Tight Bounds for Black Hole Search with Scattered Agents in Synchronous Rings[☆]

Jérémie Chalopin[a,1], Shantanu Das[b,1], Arnaud Labourel[a,1,*], Euripides Markou[c,2,**]

[a] *LIF, CNRS & Aix Marseille University, 39 rue Joliot Curie, 13453 Marseille, France.*
[b] *Ben-Gurion University & Technion-Israel Institute of Technology, Israel.*
[c] *Department of Computer Science and Biomedical Informatics, University of Central Greece, Lamia, Greece.*

Abstract

We study the problem of locating a particularly dangerous node, the so-called *black hole* in a synchronous anonymous ring network with mobile agents. A black hole is a harmful stationary process residing in a node of the network and destroying all mobile agents visiting that node without leaving any trace. Unlike most previous research on the *black hole search* problem which employed a colocated team of agents, we consider the more challenging scenario when the agents are identical and initially scattered within the network. Moreover, we solve the problem with agents that have constant-sized memory and carry a constant number of identical tokens, which can be placed at nodes of the network. In contrast, the only known solutions for the case of scattered agents searching for a black hole, use stronger models where the agents have non-constant memory, can write messages in whiteboards located at nodes or are allowed to mark both the edges and nodes of the network with tokens. This paper solves the problem for ring networks containing a single black hole. We are interested in the minimum resources (number of agents and tokens) necessary for locating all links incident to the black hole. We present deterministic algorithms for ring topologies and provide matching lower and upper bounds for the number of agents and the number of tokens required for deterministic solutions to the black hole search problem, in oriented or unoriented rings, using movable or unmovable tokens.

Keywords: Distributed Algorithms, Fault Tolerance, Black Hole Search,

[☆] A preliminary version of the paper appeared in the proceedings of SIROCCO 2011 [5].

[*] Principal corresponding author
[**] Corresponding author
Email addresses: jeremie.chalopin@lif.univ-mrs.fr (Jérémie Chalopin), shantanu@tx.technion.ac.il (Shantanu Das), arnaud.labourel@lif.univ-mrs.fr (Arnaud Labourel), emarkou@ucg.gr (Euripides Markou)

[1] Partially supported by ANR projects SHAMAN and ECSPER.
[2] Part of this work was done while this author was visiting the LIF research laboratory in Marseille, France.

Mobile Agents, Anonymous Networks, Identical tokens, Finite State Automata

1. Introduction

1.1. Overview

We consider the problem of exploration in unsafe networks which contain malicious hosts of a highly harmful nature, called *black holes*. A black hole is a node which contains a stationary process destroying all mobile agents visiting this node, without leaving any trace [16]. In the *Black Hole Search* (BHS) problem the goal for a team of agents is to locate the black hole within finite time, with the additional constraint that at least one of the agents must remain alive. In particular, at least one agent must survive and the surviving agents must have located (or marked) all edges leading to the black hole. It is usually assumed that all the agents start from the same location and have distinct identities. In this paper, we do not make such an assumption and study the problem for identical agents starting from distinct locations within the network. We focus on minimizing the resources required to find the black hole.

The only way of locating a black hole is to have at least one agent visiting it. However, since any agent visiting a black hole is destroyed without leaving any trace, the location of the black hole must be deduced by some communication mechanism employed by the agents. Four such mechanisms have been proposed in the literature: a) the *whiteboard* model in which there is a whiteboard at each node of the network where the agents can leave messages (in [2, 3, 13, 15, 16, 17]), b) the *'pure' token* model where the agents carry tokens which they can leave at nodes (in [14, 18, 29]), c) the *'enhanced' token* model in which the agents can leave tokens at nodes or edges (in [1]), and d) the time-out mechanism (only for synchronous networks) in which one agent explores a new node while another waits for it at a safe node (in [7, 8, 10, 11, 24, 25, 26]).

The most powerful inter-agent communication mechanism is having whiteboards at all nodes. Since access to a whiteboard is provided in mutual exclusion, this model could also provide the agents a symmetry-breaking mechanism: If the agents start at the same node, they can get distinct identities and then the distinct agents can assign different labels to all nodes. Hence in this model, if the agents are initially co-located, both the agents and the nodes can be assumed to be non-anonymous without any loss of generality. The BHS problem has been studied using whiteboards in asynchronous networks, with the objective of minimizing the number of agents required to locate the black hole. Note that in asynchronous networks, it is not possible to answer the question of whether or not a black hole exists in the network, since there is no bound on the time taken by an agent to traverse an edge. Assuming the existence of (exactly one) black hole, the minimum sized team of co-located agents that can locate the black hole depends on the knowledge available to the agents. If the agents have a complete map of the network including port numbers and their starting position, then two agents suffice to locate the black hole [15]. If the agents have *sense of direction*, i.e., they can determine if two paths starting from one node

lead to the same node, using only the labels of the ports along these paths, then two agents suffice [15]. If the agents have neither a complete map nor sense of direction, then $\Delta + 1$ agents are needed and suffice [15]. In any case, the prior knowledge of the network size is essential to locate the black hole in finite time.

In the case of synchronous networks two co-located distinct agents can discover one black hole in any graph by using the time-out mechanism, without the need of whiteboards or tokens. Furthermore it is possible to detect whether a black hole actually exists or not in the network. Hence, with co-located distinct agents, the issue is not the feasibility but the time efficiency of black hole search (see [7, 8, 10, 11, 24, 25, 26] for example). However when the agents are scattered in the network (as in our case), the time-out mechanism is not sufficient to solve the problem anymore since the agents need to meet in order to use the mechanism.

Most of the previous results on black hole search used agents whose memory is at least logarithmic in the size of the network (e.g. in [18]). This means that these algorithms are not scalable to networks of arbitrary size. In this paper, we consider agents modeled as finite automata, i.e., having a constant number of states. This means that these agents cannot remember or count the nodes of the network that they have explored. In this model, the agents cannot have prior knowledge of the size of the network. In our model, the agents can detect whether there is an agent of a given state, but not how many of them.

For synchronous ring networks of arbitrary size, containing exactly one black hole, we present deterministic algorithms for locating the black hole using scattered agents each having constant-sized memory. We are interested in minimizing both the number of agents and the number of tokens required for solving the BHS problem.

We use the 'pure' token model. While the whiteboard model is commonly used in unsafe networks, the token model has been mostly used for exploration of safe networks. Note that the 'pure' token model can be implemented with $O(1)$-bit whiteboards (storing the number of tokens at the whiteboard of the node) if we assume that only a constant number of tokens may be placed on a node at the same time, while the 'enhanced' token model can be implemented with $O(\log \Delta)$-bit whiteboards (storing the number of tokens at each incident edge at the whiteboard of the node). In the previous results using the whiteboard model, the capacity of each whiteboard is always assumed to be of at least $\Omega(\log n)$ bits, where n is the number of nodes of the network. Unlike previous models, we do not require mutually exclusive access to the nodes memory, i.e., two agents at the same node are allowed to place tokens simultaneously at that same node of the network. We distinguish movable tokens (which can be picked up from a node and placed on another) from unmovable tokens (which cannot be picked up once they are placed on a node). For both types of tokens, we provide matching upper and lower bounds on both the number of agents and the number of tokens per agent, required for solving the black hole search problem in synchronous rings. Although our algorithms require only a constant size memory for each agent, the impossibility results presented in this paper hold even for agents having unbounded memory.

1.2. Related Works

The exploration of an unknown graph by one or more mobile agents is a classical problem initially formulated in 1951 by Shannon [28] and it has been extensively studied since then (e.g., see [4, 12, 22]). In unsafe networks containing a single dangerous node (black hole), the problem of searching for it has been studied in the asynchronous model using whiteboards and given that all agents initially start at the same safe node (e.g., [2, 3, 13, 15, 16, 17]). It has also been studied using 'enhanced' tokens in [14, 18, 29] and in the 'pure' token model in [1]. It has been proved that the problem can be solved with a minimal number of agents performing a polynomial number of moves. Notice that in an asynchronous network the number of the nodes of the network must be known to the agents otherwise the problem is unsolvable [16]. If the network topology is unknown, at least $\Delta + 1$ agents are needed, where Δ is the maximum node degree in the graph [15]. It is usually assumed that the network is bi-connected and the existence of exactly one black hole is common knowledge.

In asynchronous networks, with scattered agents (not initially located at the same node), the problem has been investigated for arbitrary topologies [6, 21] in the whiteboard model while in the 'enhanced' token model it has been studied for rings [19, 20] and for some interconnected networks [29].

The issue of efficient black hole search has been studied in synchronous networks without whiteboards or tokens (only using the time-out mechanism) in [7, 8, 10, 11, 24, 25, 26] under the condition that all distinct agents start at the same node.

The problem has also been studied for co-located agents in asynchronous and synchronous directed graphs with whiteboards in [9, 26]. In [8] they study how to locate and repair faults (weaker than black holes) using co-located agents in synchronous known networks with whiteboards and in [23] they study the problem in asynchronous networks with whiteboards and co-located agents without the knowledge of incoming link. A different dangerous behavior is studied for co-located agents in [27], where the authors consider a ring and assume black holes with Byzantine behavior, which do not always destroy a visiting agent.

In all previous papers (apart from [1]) studying the Black Hole Search problem using tokens, the 'enhanced' token model is used. The weakest 'pure' token model has only been used in [1] for co-located agents in asynchronous networks. In all previous solutions to the problem using tokens, the agents are assumed to have non-constant memory.

1.3. Our Contributions

Unlike previous studies on BHS, we consider the scenario of anonymous (i.e., identical) agents that are initially scattered in an anonymous ring. We focus our attention on very simple mobile agents. The agents have constant-size memory, they carry a constant number of identical tokens which can be placed at nodes and, apart from using the tokens, they can communicate with other agents only when they meet at the same node. We consider four different scenarios depending on whether the tokens are movable or not, and whether the agents

agree on a common orientation. We present deterministic optimal algorithms and provide matching upper and lower bounds for the number of agents and the number of tokens required for solving BHS (See Table 1 for a summary of results). Surprisingly, the agreement on the ring orientation does not influence the number of agents needed in the case of movable tokens but is important in the case of unmovable tokens.

The lower bounds presented in this paper are very strong in the sense that they do not allow any trade-off between the number of agents and the number of tokens for solving the BHS problem. In particular we show that:

- Any constant number of agents, even having unlimited memory, cannot solve the BHS problem with less tokens than depicted in all cases of Table 1.
- Any number of agents less than that depicted in all cases of Table 1 cannot solve the BHS problem even if the agents are equipped with any constant number of tokens and they have unlimited memory.

Meanwhile our algorithms match the lower bounds, are asymptotically time-optimal and since they do not require any knowledge of the size of the ring or the number of agents, they work in any anonymous synchronous ring, for any number of anonymous identical agents (respecting the minimal requirements of Table 1).

		Resources necessary and sufficient		
Tokens are	**Ring is**	**# agents**	**# tokens**	**References in the paper**
Movable	Oriented	3	1	Theorem 3.1, 3.2 and 4.1
	Unoriented			
Unmovable	Oriented	4	2	Theorem 3.1, 3.3 and 5.1
	Unoriented	5	2	Theorem 3.1, 3.4 and 5.2

Table 1: Summary of results for BHS in synchronous rings

In Section 2, we formally describe the model and the different settings considered in the paper. In Section 3, we state impossibly results for all settings considered in the paper. In Section 4, we give an algorithm for agents with movable tokens. In Section 5, we give two algorithms for agents with unmovable tokens, one for oriented rings and another for unoriented rings. Finally, in Section 6, we conclude and give some perspectives.

2. Our Model

Our model consists of an anonymous, synchronous ring network with $k \geq 2$ identical mobile agents that are initially located at distinct nodes called *home-bases*. Each mobile agent owns a constant number t of identical tokens which

can be placed at any node visited by the agent. The tokens are indistinguishable. Any token or agent at a given node is visible to all agents on the same node, but not visible to agents on other nodes. The agents follow the same deterministic algorithm and begin execution at the same time and being in the same initial state. In all our protocols a node may contain at most two tokens at the same time. At any node of the ring, the ports leading to the two incident edges are distinguishable and locally labelled and an agent arriving at a node knows the port-label of the edge through which it arrived. In the special case of an oriented ring, the ports are consistently labelled as `Left` and `Right` (i.e., all ports going in the clockwise direction are labelled `Left`). In an unoriented ring, the local port-labeling at a node is arbitrary and each agent in its first step chooses one direction as `Left` and in every subsequent step, it translates the local port-labeling at a node into `Left` and `Right` according to its chosen orientation. In order to make this translation, the agent stores the port-label of the edge through which it arrived at the current node and the direction of the last movement (Left or Right). If the direction of the next move is the opposite of that of the last move, the agent moves using the port-label stored. Otherwise, the agent moves using the other port-label.

In a single time unit, each mobile agent completes one *step* which consists of the *Look*, *Compute* and *Move* stages (in this order). During the Look stage, an agent obtains information about the configuration of the current node (i.e., agents, tokens present at the node) and its own configuration (i.e., the port through which it arrived and the number of tokens it carries). During the Compute stage, an agent can perform any number of computations (i.e., computations are instantaneous in our model). During the Move stage, the agent may put or pick up a token at the current node and then either move to an adjacent node or remain at the current node. If during the computation stage, the agent detects that one of neighbors of the current node is the black hole, then the agent may permanently mark the link as dangerous, during the Move stage. Since the agents are synchronous they perform each stage of each step at the same time. We call a token *movable* if it can be put on a node and picked up later by any mobile agent visiting the node. Otherwise we call the token *unmovable* in the sense that, once released, it can occupy only the node where it has been released.

Formally we consider a mobile agent as a finite Moore automaton $\mathcal{A} = (\mathcal{S}, S_0, \Sigma, \Lambda, \delta, \phi)$, where $\mathcal{S}$ is a set of $\sigma \geq 2$ states among which there is a specified state S_0 called the *initial* state; $\Sigma \subseteq \mathcal{D} \times \mathcal{C}_v \times \mathcal{C}_A$ is the set of possible configurations an agent can see when it enters a node; $\Lambda \subseteq \mathcal{D} \times \mathcal{P} \times \mathcal{X}$ is the set of possible actions by the agent; $\delta : \mathcal{S} \times \Sigma \to \mathcal{S}$ is the transition function; and $\phi : \mathcal{S} \to \Lambda$ is the output function. $\mathcal{D} = \{\texttt{left}, \texttt{right}, \texttt{none}\}$ is the set of possible directions through which the agent arrives at or leaves a node (`none` represents no move by the agent). $\mathcal{P} = \{\texttt{put}, \texttt{pick}, \texttt{no action}\}$ is the action performed by the agent on the tokens, while $\mathcal{X} = \{\texttt{mark left}, \texttt{mark right}, \texttt{no action}\}$ is the action performed by the agent on the links incident to the current node. $\mathcal{C}_v = \{0,1\}^\sigma \times \{0,1,2\}$ is the set of possible configurations at a node, consisting of (i) a bit string that denotes for each possible state whether there is an agent in that

state at that node and (ii) an integer that denotes the number of tokens at that node (in our protocols at most 2 tokens reside at a node at any time). Finally, $\mathcal{C}_A = \{1, 2\} \times \{0, 1, 2\}$ is the set of possible configurations of an agent, i.e., its orientation and whether it carries zero, one or two tokens (in our protocols, an agent cannot carry more than two tokens). Observe that this definition is only used for our algorithms since our impossibility results works even if agents have unlimited memory, can carry and see at a node any number of tokens.

Notice that all computations by the agents are independent of the size n of the network and the number k of agents. The agents have no knowledge of n or k. The agents only know the number of tokens they have. Since the agents are identical they face the same limitations on their knowledge of the network. There is exactly one black hole in the network. An agent can start from any node other than the black hole and no two agents are initially colocated[3]. Once an agent detects a link to the black hole, it marks the link permanently as dangerous. We require that at the end of a black hole search scheme, all links incident to the black hole (and only those links) are marked dangerous and that there is at least one surviving agent. The time complexity of a BHS scheme is the number of time units needed for completion of the scheme, assuming the worst-case location of the black hole and the worst-case initial placement of the scattered agents.

Note that our definition of a successful BHS scheme is slightly different from the traditional definition. Indeed, in the original definition, it was required that there is at least one surviving agent, and this agent knows the location of all edges incident to the black hole. However, since we consider finite state agents, it is not possible for the agents to remember the location of the edges incident to the black hole. Thus we need the additional capability of marking these links as dangerous, for the purpose of reporting the solution. During the execution of the algorithm, the agents cannot see which edges are marked dangerous (by other agents). Hence, this capability does not provide any additional power of communication to the agents.

3. Impossibility Results

3.1. Oriented Rings

We first show that when the tokens are unmovable, a team of any constant number of agents needs at least two tokens per agent to solve the BHS problem.

Theorem 3.1. *For any constant k, there exists no algorithm that solves BHS in all oriented rings containing one black hole and k or more scattered agents, when each agent is provided with only one unmovable token. The result holds even if the agents have unlimited memory.*

[3]Since there is no symmetry breaking mechanism at a node, two agents starting at the same node and in the same state, would behave as a single (merged) agent.

Proof : Suppose there is a correct BHS algorithm that solves the problem with k or more agents in rings of any size. If the algorithm does not require any agent to put down its token, such an algorithm should work even if every agent puts its token on its homebase in the first step. So without loss of generality, we assume that an agent puts down its token after executing a finite number of steps of the algorithm (unless it encounters some agents, some tokens or the black hole within this time). Now consider the behavior of this agent when placed on an infinite line (with no other agent). Suppose the agent puts down its token at a distance of x (w.l.o.g. in the left direction) from its homebase. Further let p be the maximum distance that the agent has travelled from its homebase (in either direction) before it puts down its token (thus, $x \leq p$). Now, consider a ring R_1 of size $n = 2k(p+1)$ with one black hole and k agents such that the agents are initially placed at a distance of $2(p+1)$ apart (see Figure 1(a)). The black hole is located in the middle of a segment between two consecutive agents (i.e., it is at a distance $(p+1)$ from the closest agents). Since the agents start in the same state, they take the same actions and remain in identical states (until they encounter another agent, or a token or the black hole). As long as the agents do not travel any further than a distance p from their homebase, they will not see anything different from the agent on the infinite line. Thus, each agent will put its token $x \leq p$ places to the left of its homebase. Each agent will do so at the same time and in the same state. During the rest of the algorithm, an agent can only move, observe the tokens and possibly mark some link as dangerous. Due to the correctness of the algorithm, at some time τ, some agent will mark one link leading to the black hole as dangerous. Up to time τ, all surviving agents behave the same since they see at each step the same configuration at their nodes. If there are more than one surviving agent, all of them are in the same state β and still at a distance of $2(p+1)$ apart. Thus, if one such agent marks a link as dangerous, the next agent would mark a link at a distance of $2(p+1)$ away. So, one of the agents would have incorrectly marked a link—a contradiction to the correctness of the algorithm.

The remaining case to consider is when the agent that marks a link, is the last surviving agent. In this case, we can construct another ring R_2 of size $n = 2(k+1)(p+1)$ with one black hole and $(k+1)$ agents initially placed the same distance apart as in the ring R_1 (see Figure 1(b)). In an execution of the same algorithm on ring R_2, after exactly τ time steps, there will be two surviving agents both in the same state β and at a distance of $2(p+1)$ from each other. Thus the two agents will mark as dangerous, two distinct links at a distance of $2(p+1)$ apart. Hence the algorithm fails for the ring R_2. □

We now derive some lower bounds on the number of agents necessary to solve the BHS problem. The following result proves that at least one agent needs to be sacrificed for detecting each link leading to the black hole.

Lemma 3.1. *During any execution of any BHS algorithm, if a link to the black hole is correctly marked, then at least one agent must have entered the black hole through this link.*

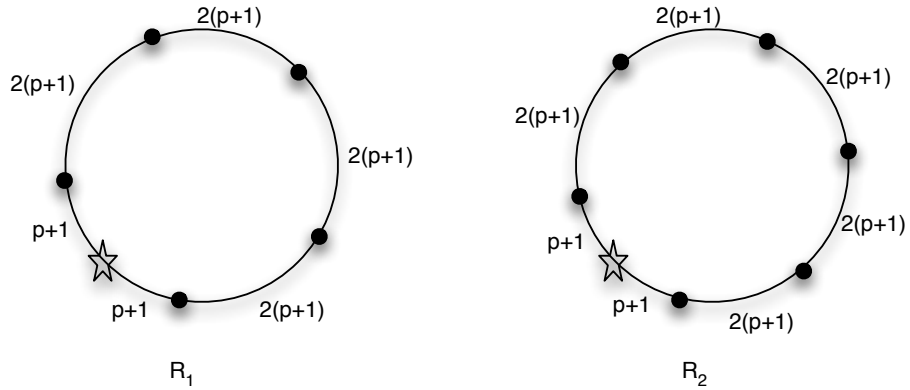

Figure 1: (a) An oriented ring R_1 with k agents and a black hole, (b) a larger oriented ring R_2 with $k+1$ agents and one black hole.

Proof : Suppose for the sake of contradiction that there exists a correct BHS algorithm such that during any execution of this algorithm one link incident to the black hole is marked before any agent traverses this link. Assume without loss of generality that the link is on the left of the black hole. Consider the ring of size n which leads to this execution. Now, add a vertex on the left of the black hole, obtaining a ring of size $n+1$, while keeping the same initial positions for the agents. The agents will behave as in the ring of size n since they do not know the size of the ring and will see exactly the same configuration. Hence they will mark the left link of the new node as the link leading to the black hole. This contradicts with the correctness of the BHS algorithm. □

To solve the BHS problem in a ring, both links leading to the black hole need to be marked as dangerous. Thus, we immediately arrive at the following result.

Theorem 3.2. *Two mobile agents carrying any number of movable (or unmovable) tokens each, cannot solve the BHS problem in an oriented ring, even if the agents have unlimited memory.*

When the tokens are unmovable, even three agents are not sufficient to solve BHS as shown below.

Theorem 3.3. *Three mobile agents carrying a constant number of unmovable tokens each, cannot solve the BHS problem in an oriented ring, even if agents have unlimited memory.*

Proof : Suppose for the sake of contradiction that there exists an algorithm which solves the BHS problem for three agents each carrying a constant number t of unmovable tokens. Let x and y be two integers chosen by the adversary, such that $1 \leq x, y \leq 2t$. Suppose the three agents are initially placed on a ring of size $8t + x + y$ such that the distance between the first and second and between the second and third agent is $4t$. The black hole is between the third

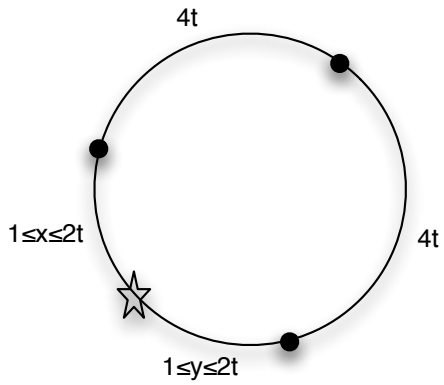

Figure 2: Three agents with t unmovable tokens each in an oriented ring.

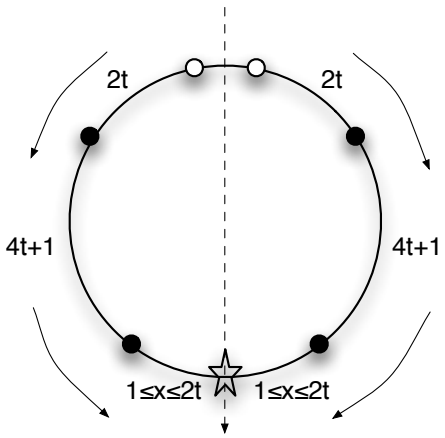

Figure 3: Four agents with t unmovable tokens each in an unoriented ring.

and the first agent at a distance x from one of them and at distance y from the other (as in Figure 2). By Lemma 3.1, at least one agent would fall into the black hole before any link to the black hole is identified. Consider the phase P of the algorithm from the start until the first time an agent falls into the black hole. Let us call this agent a. Assume without loss of generality that agent a enters the black hole by going right (i.e., after traveling a distance of x from its homebase). First, notice that the agents never meet each other during phase P since they execute the same algorithm.

Suppose for the sake of contradiction that after agent a has vanished, the two surviving agents can identify the link used by agent a to enter the black hole without sacrificing another agent. This is only possible if, whenever agent a explores a new node to the right, it leaves a message encoding this fact and the only way to do this is by leaving another token at some node of the ring that may already have tokens. However, after t explored nodes agent a runs out of tokens. The adversary may then set x to be any value between $t+1$ and $2t$. The remaining agents would not have enough information to determine the position of the black hole from the left (without the sacrifice of another agent). However, by Lemma 3.1, at least one agent must enter the black hole from the other link (on the right). Thus either one of the links to the black hole is never marked or there are no surviving agents. □

3.2. Unoriented Rings

In an unoriented ring, even four agents do not suffice to solve the BHS problem with unmovable tokens. In fact we show a stronger result that it is not even possible to identify just one of the links to the black hole, using four agents.

Theorem 3.4. *In an unoriented ring, four agents carrying any constant number of unmovable tokens each, cannot correctly mark any link incident to the black hole, even when the agents have unlimited memory.*

Proof : Suppose for the sake of contradiction that there exists an algorithm which marks one of the links incident to the black hole, using four agents each carrying t unmovable tokens. For some integer x, $1 \leq x \leq 2t$, chosen by the adversary, suppose that the four agents and the black hole are initially placed as in Figure 3. The distance between two consecutive agents is $4t + 1$ and the distance between the black hole and the closest agent on each side is x. Thus, the initial configuration is symmetric and the axis of symmetry crosses an edge and the black hole. The adversary can choose the orientations of the agents in such a way that the two agents closest to the black hole would fall into the black hole at the same time and before any agent meets another agent. Thus, the two surviving agents would continue to be in symmetric situation and they would take similar actions. Using the same argument as in the proof of Theorem 3.3, the information left by the vanished agent is not sufficient for one agent to correctly identify any link incident to the black hole. Due to the symmetry of the resulting configuration (and the fact that agents cannot meet on an edge), the two remaining agents can never meet and will always be in the same state until they both fall into the black hole (without marking any of the links incident to the black hole). □

4. BHS Scheme with Movable Tokens

We first consider the case when the agents have movable tokens. If each agent has a movable token it can perform a cautious walk [16]. The `Cautious-Walk` procedure consists of the following actions: Put the token at the current node, move one step in the specified direction, return to pick up the token, and again move one step in the specified direction (carrying the token). After each invocation of the Cautious Walk, the agent looks at the configuration of the current node[4] and decides whether to continue performing Cautious Walk.

We show that only three agents are sufficient to solve BHS, when they have one movable token each. Algorithm 1 achieves this, both for oriented and unoriented rings. The procedure `Mark-Link` permanently marks as dangerous the specified link.

Theorem 4.1. *Algorithm 1 solves the BHS problem in an unoriented ring with $k \geq 3$ agents having constant memory and one movable token each.*

Proof : Notice that all agents start at the same time executing Procedure *CautiousWalk(dir)* and at each time they are at the same phase of this procedure.

[4]Recall that only the tokens put on the node are counted, not the tokens carried by the agent itself.

Algorithm 1: BHS-Ring-1

```
/* BHS in unoriented ring using k ≥ 3 agents having 1 movable token each  */
repeat CautiousWalk(Left);
until current node has a token and no agent;
;
Mark-Link(Left);
repeat CautiousWalk(Right);
until current node has a token and no agent;
;
Mark-Link(Right);
```

Procedure CautiousWalk(*direction*)

```
Put a token;
Move one step in specified direction;
Move one step back;
Pick a token;
Move one step in specified direction;
```

Since the only time an agent checks the number of tokens it sees is after completing an execution of the procedure, if the agent sees at least one token (not including the one it carries) then, either (i) there is another agent at the current node (i.e., the two agents arrived from opposite directions) or (ii) there is no other agent (which means that the token was left by an agent that disappeared). In case (i) the agent continues executing Procedure *CautiousWalk(dir)*. In case (ii) it is clear that the black hole resides at the next node in direction *dir*. In this case, the agent marks the edge to the black hole, reverses direction and repeats the process. Since the agents start from distinct locations at the same time, no two agents can arrive at the black hole at the same time through the same link. Thus, exactly one agent would fall into the black hole from each direction (leaving at least one surviving agent) and both links to the black hole would be eventually marked. □

5. BHS Scheme with Unmovable Tokens

For agents having only unmovable tokens, we use the technique of *Paired Walk* (called *Probing* in [7]) for exploring new nodes. The procedure is executed by two co-located agents with different roles and the same orientation. One of the agents called the *leader* explores an unknown edge while the other agent, called *follower* waits for the leader. If the other endpoint of the edge is safe, the leader immediately returns to the previous node to inform the follower and then both move to this new node. On the other hand, if the leader does not return in two time steps, the follower knows that the next node is the black hole. (See Procedure *Paired Walk*).

In order to use the *Paired Walk* technique, we need to gather two agents at the same node and then break the symmetry between them, so that distinct

roles can be assigned to each of them. The basic idea of our algorithms is the following. We first identify the two homebases that are closest to the black hole (one on each side). These homebases are called *gates*. The gates divide the ring into two segments: one segment contains the black hole (thus, is dangerous); the other segment contains all other homebases (and is safe). Initially all agents are in the safe part and an agent can move to the dangerous part only when it passes through the gate node. We ensure that any agent reaching a gate node, waits for a partner agent in order to perform the *Paired Walk* procedure. We now present two BHS algorithms, one for oriented rings and the other for unoriented rings.

Procedure PairedWalk(isLeader)

```
if isLeader then
    Move one step in specified direction;
    Move one step back;
    Move one step in specified direction;
else
    WAIT(2);
    if there is a leader then
        Move one step in specified direction;
```

5.1. Oriented Rings

In this section, we describe an algorithm using at least four agents with two unmovable tokens. In an oriented ring, all agents may move in the same direction (i.e., Left). The algorithm executed by the agents essentially runs in five phases:

1. **Init phase:** During this phase, each agent places a token on its homebase (state START), moves left until the next homebase, i.e., next node with a token (state CHECK-LEFT), returns to its homebase to put down the second token (state GO-BACK). The agents may not complete this phase of the algorithm at the same time. If the agent meets another agent at its homebase, it forms a pair with it (entering phase Left-exploration) if the other agent is alone, or it becomes a LEFT-SEARCHER agent otherwise (entering phase Left-pairing). During this phase, only one agent will fall into the black hole and there will be a unique homebase with a single token (we call this node the *gate* node) and all the other homebases will eventually contain exactly two tokens each.

2. **Left-pairing:** During this phase, agents try to meet at the gate node by moving to the left until they reach a node containing a single token (state ALONE). The first agent reaching node with one token waits for a partner agent (state WAITING). When another agent arrives at the node, they form a pair (LEFT-LEADER, LEFT-FOLLOWER) and proceed to the next phase. A pair can be formed either by :

- a WAITING agent and an ALONE agent,
- or a WAITING agent and a GO-BACK agent,
- or an ALONE agent and a GO-BACK agent.

In all the above cases, the two agents forming a pair have distinct states and they can be assigned the distinct roles of leader and follower for the next phase. Among the two agents forming a pair, if there is a WAITING agent, it becomes LEFT-LEADER, else the ALONE agent becomes LEFT-LEADER. The algorithm also has some additional rules to ensure that no two LEFT-LEADERs are created at the same node at the same time: no agent becomes a LEFT-LEADER if there is already another LEFT-LEADER at the same node (In this case, the agent become a LEFT-SEARCHER). Another rule is that if a ALONE agent and a GO-BACK agent arrive at the same time to a node with a WAITING agent, then only the WAITING and GO-BACK agents form a pair (LEFT-LEADER, LEFT-FOLLOWER) while the ALONE agent becomes a LEFT-SEARCHER.

3. **Left-exploration:** During this phase, the pair formed during the previous phase performs *Paired Walk* in the left direction. One of the agents of a pair (state LEFT-LEADER) eventually falls into the black hole and the other agent (state LEFT-FOLLOWER) marks the edge leading to the black hole.

4. **Right-pairing:** During this phase, the LEFT-FOLLOWER agent of the previous phase changes state to RIGHT-SEARCHER and returns to the gate node by moving to the right. On reaching the gate node, this agent forms a pair with the RIGHT-FOLLOWER or RIGHT-LEADER agent at the node, if any, otherwise it waits for a partner agent to arrive in order to form a pair. Recall that any additional agent that did not pair-up during the *Left-pairing* phase of the algorithm became a LEFT-SEARCHER. Such a LEFT-SEARCHER agent moves left until reaching the gate node and waits for a RIGHT-LEADER in order to form a pair (in state RIGHT-FOLLOWER).

5. **Right-exploration:** During this phase, the pair formed during the previous phase performs *Paired Walk* in the right direction. One of the agents of a pair (state RIGHT-LEADER) eventually falls into the black hole and the other agent (state RIGHT-FOLLOWER) marks the edge leading to the black hole.

The complete formal version of the algorithm is presented in Algorithm 2 and the state transitions during the algorithm are shown in Figure 4.

Lemma 5.1. *During the algorithm 2, the following holds, assuming there are at least* 4 *agents, each carrying two unmovable tokens*

(i) Exactly one CHECK-LEFT agent falls into the black hole.

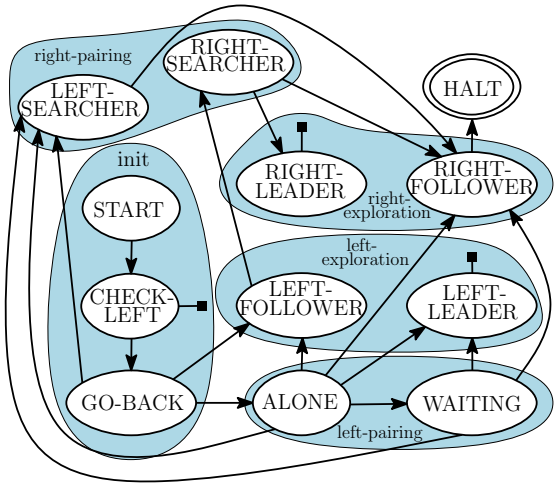

Figure 4: State transitions during the algorithm BHS-Ring-2 with $k \geq 4$ agents and 2 unmovable tokens per agent. Agents in state CHECK-LEFT, LEFT-LEADER, or RIGHT-LEADER may fall into the black hole.

(ii) An agent in any state other than CHECK-LEFT, LEFT-LEADER, or RIGHT-LEADER, never enters the black hole.

(iii) At least one Paired Walk *is performed in the left direction starting from the gate node, marking one edge incident to the black hole as dangerous.*

(iv) At least one Paired Walk *is performed in the right direction starting from the gate node, marking one edge incident to the black hole as dangerous.*

Proof : (i) After the first step (performed simultaneously by each agent), there is at least one token at each homebase. A CHECK-LEFT agent moves from its homebase until the next node on the left that contains a token. Thus, only the agent whose homebase is the first one on the right starting from the black hole, may enter the black hole. Any other agent would successfully reach a node containing a token and at that point, change state and never enter state CHECK-LEFT again.
(ii)Any agent that is not in state CHECK-LEFT never goes beyond the gate node in the left direction, unless it is performing a *Paired Walk* as a (LEFT-LEADER, LEFT-FOLLOWER) pair. Thus, only a LEFT-LEADER or a CHECK-LEFT agent may enter the black hole while going left. Now consider an agent that is going in the right direction. If the agent is going back to the gate node or back to its homebase, it stops before reaching the black hole. Otherwise an agent going right must be part of a *Paired Walk* in the right direction. Thus, only a RIGHT-LEADER may enter the black hole while going in the right direction.
(iii) An agent that does not enter the black hole in state CHECK-LEFT, returns to its homebase to put the second token. If the agent meets another agent at its

homebase then a *Paired Walk* (in the left direction) is started at this homebase, eventually arriving at the gate node; thus the property holds. Otherwise, the agent is in state ALONE on arriving at its homebase and it moves left until the next homebase node. Suppose for the sake of contradiction that no *Paired Walk* is started in the left direction. This means that all the surviving agents are in state ALONE or WAITING. An agent can wait only at a homebase node containing a single token. If an agent is waiting at a node that is not the gate node, then eventually the owner of the homebase would return to that node to put the second token. At this point a *Paired Walk* in the left direction would be started from this node. Thus, due to the assumption that no *Paired Walk* has been started, all agents that are in state ALONE or WAITING would eventually be at the gate node. Since there are at least three such agents, two of them would meet at the gate node and thus, start a *Paired Walk* together.
(iv) Note that an agent may become LEFT-LEADER only if there are no other LEFT-LEADER agents at the same node. Thus, two LEFT-LEADER agents cannot enter the black hole at the same time. This means that two agents cannot become RIGHT-SEARCHER at same time. From (iii), we know that at least one agent will become a RIGHT-SEARCHER (the LEFT-FOLLOWER agent of a *Paired Walk* in the left direction eventually becomes RIGHT-SEARCHER). The first agent in state RIGHT-SEARCHER to reach the gate node, will become a RIGHT-LEADER. If there is another RIGHT-SEARCHER agent, this agent will eventually reach the gate node, become a RIGHT-FOLLOWER, and start a *Paired Walk* in the right direction with RIGHT-LEADER. Otherwise, if no *Paired Walk* in the right direction has been started then there are at least two surviving agents and all surviving agents would eventually be at the gate node. Any agents in state ALONE or WAITING at the gate node will become a RIGHT-FOLLOWER and thus join in a *Paired Walk* with the RIGHT-LEADER. □

Theorem 5.1. *Algorithm BHS-Ring-2 correctly solves the black hole search problem in any oriented ring with* 4 *or more agents having constant memory and carrying two unmovable tokens each.*

Proof : Due to Lemma 5.1, we know that at least one *Paired Walk* is performed in each direction during the algorithm. Thus, both links to the black hole are discovered and marked. Further we know that the RIGHT-FOLLOWER agent in the pair performing the *Paired Walk* to the left will never enter the black hole. Thus there is at least one surviving agent. □

5.2. Unoriented Rings

For unoriented rings, we need at least 5 agents with two unmovable tokens each. The algorithm for unoriented rings with unmovable tokens is similar to the one for oriented rings, except that each agent initially chooses an orientation. When two agents meet and one has to follow the other, we assume in our model that the state of the agent contains information about the orientation of the

Algorithm 2: BHS-Ring-2

```
/* BHS in an Oriented Ring, using k ≥ 4 agents having 2 unmovable tokens
   each.  All agents have the same initial state START.                  */
START:
    Decide to place a token at homebase; State := CHECK-LEFT;
CHECK-LEFT:
    Move Left until the next node that contains a token; State := GO-BACK ;
GO-BACK:                                   /* Go home to put the second token */
    Move Right until the next node that contains a token;
    Decide to place the second token;
    if there is a LEFT-LEADER agent then State := LEFT-SEARCHER;
    ;
    else if there is a ALONE agent or a WAITING agent then
        State := LEFT-FOLLOWER;
    ;
    else State := ALONE;
    ;
ALONE:                                 /* Move alone to the node with single token */
    Move Left until a node that contains either only one token or two tokens and a
    GO-BACK agent ;
    if there is a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    else if there is a LEFT-LEADER agent then State := LEFT-SEARCHER;
    ;
    ;
    else if there is a GO-BACK agent and no WAITING agent then
        State := LEFT-LEADER;
    ;
    else if there is a WAITING agent and no GO-BACK agent then
        State := LEFT-FOLLOWER;
    ;
    else if there is a WAITING agent and a GO-BACK agent then
        State := LEFT-SEARCHER;
    ;
    else State := WAITING;
    ;
WAITING:                                           /* Wait for a partner agent */
    Wait until other agents arrive at the current node;
    if there is a LEFT-LEADER agent then State := LEFT-SEARCHER;
    ;
    else if there is a GO-BACK agent or ALONE agent then
    State := LEFT-LEADER;
    ;
    ;
    else if there is a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    ;
LEFT-LEADER:                          /* Perform Paired walk with Follower agent */
    while true do PairedWalk (1) in Left direction;
    ;
```

Algorithm 2: BHS-Ring-2 (continued)

```
/* BHS in an Oriented Ring, using k ≥ 4 agents having 2 unmovable tokens
   each.  All agents have the same initial state START.                  */
LEFT-FOLLOWER:                  /* Perform Paired walk with Leader agent */
    while there is a LEFT-LEADER agent do
        PairedWalk (0) in Left direction;
        if the LEFT-LEADER did not return during the last step then
            Mark-Link (Left); State := RIGHT-SEARCHER; exit loop;
LEFT-SEARCHER:                  /* Move Left to become a RIGHT-FOLLOWER */
    while there is no RIGHT-LEADER agent do
        repeat
            Move Left;
        until not at a node with one token;
        Wait until there is another agent;
    State := RIGHT-FOLLOWER;
RIGHT-SEARCHER:                 /* Move Right to become a RIGHT-LEADER or a
RIGHT-FOLLOWER */
    while not at a node with one token do Move Right;
    ;
    if there is a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    else  State := RIGHT-LEADER;
    ;
RIGHT-LEADER:                   /* Perform Paired walk in the other direction */
    Wait until there is another agent;
    while true do  PairedWalk (1) in Right direction;
    ;
RIGHT-FOLLOWER:                 /* Perform Paired walk in the other direction */
    while there is a RIGHT-LEADER agent do
        PairedWalk (0) in Right direction;
        if the RIGHT-LEADER did not return during the last step then
            Mark-Link (Right); exit loop;
    State := HALT;
```

Algorithm 3: BHS-Ring-3

```
/* BHS in Unoriented Ring, using k ≥ 5 agents having 2 unmovable tokens
   each.  All agents have the same initial state START.                  */
START:
    Decide to place a token at homebase; State := CHECK-LEFT;
CHECK-LEFT:
    Move Left until the next node that contains a token; State := CHECK-RIGHT;
CHECK-RIGHT:
    Move Right until the next node that contains a token;
    Move Right again until the next node that contains some token;
    State := GO-BACK;
GO-BACK:                                   /* Go home to put the second token */
    Move Left until the next node that contains one token;
    Decide to place the second token;
    if there is a LEFT-LEADER agent then State := SEARCHER;
    ;
    else if there is a ALONE agent or a WAITING agent then
        State := LEFT-FOLLOWER;
    ;
    else if there is a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    ;
    else State := ALONE;
    ;
WAITING:                                          /* Wait for a partner agent */
    Wait until other agents arrive at the current node;
    if there is a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    else if there is a GO-BACK or ALONE agent a and no LEFT-LEADER agent
    then
        if there is another WAITING agent having same orientation as agent a then
            State := SEARCHER;
        else
            State := LEFT-LEADER;
    ;
    else State := SEARCHER;
    ;
```

Algorithm 3: BHS-Ring-3 (continued)

```
ALONE:                                   /* Move alone to the node with single token */
    Move Left until a node that contains either only one token or two tokens and a
    GO-BACK agent ;
    if there are no other agents then State := WAITING;
    ;
    if  there is a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    else  if there is a LEFT-LEADER agent then State := SEARCHER;
    ;
    ;
    else  if there is a WAITING agent and no GO-BACK agent then
        if  there is another ALONE agent having same orientation as the WAITING
        agent then
            State := SEARCHER;
        else
            State := LEFT-FOLLOWER;
    ;
    else  if there is a GO-BACK agent and no WAITING agent then
        if  there is another ALONE agent having same orientation as the GO-BACK
        agent then
            State := SEARCHER;
        else
            State := LEFT-LEADER;
    ;
    else  State := SEARCHER;
    ;
LEFT-LEADER:
    while true do PairedWalk (1) in Left direction;
    ;
LEFT-FOLLOWER:
    Align orientation with the LEFT-LEADER agent;
    while true do
        PairedWalk (0) in Left direction;
        if the LEFT-LEADER did not return during the last step then
            Mark-Link (Left); Exit Loop;
    State := RIGHT-LEADER;
CAUTIOUS-SEARCHER:     /* Find a RIGHT-LEADER in order to form a pair */
    repeat
        Move Right;
        if there is a RIGHT-LEADER agent then break;
        ;
        Move Left; Move Right;
    until there is a RIGHT-LEADER agent or a node with one token;
    if there a RIGHT-LEADER agent then State := RIGHT-FOLLOWER;
    ;
    else State := HALT;
```

Algorithm 3: BHS-Ring-3 (Continued)

```
SEARCHER:                    /* Go to a gate node to become RIGHT-FOLLOWER */
    while not at a node with one token do Move Right;
    ;
    State := RIGHT-FOLLOWER;
RIGHT-LEADER:                /* Find a Follower and perform Paired Walk */
    while not at a node with one token do
        Move Right;
    if there is no other agent then
        number_of_homebases := 0;
        repeat
            Move Right; Wait (2);
            if the current node contains two tokens and number_of_homebases ≠ 3
            then number_of_homebases := number_of_homebases + 1;
            ;
        until the current node contains one token;
        if number_of_homebases ≤ 2 then
            Wait until other agents arrive at the current node;
        else
            if there is no other agent then
                repeat
                    Move Left;
                    if there is a CAUTIOUS-SEARCHER agent then break;
                    ;
                    Wait (2);
                until the current node contains one token or a
                CAUTIOUS-SEARCHER;
    if there is another agent then
        while true do PairedWalk (1) in Right direction;
        ;
    else
        State := CAUTIOUS-SEARCHER;
RIGHT-FOLLOWER:
    Wait until there is a RIGHT-LEADER agent;
    Align orientation with the RIGHT-LEADER agent;
    while true do
        PairedWalk (0) in Right direction;
        if the RIGHT-LEADER did not return during the last step then
            Mark-Link (Right); Exit loop;
    State := HALT;
```

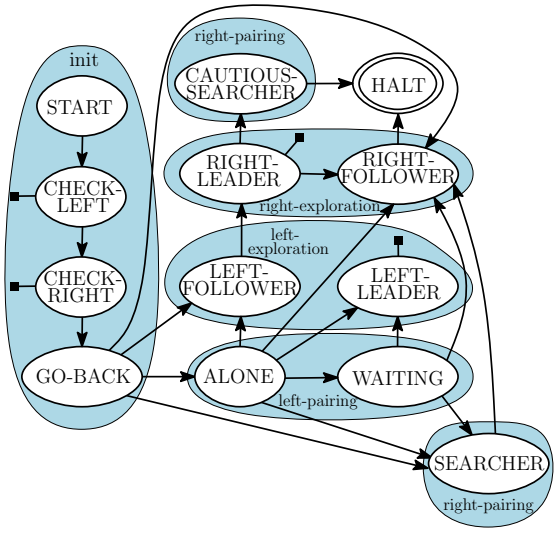

Figure 5: State transitions during the algorithm BHS-Ring-3 with $k \geq 5$ agents and 2 unmovable tokens per agent. Only agents in state CHECK-LEFT, CHECK-RIGHT, LEFT-LEADER, or RIGHT-LEADER may fall into the black hole.

agent (i.e., the port at the current node considered by the agent to be *Left*). Thus, when two agents meet at a node, one agent (e.g. the Follower) can orient itself according to the direction of the other agent (e.g. the Leader). As in the algorithm in oriented ring, the algorithm in unoriented ring executed by the agents essentially runs in five phases:

1. **Init phase:** During this phase, each agent places a token on its homebase (state START), moves left until the next homebase, i.e., the next node with a token (state CHECK-LEFT), returns to its homebase and moves right until the next homebase (state CHECK-RIGHT). Then the agent returns again to its homebase (state GO-BACK). The agents may not complete this phase of the algorithm at the same time. If the GO-BACK agent meets a single agent on returning to its homebase, the two agents form a pair (and start phase *Left-exploration*); otherwise if the GO-BACK agent meets a pair of agents, it becomes a SEARCHER agent (and starts phase *Right-pairing*). During this phase, exactly two agents will fall into the black hole and there will be two homebases with a single token (we call these node the *gate* nodes) and all the other homebases will eventually contain exactly two tokens each.

2. **Left-pairing:** During this phase, agents try to meet at the gate nodes by moving to the left until they reach a node containing a single token (state ALONE). The first agent reaching node with one token waits for a partner agent (state WAITING). When another agent arrives at the node,

they form a pair and proceed to the next phase. A pair can be formed either by :

- a WAITING agent and an ALONE agent,
- or a WAITING agent and GO-BACK agent,
- or an ALONE agent and a GO-BACK agent.

In all the above cases, the agents forming a pair have different states and they are assigned the roles of LEFT-LEADER and LEFT-FOLLOWER respectively. The role of LEFT-LEADER is assigned to the WAITING agent of the pair if there is one, or to the ALONE agent otherwise. The algorithm also has some additional rules to ensure that no two LEFT-LEADERs are created at the same node at the same time: no agent becomes a LEFT-LEADER if there is already another LEFT-LEADER at the same node (In this case, the agent become a SEARCHER). Another rule is that if a ALONE agent and a GO-BACK agent arrive at the same time to a node with a WAITING agent, then only the WAITING and GO-BACK agents form a pair (LEFT-LEADER, LEFT-FOLLOWER) while the ALONE agent changes to state SEARCHER (and starts phase Right-pairing).

Unlike in the oriented ring, it may happen that two ALONE agents arrive at the same time from opposite directions to a node u with one token. In this case, both agents wait (in state WAITING) until another agent arrives. Note that in this case, the ring is safe in both directions until the next homebase and thus, an agent b (whose homebase is u) would arrive within a finite time. When agent b arrives, only one of the WAITING agents (the one having the same orientation as b) changes to state LEFT-LEADER and pairs-up with agent b. The other agent changes to state SEARCHER (phase Right-pairing). A similar case occurs when an agent a is waiting and two agents (both in state ALONE) arrive from different directions. Among these two agents, the one having the same orientation as agent a pairs up with agent a and starts the *Paired Walk* procedure as LEFT-LEADER. The other agent changes to state SEARCHER (to start phase Right-pairing).

3. **Left-exploration:** During this phase, the pair formed during the previous phase performs *Paired Walk* in the left direction. One of the agents of a pair (state LEFT-LEADER) eventually falls into the black hole and the other agent (state LEFT-FOLLOWER) marks the edge leading to the black hole.

   As before there can be multiple leader-follower pairs performing *Paired Walk* in different parts of the ring. However the rules of the algorithm ensure that no two LEFT-LEADER agents may be created at the same node at the same time. Thus, two LEFT-LEADER agents cannot enter into the black hole at the same time from the same direction.

4. **Right-pairing:** When a LEFT-LEADER agent falls into the black hole, the corresponding LEFT-FOLLOWER agent, say agent r, becomes a RIGHT-LEADER. The RIGHT-LEADER agent r moves in the other direction (i.e. "right") until it reaches a gate node u containing one token. If the RIGHT-LEADER does not find a RIGHT-FOLLOWER at u to pair with, the RIGHT-LEADER r performs a *slow* walk in the same direction until it reaches a node v with one token. During the slow walk, an agent moves at one-third the speed of any other agent (i.e., waits two steps after each move). Agent r also counts (up to a maximum of 3) the number of nodes it sees that have two tokens (i.e. the number of homebases). If the agent r encounters no more than two homebases (nodes with two tokens) between u and v, then it knows that v is not a gate node and thus, the agent whose homebase is node v, will eventually come back to this node. In this case, agent r waits and form a pair (RIGHT-LEADER, RIGHT-FOLLOWER) with the first agent that arrives. In the other case, when agent r has seen at least 3 homebases, it is not possible to determine if v is a gate node or not, so agent r cannot wait. If there is no other agent already at v to form a pair with, the agent r switches direction and moves back towards u performing a slow walk. This ensures that it will meet and form a pair with any *unpaired* agent, i.e., any agent that did not perform a *Paired Walk* in the previous phase. Recall that an agent that did not pair up during the previous phases of the algorithm, became a SEARCHER. Such a SEARCHER agent moves left until reaching the gate node and waits for a RIGHT-LEADER in order to form a pair (state RIGHT-FOLLOWER).

   If there is no such unpaired agent, then agent r may not find any agent on reaching node u. In that case, agent r becomes a CAUTIOUS-SEARCHER, switches direction again and moves back to v by repeating the following sequence of three moves: move right, move left, move right . This ensures that agent r will meet and form a pair with any RIGHT-LEADER agent that has the same orientation as agent r and is looking for a partner. Finally, if agent r still did not meet any other agent during this phase, we can show that there must be a pair of agents with the opposite orientation (as agent r). Such a pair would have already detected and marked the other link leading to the black hole. Thus, agent r terminates.

5. **Right-exploration:** During this phase, the pair formed during the previous phase performs *Paired Walk* in the right direction. One of the agents of a pair (state RIGHT-LEADER) eventually falls into the black hole and the other agent (state RIGHT-FOLLOWER) marks the edge leading to the black hole.

A formal description of the algorithm can be found in Algorithm 3 and the state transitions during the algorithm are shown in Figure 5.

Lemma 5.2. *During the algorithm BHS-Ring-3, the following holds, assuming there are at least* 5 *agents, each carrying two unmovable tokens*

(i) Exactly two agents fall into the black hole before placing their second token.

(ii) An agent that is not in state LEFT-LEADER or RIGHT-LEADER never enters the black hole after placing its second token.

(iii) There is at least one LEFT-LEADER and each LEFT-LEADER has a corresponding LEFT-FOLLOWER.

(iv) At least one Paired Walk is performed in each direction, marking the corresponding edge incident to the black hole as dangerous.

Proof : (i) This fact follows directly from the description of the algorithm. Only the two agents whose homebases are closest to the black hole (from either side) would fall into the black hole before placing the second token.
(ii) After placing its second token on the homebase, an agent is not allowed to move beyond the gate nodes, unless it is performing a *Paired Walk* . If an agent enters the black hole while performing *Paired Walk* then it must be in state LEFT-LEADER or RIGHT-LEADER.
(iii) There are at least three agents which do not enter the black hole before placing their second token. At least two of these must have the same orientation. Among those surviving agents having same orientation, at least two would eventually meet at a node containing a single token (note that the agents are allowed to wait only at nodes with a single token and eventually only the *gate* nodes will contain a single token). Whenever multiple agents meet at a node, no two of them have the same state and the same orientation. According to the rules of the algorithm exactly one of them becomes LEFT-LEADER and exactly one of them becomes LEFT-FOLLOWER.
(iv) Due to Property (iii) above, we know that at least one pair (LEFT-LEADER, LEFT-FOLLOWER) performed a *Paired Walk* and has marked one link incident to the black hole. Let us consider the first LEFT-LEADER agent l fell into the black hole, and the corresponding LEFT-FOLLOWER agent r that became a RIGHT-LEADER. Agent r goes back to the gate node u and then goes right until it finds a node v with one token. If while going right, this agent saw no more than two homebases with two tokens (i.e. $number\_of\_homebases \leq 2$ in Algorithm 3), then v is not a gate node since there must be at least three homebases that are not gate nodes. In this case, the agent whose homebase is v will eventually come back and form a pair with the agent r at node v. This pair of agents will perform *Paired Walk* in the other direction.

Now, assume that agent r has seen three or more homebases with two tokens between u and v. Thus, there are one or more agents denoted by $a_1, a_2, \ldots$ that are distinct from l and r and whose homebases are between u and v. Now we consider multiple scenarios depending on the orientation of agents a_i and how they have paired.

First, we assume that agent a_1 is unpaired. In this case, agent a_1 will go to either u or v and wait there. Since agent a_1 has already put its second token in its homebase before agent r reaches that node, agent r will find a_1 waiting either at node v or at u, when agent r returns. This waiting agent becomes a RIGHT-FOLLOWER and joins the *Paired Walk* procedure with agent r. Hence, the second link to the black hole would be discovered by this pair of agents.

The second scenario to consider is when a LEFT-LEADER agent a_1 has paired with a LEFT-FOLLOWER agent a_2. If this pair has a different orientation from the pair (l, r) then there was a *Paired Walk* in each direction and the lemma holds. Hence, we can assume this pair (a_1, a_2) has the same orientation as the pair (l, r). Since no two LEFT-LEADERs are created at the same node at the same time, l and a_1 enter the black hole at different moments in time. By definition (l, r) is the first pair to reach the black hole. When agent r becomes a CAUTIOUS-SEARCHER, agent a_2 must have already come back to the gate node u in state RIGHT-LEADER. Indeed, in order to become a CAUTIOUS-SEARCHER agent r must have moved from u to the black hole w during the *Paired Walk* ($3d(w, u)$ steps), returned back to u ($d(w, u)$ steps), moved to v in a slow walk ($3d(u, v)$ steps), and moved back to u in another slow walk ($3d(u, v)$ steps), for a total of at least $6d(u, v) + 4d(w, u)$ steps. After at most $5d(u, v)$ steps from the time the (l, r) pair was formed, agents a_1 and a_2 must have paired and reached the gate node u. Hence after a total of $6d(u, v) + 4d(w, u)$ steps, agent a_2 must be already back at the gate node u. Thus, when agent r becomes a CAUTIOUS-SEARCHER, agent a_2 is either moving from u to v or the other way. In the former case, agents r and a_2 move at the same speed to the right until agent a_2 reaches a node with one token. Agent a_2 will eventually turn back and the agents will move in opposite directions. Assume by contradiction, that agents r and a_2 did not meet while moving in opposite directions. It means that they have passed each other on an edge (t, s). If this happens when agent r was executing the first movement to the right of its loop from node t to node s then they meet two steps later since r will come back to t while a_2 is waiting. This cannot happen when r was executing the second movement to the right of its loop from t to s since r would have met a_2 waiting at s two steps before. Thus, the two agents will meet in all cases. The CAUTIOUS-SEARCHER agent becomes RIGHT-FOLLOWER and joins the *Paired Walk* procedure with the RIGHT-LEADER. Hence, the second link to the black hole would be discovered by this pair of agents. □

Theorem 5.2. *Algorithm BHS-Ring-3 correctly solves the black hole search problem in an unoriented ring with 5 or more agents having constant memory and carrying two unmovable tokens each.*

Proof : Due to the above lemma, we know that only a leader agent can fall into the black hole after putting its two tokens. For each leader falling into the black hole, there is a follower agent that survives. Hence, at least one agent will never fall into the black hole. Both links to the black hole are actually discovered and marked as dangerous by property (iv) of the lemma. The pseudo-code of

Algorithm 3 can be implemented with a finite state automaton since the only variable ($number\_of\_homebases$) can only take four different values (0, 1, 2 or 3). □

6. Conclusions

The results of this paper determine the minimum resources necessary for locating a black hole in synchronous ring networks. We presented algorithms that use the optimal number of agents and the optimal number of tokens per agent, while requiring only constant-size memory. Thus, all resources used by our algorithms are independent of the size of the network. Notice that all the algorithms presented in the paper have a time complexity of $O(n)$ steps, so, they are asymptotically optimal for BHS in a ring.

The model introduced in this paper differs from that of the previous studies, since we consider constant memory agents independent of the size of the network. Another difference is that the agents were equipped with the capability of marking as dangerous the links to the black-hole once they have been detected. The purpose of giving the agents the ability to mark as dangerous the links which lead to the black-hole is the following: Since the agents have only constant memory, upon completion of any protocol which solves the problem, one agent cannot report both incident links to the black-hole. Hence those links should be only clearly indicated, somehow. It is true that when the agents are equipped with one movable token each, the final configuration of tokens constructed by Algorithm 1 clearly indicates where the black-hole is. However for the cases with unmovable tokens, it is unclear whether there exist algorithms which use the same minimum number of tokens and agents as our algorithms, and at the same time manage to construct configurations with tokens to clearly indicate the location of each link to the black-hole. We conjecture that such algorithms do not exist. Nevertheless, in this work we were interested in determining the minimal resources necessary and sufficient for *discovering* both links incident to the black hole. Hence in our algorithms, the agents can mark the dangerous links only for the purpose of reporting the location of the black hole. We emphasize here that the agents cannot see such marked links and thus, the execution of the algorithm is not affected by assuming this additional capability.

The main question answered by the paper is how the limitation on the memory of the agents influences the resources required for solving BHS. We show that the constant memory limitation has no influence on the resource requirements since the (matching) lower bounds hold even if the agents have unlimited memory. It would be interesting to investigate if similar tight results hold for BHS in other network topologies. We would also like to investigate the difference between 'pure' and 'enhanced' token model in terms of the minimum resources necessary for black hole search in higher degree networks.

An interesting extension to our model is the following: Agents do not start at the same time but take each step synchronously. In other words, what happens when an adversary introduces arbitrary (but finite) delays before the agents start executing a protocol. Although our algorithms rely on the fact that the

agents start at the same time, we believe that our algorithms could be modified to work also for the case of agents starting with delays. However this requires taking care of several technicalities, mainly due to the fact that our algorithms have been designed to work for any number of agents (as long as this number of agents is enough for solving the problem) but also due to lack of a mutual exclusion mechanism for reading and writing at nodes. We thus leave as open the question of how much resources are necessary to solve the BHS problem with constant memory in the presence of delays.

Reference

[1] B. Balamohan, 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.

[2] B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Improving the optimal bounds for black hole search in rings. In *SIROCCO 2011*, pages 198–209, 2011.

[3] B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Time optimal algorithms for black hole search in rings. *Discrete Mathematics, Algorithms and Applications*, 3(4):457–471, 2011.

[4] M. A. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 75–85, 1994.

[5] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for scattered black hole search in a ring. In *Proceedings of 18th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 6796 of *LNCS*, pages 186–197, 2011.

[6] J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. In *Proceedings of 21st International Conference on Distributed Computing*, pages 108–122, 2007.

[7] C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *Proceedings of 10th International Conference on Principles of Distributed Systems*, LNCS 4305, pages 320–332, 2006.

[8] C. Cooper, R. Klasing, and T. Radzik. Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, 411(14-15):1638–1647, 2010.

[9] J. Czyzowicz, S. Dobrev, R. Kralovic, S. Miklik, and D. Pardubska. Black hole search in directed graphs. In *Proceedings of 16th International Colloquium on Structural Information and Communication Complexity*, pages 182–194, 2009.

[10] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2,3):229–242, 2006.

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

[12] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.

[13] S. Dobrev, P. Flocchini, R. Kralovic, G. Prencipe, P. Ruzicka, and N. Santoro. Optimal search for a black hole in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[14] S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring a dangerous unknown graph using tokens. In *Proceedings of 5th IFIP International Conference on Theoretical Computer Science*, pages 131–150, 2006.

[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):1–19, 2006.

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

[17] S. Dobrev, P. Flocchini, and N. Santoro. Improved bounds for optimal black hole search in a network with a map. In *Proceedings of 10th International Colloquium on Structural Information and Communication Complexity*, pages 111–122, 2004.

[18] S. Dobrev, R. Kralovic, N. Santoro, and W. Shi. Black hole search in asynchronous rings using tokens. In *Proceedings of 6th Conference on Algorithms and Complexity*, pages 139–150, 2006.

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

[20] S. Dobrev, N. Santoro, and W. Shi. Using scattered mobile agents to locate a black hole in an un-oriented ring with tokens. *International Journal of Foundations of Computer Science*, 19(6):1355–1372, 2008.

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

[22] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48:166–177, 2006.

[23] P. Glaus. Locating a black hole without the knowledge of incoming link. In *Proceedings of 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 128–138, 2009.

[24] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and approximation results for black hole search in arbitrary graphs. *Theoretical Computer Science*, 384(2-3):201–221, 2007.

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

[26] A. Kosowski, A. Navarra, and C. Pinotti. Synchronization helps robots to detect black holes in directed graphs. In *Proceedings of 13th International Conference on Principles of Distributed Systems*, pages 86–98, 2009.

[27] R. Kràlovic and S. Miklìk. Periodic data retrieval problem in rings containing a malicious host. In *Proceedings of 17th International Colloquium on Structural Information and Communication Complexity*, pages 156–167, 2010.

[28] C. E. Shannon. Presentation of a maze-solving machine. In *Proceedings of 8th Conference of the Josiah Macy Jr. Found. (Cybernetics)*, pages 173–180, 1951.

[29] W. Shi. Black hole search with tokens in interconnected networks. In *Proceedings of 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 670–682, 2009.