the node does not return, then the next node is the black hole (the agent terminates by triggering predicate $NextUnsafe$).

If two START agents meet on the same node (predicate *meeting*[START]), they synchronise their movements such that they will never cross an edge leading to a possibly unsafe node in the same round. Specifically, the agents enter in the synchronisation state Two, where one agent becomes FOLLOWER (Algorithm 3) and the other becomes EXPLORER (Algorithm 2).



(a) Phase 2: case of two agents gathered on the same edge. At round r_1 agent a triggers the timeout and goes in state Forward starting moving clockwise. At round r_2 it finds the marked node and terminates. Note that the other agent is either unblocked and thus reach the marked node and correctly terminate or it waits forever on a missing edge.



(b) Phase 2: case of three agents gathered, two on the same node the other on the neighbour. At round r_1 the RETROGUARD enters in the black hole. At round r_2 the MLEADER detects that RETROGUARD failed to report an terminates. Note that this happens before a timeouts.

Figure 3 Example of runs for Phase 2 of Gather&Locate.

The role of EXPLORER is to visit new nodes, while FOLLOWER just follows EXPLORER when a node is safe (this is similar to LEADER and AVANGUARD in CautiousPendulum). If the remaining START agent meets with either the FOLLOWER or the EXPLORER, it will assume the behaviour of the FOLLOWER (predicate *meeting*[Follower] in state Init and state Copy). Finally, if the three agents meet on the same node, Phase 1 terminates (see predicate $\#A = 3$ in all states). In all cases, at the end of round $9n$, this phase ends.
In Section 6.2, we will show that, if in Phase 1 all the alive agents have not localised the BH, then either:

- three agents gathered: either three agents are on the same node, or two agents are on a node v and the third agent is blocked on the clockwise neighbour v' of (the marked) node v; or
- the counter-clockwise neighbour of the black hole has been marked, at most one agent is lost, and the two remaining agents are gathered. The two agents are either on the same node, or on two different neighbours node and one of them has marked the node where the other resides.
- the counter-clockwise neighbour of the black hole has been marked, at most one agent is lost, one agent correctly terminated, while the other is still looking for the BH.

Algorithm 1 Gather&Locate; Phase 1 – Algorithm for scattered agents – START.

Predicates Shorthands: $NextUnsafe = Etime > EMtime[C]$
$NextSafe$ = the agent that marked the node returned.
States: {Init, Wait, EndPhase1, Terminate, Copy}.
<u>In state Init:</u>
  CAUTIOUSEXPLORE($right$ | $Ttime = 9n \vee \#A = 3$: EndPhase1; *marked*: Wait; *meeting*[START]: Two; *meeting*[FOLLOWER] $\vee$ *meeting*[EXPLORER]: Copy)
<u>In state Wait:</u>
  EXPLORE(nil | $Ttime = 9n \vee \#A = 3$: EndPhase1; $NextUnsafe$: Terminate; $NextSafe$: Init)
<u>In state Two:</u>
  Use IDs to assign to yourself a role in {FOLLOWER, EXPLORER} in a mutual exclusive fashion.
  Execute the corresponding Algorithm, that is Alg. 3 in state WaitFollower, or Alg. 2 in state Explorer.
<u>In state Copy:</u>
  set your role to FOLLOWER.
<u>In state EndPhase1:</u>
  take the role of START
  starts Phase 2 by entering state InitP2 of Alg. 4.
<u>In state Terminate:</u>
  terminate, BH is the next node in clockwise direction.

Algorithm 2 Gather&Locate; Phase 1 – Algorithm for EXPLORER.

States: {Explorer, Back, MoveForward, EndPhase1, Terminate}. ▷ Terminate and EndPhase1 as in Algorithm 1
In state Explorer:
  if current node is not marked **then**
    mark current node
    EXPLORE($right$ | $Ttime = 9n \vee \#A = 3$: EndPhase1; $Enodes > 0$: Back)
  else
    EXPLORE(nil | $Ttime = 9n \vee \#A = 3$: EndPhase1; $NextUnsafe$: Terminate) ▷ If the node is marked we have to wait to see if it is safe to move
In state Back:
  EXPLORE($left$ | $Ttime = 9n \vee \#A = 3$: EndPhase1; $Enodes > 0$: MoveForward)
In state MoveForward:
  unmark current node
  EXPLORE($right$ | $Ttime = 9n \vee \#A = 3$: EndPhase1; $Enodes > 0$: Explorer)

Algorithm 3 Gather&Locate; Phase 1 – Algorithm for FOLLOWER.

States: {WaitFollower, Follow, EndPhase1}. ▷ EndPhase1 as in Algorithm 1
In state WaitFollower:
  EXPLORE(nil | $Ttime = 9n \vee \#A = 3$: EndPhase1, $Meeting[Back]$: Follow)
In state Follow:
  EXPLORE($right$ | $Ttime = 9n \vee \#A = 3$: EndPhase1, $Enodes > 0$: WaitFollower)

Phase 2. The agents start in InitP2 state of Algorithm 4: here, several checks are executed to understand how Phase 1 ended and to orchestrate the behaviour of the agents. In more details:

- Each agent checks if there are other agents on the same node: in case there are two agents, they get the roles of RETROGUARD and MLEADER (their behaviour is similar to RETROGUARD and LEADER in CautiousPendulum). If there are three agents, they start algorithm CautiousPendulum.
- If the agent is missing the pebble, then it was blocked while trying to recover its pebble at the end of Phase 1. The agent tries to recover the pebble by moving counter-clockwise on one step for $4 \cdot n^2$ rounds. If during this period it succeeds then it goes to state Forward. If $4 \cdot n^2$ rounds have passed without succeeding, then the agent goes in the Forward state. This move has the following goal: if there are two agents on its counter-clockwise node, then they have role RETROGUARD and MLEADER, and in $4 \cdot n^2$ rounds can locate black hole. Otherwise, if there is just one agent on the clockwise node or if there is no one, this timeout avoids that the agent is blocked forever on a missing edge trying to recover a pebble.
- If the agent is on a marked node, then it waits there until either it meets the agent that marked the node, or $4 \cdot n^2$ rounds have passed. If they meet, they get the roles of RETROGUARD and MLEADER; otherwise, if the timeout triggers, the agent goes in state Forward.
- If none of the above applies, the agent goes in state Forward.

We now detail the behaviour of the agents:

- Agent MLEADER and agent RETROGUARD. The MLEADER moves clockwise, while RETROGUARD acts as in Algorithm CautiousPendulum. If RETROGUARD fails to report, MLEADER identifies the black hole and terminates. Finally, if MLEADER and an agent that is not RETROGUARD meet, then this new agent takes the role of AVANGUARD and MLEADER the role of LEADER, and they behave exactly as in Algorithm CautiousPendulum (predicate $meeting[Leader]$ and state BeAvanguard for the agent with role START; and predicate $meeting$[Start] and state GoToCP for the LEADER). The only caveat in this case, is that MLEADER keeps the value of variable $\#Meets[Retroguard]$ when switching to LEADER.

- Agent in state Forward. In state Forward an agent moves in the clockwise direction. If it reaches a marked node, then it discovered the black hole and the agent terminates. If two agents in state Forward meet, they use their IDs to get the roles of MLeader and Retroguard.

Algorithm 4 Gather&Locate; Phase 2 – Algorithm for scattered agents – Start.

Predicates Shorthands: $NextUnsafe = Etime > EMtime[C]$
States: {InitP2, BeAvanguard, Terminate, AssignRoles}.
<u>In state InitP2:</u>
    if $\#A > 1$ **then**
        go to state AssignRoles
    else if my pebble is missing **then**
        Explore($left$ $|meeting$[Start]: AssignRoles; $meeting$[MLeader]: BeAvanguard; $Enodes > 0$: Forward ; $Ttime > 4n^2$: Forward)
    else if the current node is marked **then**
        take the pebble if yours
        Explore(nil $|meeting$[Start]: AssignRoles; $Ttime > 4n^2$: Forward)
    else
        go to state Forward
<u>In state Forward:</u>
    Explore($right$ | $marked \wedge Enodes > 0$: Terminate; $meeting$[Start]: AssignRoles; $meeting$[MLeader]: BeAvanguard)
<u>In state AssignRoles:</u>
    if your pebble is on the node take it.
    if #A=2 **then**
        Use ID to take a role in { Retroguard, MLeader } in a mutual exclusive fashion.
        execute the CautiousPendulum if Retroguard or Alg. 5 in state Go if MLeader.
    else
        Use ID to take a role in { Retroguard, Leader, Avanguard } in a mutual exclusive fashion.
        start algorithm CautiousPendulum.
<u>In state BeAvanguard:</u>
    if your pebble is on the node take it.
    take the role of Avanguard.
    start algorithm CautiousPendulum.
<u>In state Terminate:</u>
    Terminate Bh is the next node in clockwise direction.

Algorithm 5 Gather&Locate; Phase 2 – Algorithm for MLeader.

Predicates Shorthands: $NextUnsafe = Etime > EMtime[C]$.
$FailedReport[Retroguard] = EMtime[C] > 2((\#Meets[Retroguard] + 1) + Tnodes)$.
States: {Go, Cautious, StartCP, Terminate, TerminateR}.
<u>In state Go:</u>
    Explore($right$ | $marked$: Cautious; $meeting$[Start]=StartCP; $FailedReport[Retroguard]$: TerminateR;)
<u>In state Cautious:</u>
    Explore(nil | $meeting$[Start]: StartCP; $NextUnsafe$: Terminate; $FailedReport[Retroguard]$: TerminateR)
<u>In state StartCP:</u>
    start algorithm CautiousPendulum with the role of Leader keeping the value of variable $\#Meets[Retroguard]$.
<u>In state Terminate:</u>
    Terminate, Bh is in the next node in clockwise direction.
<u>In state TerminateR:</u>
    Terminate, Bh is in the node that is at distance $\#Meets[Retroguard] + 1$ from counter-clockwise direction from the reference node.

6.2 Correctness of Gather&Locate.

▶ **Definition 8** (Gathered configuration). *We say that a group of k agents* gathered *if either:*

- *There are k agents on the same node; or,*
- *There are $k-1$ agents on node v_i, and one agent a on node v_{i+1}. Moreover, agent a marked node v_i with a pebble and has to still unmark it.*

Let us first start with a technical lemma, derived from [12], and adapted to our specific case.

▶ **Lemma 9.** *If k agents perform a cautious walk in the same direction for an interval I of $9n$ rounds and one of the alive agents does not explore n nodes and no agent terminates, then the agents gathered.*

Proof. Let A be the set of agents performing a cautious walk, say in clockwise direction, and let a^* be the agent that does not explore n nodes.

Agent a^* can be blocked in progressing its cautious walk in two possible ways: (i) when it is trying to explore a new node using a missing edge in its clockwise direction (we say that a^* is *forward blocked*); (ii) when it is returning to a previously explored node to unmark it (it is blocked by an edge missing in its counter-clockwise direction, and we say that a^* is *backward blocked*). Thus, If in a round r an agent is forward blocked and another one is backward blocked, then they are on two endpoints of the same missing edge.

If a^* is not blocked for $3(n-1)$ rounds then it has explored n nodes. Therefore, a^* has been blocked for at least $6n-3$ rounds or more rounds over an interval of $9n$ rounds. If there is a round r' when a^* is blocked, then every $a \in A$ that at round r' is not blocked does move (note that all blocked agents are either backward or forward blocked on the same edge of a^*).

Thus, all agents in A that are not blocked move towards a^* of at least $\frac{6n-3}{3} = 2n-1$ steps. On the other hand, every time a^* moves, the other agents might be blocked; however, by hypothesis, this can occur less than $3n$ times.

Since the initial distance between a^* and an agent in A is at most $n-1$, it follows that such a distance increases less than $n-1$ (due to a^* movements); however, it decreases by $2n-1$ (due to a^* being blocked). In conclusion, by the end of I, all agents are either at the same node or at the two endpoints of a missing edge and the lemma follows. ◀

▶ **Lemma 10.** *Given three agents executing Phase 1, at most one of them enters the black hole. In this case, the counter-clockwise neighbour node of the black hole is marked by a pebble.*

Proof. If agents have not already met, then each agent performs a cautious walk, all in the same direction, marking a node and avoiding that other agents visit a possibly unsafe node (see state Init in Algorithm 1): when the agent sees a marked node, it goes in state Waits. In this state, the agent waits until it is sure that the next node is safe (that is, until the agent that marked the node returns to remove the pebble).

When two agents meet, they become FOLLOWER and EXPLORER. By construction, FOLLOWER never reaches BH: in fact, FOLLOWER moves a step clockwise only when it sees EXPLORER returning (see state Wait and predicate *meeting*[*Explorer*]); this implies that the node where it moves is safe. Also note that EXPLORER never visits a possibly unsafe node if there is another agent on it: in fact, in state Explore, there is a check on whether the current node is marked or not; if marked, EXPLORER waits (thus, also blocking FOLLOWER) until the next node can be deemed as safe.

If the third agent reaches FOLLOWER, it will also become FOLLOWER and it will never visit an unsafe node. Moreover, the EXPLORER agent always marks a node before visiting its unexplored neighbour (see state Explore of Algorithm 2).

In conclusion, we have that at most one agent enters BH, and the counter-clockwise neighbour node of BH will be marked by a pebble, and the lemma follows. ◀

▶ **Observation 11.** *If an agent terminates while executing Phase 1, then it correctly terminates.*

Proof. The claim follows immediately by observing that the state Terminate is always reached when an agent visits a marked node, the clockwise edge is not missing, and the agent that marked the node does not return. ◀

▶ **Lemma 12.** *Let us consider three agents executing Phase 1. If not all agents terminated locating the* Bh*, then Phase 1 ends by round* $9n$ *and, when it ends, one of the following scenarios holds:*
- *(1) all agents gathered;*
- *(2) at most one agent disappeared in the black hole, the counter-clockwise neighbour of the black hole is marked, and the remaining agents gathered;*
- *(3) one agent terminated, the counter-clockwise neighbour of the black hole is marked, and the remaining agent has to still locate the* Bh*.*

Proof. By construction, in all states the agents check predicate $Ttime = 9n$; thus, Phase 1 ends after at most $9n$ rounds. By Lemma 10 we have that at most one agent enters the Bh, leaving its counter-clockwise neighbour marked. There are three possible cases:
- One agent terminates, and by Observation 11 it terminates correctly solving the BHS-Problem. The other agent has to still locate the Bh
- One agent enters in the Bh and no one terminates. If no alive agent terminates, then no one of them has explored n nodes. Therefore, at the end of Phase 1 $Ttime = 9n$ and by Lemma 9 the agents gathered, and the lemma follows.
- No agent enters the Bh and no agent terminates. In this case we have that three agents gather by the end of Phase 1. If agents end Phase 1 because predicate $\#A = 3$, then the statement immediately follows. Otherwise, $Ttime = 9n$, by Lemma 9 the agents gathered, and the lemma follows. ◀

The next lemma shows that, if Bh has been marked in Phase 1, then two agents executing Algorithm 4 solve BHS-Problem in at most $\mathcal{O}(n^2)$ rounds.

▶ **Lemma 13.** *Let us assume that the counter-clockwise neighbour* v *of* Bh *has been marked by a pebble. If two agents execute Algorithm 4, at least one of them terminates correctly locating the* Bh *in* $\mathcal{O}(n^2)$ *rounds; the other agent either terminates correctly locating the* Bh *or it never terminates.*

Proof. By Lemma 12, at the first round of Phase 2 we have two possible cases:
- The two agents are at the same node. In this case, they immediately enter in state AssignRoles. Let a be the agent that takes the role of MLeader and b be the one that becomes Retroguard. Their movements are similar to the ones of Leader and Retroguard in CautiousPendulum, with the only difference that MLeader moves until it reaches a marked node. By Lemma 12, this marked node is the counter-clockwise neighbour of Bh; thus, if MLeader reaches it, MLeader correctly terminates.
  If MLeader does not visit the marked node because of a missing edge, Retroguard is able to move. By using a similar argument to the one used in the proof of Theorem 5, the black hole is located in at most $\mathcal{O}(n^2)$ rounds, and the lemma follows. Also note that the only agent that can go in a termination state is MLeader, therefore Retroguard cannot terminate incorrectly.
- The two agents occupy two neighbouring nodes, and the most clockwise agent does not have the pebble. More precisely, agent a is at node v, agent b at node v'; also, agent b is missing its pebble, and node v is marked by a pebble. In this case, agent a executes lines 9-10 of Algorithm 4: it removes the pebble from v, and waits for $4 \cdot n^2$ rounds. Agent b executes line 7 of Algorithm 4: it moves towards node v for $4 \cdot n^2$ rounds.

If edge $e = (v, v')$ appears before the timeout, then a and b meet, and previous case applies. Otherwise, both agents go in state Forward. In this state they both move clockwise. If one of them reaches the marked node, it correctly terminates. Otherwise the path towards Bh is blocked by a missing edge and the agents would meet in at most $\mathcal{O}(n)$ rounds. When they meet, they both go in state AssignRoles, and previous case applies again. Note that this implies that at most one of the agents in state Forward can be blocked forever by a missing edge. ◀

▶ **Lemma 14.** *Let us assume that three agents are gathered after Phase 1. Then, if three agents executes Algorithm 4, at least one of them terminates correctly locating the* Bh *in* $\mathcal{O}(n^2)$ *rounds; the other agents either terminate correctly locating the* Bh *or never terminate.*

Proof. If three agents are on the same node, then they start CautiousPendulum algorithm and the correctness follows from Theorem 5. Otherwise, we have two agents on a node v, with v marked with a pebble, and the other agent b on v', with v' the clockwise neighbour of v. Upon the start of Phase 2, the two agents will become Retroguard and MLeader, respectively; MLeader waits on the marked node, while b tries to go back to v.

If edge $e = (v, v')$ is missing for $4n^2$ rounds, then Retroguard has enough time to reach the black hole, and MLeader to terminate because of the fail to report of Retroguard, hence the lemma follows. Note that after the termination of MLeader, the agent b goes in state Forward, it moves clockwise and either it enters the Bh or is blocked forever by a missing edge; in all cases it cannot terminate.

Finally, the last case to analyse is when Retroguard is blocked by a missing edge in the first $4n^2$ rounds. In this case, MLeader and b meet, and algorithm CautiousPendulum starts. The lemma follows by Theorem 5. ◀

▶ **Lemma 15.** *Let us assume that a single agent a starts Phase 2. This agent, by executing Algorithm 4, either terminates correctly or it waits forever on a missing edge.*

Proof. By algorithm construction after at most $4n^2$ rounds from the beginning of Phase 2 the agent a goes in state Forward and it starts moving in clockwise direction. Being the only agent still active, it will never change behaviour until it reaches the marked node or another agent.

By Lemma 12 the counter-clockwise neighbour of Bh is marked, and the terminated agent is located at that node. If no edge is removed forever, a will reach the marked node, and it will terminate. Otherwise, a will be forever blocked on a missing edge. ◀

▶ **Theorem 16.** *Given a dynamic ring* $\mathcal{R}$*, three agents with visible IDs and pebbles running* Gather&Locate*, solve* Bhs *in* $\mathcal{O}(n^2)$ *moves and* $\mathcal{O}(n^2)$ *rounds.*

Proof. By Lemma 12, Phase 1 terminates in at most $\mathcal{O}(n)$ rounds. At this time, either: (1) BHS-Problem is solved and all agents terminated, or (2) the agents gathered, or (3) the counter-clockwise neigbhour of Bh is marked and the remaining agents are gathered, or (4) there is still an agent active while an agent correctly terminated. In case (2), the proof follows by Lemma 14. In case (3), the proof follows by Lemma 13. In case (4), the proof follows by Lemma 15. ◀

By Th. 7 and Th. 16 we have:

▶ **Theorem 17.** *Algorithm* Gather&Locate *is size-optimal with optimal cost and time.*

7 Conclusion

In this paper, we have investigated the BHS on dynamic rings in scenarios where agents are scattered. However, several questions remain open. In particular, we believe that the findings in Observation 6 could potentially be applied to any constant number of agents. Another unresolved issue is the examination of unoriented rings with scattered and anonymous agents, as well as the development of faster BHS algorithm for groups larger than three agents.

References

1 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. doi:10.1007/978-3-319-09620-9_16.

2 B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Time optimal algorithms for black hole search in rings. In *10th International Conference on Combinatorial Optimization and Applications*, pages 58–71, 2010. doi:10.1007/978-3-642-17461-2_5.

3 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. doi:10.1007/978-3-319-49259-9_5.

4 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. doi:10.1109/ICDCS.2017.80.

5 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. doi:10.1080/17445760.2012.668546.

6 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. doi:10.1016/J.TCS.2013.02.010.

7 J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71:229–242, 2006. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi71-2-3-05.

8 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. doi:10.1017/S0963548306008133.

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

10 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. doi:10.1145/2767386.2767442.

11 G.A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Distributed exploration of dynamic rings. *Distributed Computing*, 33:41–67, 2020. doi:10.1007/S00446-018-0339-1.

12 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. doi:10.1016/J.TCS.2018.10.018.

13 G.A. Di Luna, P. Flocchini, G. Prencipe, and N. Santoro. Black hole search in dynamic rings. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 987–997, 2021. doi:10.1109/ICDCS51616.2021.00098.

14 S. Dobrev, P. Flocchini, R. Královič, and N. Santoro. Exploring an unknown dangerous graph using tokens. *Theoretical Computer Science*, 472:28–45, 2013. doi:10.1016/J.TCS.2012.11.022.

15 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. doi:10.1007/S00446-006-0154-Y.

16 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. doi:10.1007/S00453-006-1232-Z.

17 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. doi:10.1007/978-3-540-74466-5_64.

18 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. doi:10.1007/S00453-011-9496-3.

19 P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, 2012. doi:10.1007/S00224-011-9341-8.

20 P. Flocchini, G. Prencipe, and N. Santoro *(Eds.)*. *Distributed Computing by Mobile Entities*. Springer, 2019. doi:10.1007/978-3-030-11072-7.

21 B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In *26th Int. Symp. on Distributed Computing*, pages 166–180, 2012. doi:10.1007/978-3-642-33651-5_12.

22 F. Kuhn, T. Locher, and R. Oshman. Gradient clock synchronization in dynamic networks. *Theory of Computing Systems*, 49(4):781–816, 2011. doi:10.1007/S00224-011-9348-1.

23 F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *42nd ACM Symposium on Theory of Computing*, pages 513–522, 2010. doi:10.1145/1806689.1806760.

24 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. doi:10.1007/978-3-642-35476-2_17.

25 E. Markou and W. Shi. *Dangerous Graphs,* Chapter 18 of [20]. Springer, 2019.

26 R. O'Dell and R.Wattenhofer. Information dissemination in highly dynamic graphs. In *Joint Workshop on Foundations of Mobile Computing*, pages 104–110, 2005. doi:10.1145/1080810.1080828.