# Searching for an Eventually-Emerging Black Hole in Rings

François Bonnet, Quentin Bramas, Anissa Lamani

# Searching for an Eventually-Emerging Black Hole in Rings

François Bonnet[1], Quentin Bramas[2], Anissa Lamani[2]

[1] Institute of Science Tokyo, Japan
[2] University of Strasbourg, CNRS, ICUBE, France

**Abstract.** We study a novel variant of the Black Hole Search (BHS) problem where the black hole, a node that silently destroys visiting agents, can appear at any time during execution, rather than being present initially, as is assumed in all previous work. Our focus is on ring networks, and we examine this variant of the BHS problem under various assumptions, including whether the ring size is known and whether agents can use pebbles for marking nodes.
For synchronous agents, we provide four solutions: (1) a 4-agent algorithm for rings without additional assumptions, (2) a 3-agent algorithm assuming known ring size, (3) a 3-agent algorithm using pebbles, and (4) a 3-agent solution without additional assumptions but having a quadratic time complexity. For asynchronous agents, we develop two algorithms: one using $n$ agents without additional assumptions, and another using only 4 agents with pebbles.

**Keywords:** Black hole search, mobile agents, distributed algorithms

*Number of pages: 12 excluding cover page and references. Font size: 11pt*

*A preliminary version of this work appears as a short paper in the 27th International Symposium on Stabilization, Safety, and Security of Distributed Systems, 2025*

## 1 Introduction

The Black Hole Search (BHS) problem is a well-studied theoretical problem in distributed computing and algorithm design. It focuses on finding a dangerous node (black hole) in a network where any agent entering the node gets destroyed, and no information is returned. This problem has practical implications for real-world scenarios such as intrusion detection, where the black hole represents a compromised node or malicious activity that disrupts normal operations without immediate detection.

This paper investigates the Black Hole Search problem within a ring topology, but we assume that the black hole **can appear at any time during the execution**, so it is not necessarily present initially, as is commonly assumed in previous work. We explore various strategies for detecting the black hole using multiple agents under different conditions, such as synchronous and asynchronous communication models, known and unknown network sizes, and with or without external marking mechanisms like pebbles. The study aims to provide a comprehensive understanding of the complexities involved and proposes asymptotically optimal solutions for potential applications in intrusion detection systems.

**Related Work** Probably the first paper introducing the Black Hole Search problem was [16] followed by several papers extending the problem to other topologies and settings [2,7,6,12,10,11,19,13,14,18,35]. In these papers, different conditions have been considered, such as prior knowledge of the network size, the synchrony level of the agents (synchronous or asynchronous), communication mechanisms (e.g., whiteboard/pebbles) [15], and whether the agents are initially scattered [24] or start from a single node called a base station. In these previous works, the main goal is to first identify the conditions that allow a team of agents to successfully solve the BHS problem, and then propose distributed solutions that are optimal in terms of the number of mobile agents required and the move complexity [30,31,1,8,32]

Variants of the BHS problem include having multiple black holes [9], gray/Byzantine holes [36,29], and rendezvous in dangerous networks [18,22]. Byzantine black holes [29] can be seen as a generalization of eventually-emerging black holes as they may decide not to kill an agent passing by and can also erase the content of the whiteboard where they are located. In [29], the exploration problem is considered assuming that the starting position of the agents remains safe throughout the execution. The algorithms and corresponding bounds obtained are different from what we obtain in our setting.

Exploration with fault tolerance [28,27,25] and decontamination of mobile threats [5] extend BHS to broader fault models.

Recent investigations have extended the study of the problem to dynamic graphs [26,4,3], primarily focusing on ring networks with 1-interval connectivity (e.g., [34,33]). The goal is to determine how the computational complexity of finding a black hole changes when the graph is dynamic. The problem was again studied under different assumptions.

To the best of our knowledge, all previous investigations of the Black Hole Search problem assume that the black hole is present in the initial configuration. In this paper, we address the case in which the black hole can emerge at any time during execution, preventing solutions from relying on strategies based only on cautious walks (where agents move in such a way that two or more agents never enter the black hole at the same time from the same edge [20]). Indeed, during execution, all agents cannot be on the same node; otherwise, if the black hole appears on the node hosting the agents, no agent would be left to deduce the location of the black hole. We explore the impact of such an eventual appearance on ring networks under different assumptions, such as whether the size of the ring is known and whether agents can use pebbles.

**Contributions** We address the aforementioned problem on ring networks and present four black hole search algorithms for synchronous agents, each one based on a particular set of assumptions. The first one uses four agents; the second one assumes that the agents know the size of the ring; the third one considers that the agents can use a pebble; and the fourth one has no additional assumptions but has a higher time complexity. When the agents are asynchronous, we provide two algorithms: one that uses $n$ agents without any assumptions, where $n$ denotes the size of the ring, and a second that uses only 4 agents with pebbles.

Table 1: Summary of our results (we ignored constant terms in the complexity). Underlined values are not optimal.

| **Algorithm** | **# agents** | **pebble** | **knowledge $n$** | **safe HB** | **complexity** | **termination** |
|---|---|---|---|---|---|---|
| Synchronous agents | | | | | | |
| Imp. Thm. 1 | 2 | Yes | Yes | Yes | $*$ | implicit |
| $\mathcal{A}_1$ | 4 | No | No | No | $3n$ | explicit |
| $\mathcal{A}_2$ | 3 | Yes | No | No | $4n$ | explicit |
| $\mathcal{A}_3$ | 3 | No | Yes | No | $4n$ | explicit |
| $\mathcal{A}_4$ | 3 | No | No | No | $O(n^2)$ | explicit |
| Asynchronous agents | | | | | | |
| Imp. Thm. 6 | * | Yes | Yes | Yes | $*$ | implicit |
| $\mathcal{A}_5$ | n | No | No | No | $2n$ | implicit |
| $\mathcal{A}_6$ | 4 | Yes | No | No | $5n$ | implicit |

Table 1 summarizes our results. All our algorithms can be visualized in an interactive simulator to see the executions with a given black hole appearance round and location[3]. The pseudo-code we provide matches the implementation of the online simulation, and one can see the current state of each agent, including the line number of its last instruction in the algorithm.

Due to space limitations, several proofs and algorithm pseudo-codes have been omitted and are available in the appendix.

## 2 Model

We consider a team $\mathcal{A}$ of $k$ agents located on a ring of size $n \geq 2$. The ring consists of $n$ nodes of degree 2, denoted $u_0, u_1, \ldots, u_{n-1}$ where $u_i$ and $u_j$ are connected only if $|j-i| = 1 \mod n$. The nodes are anonymous for the agents but the ring is consistently oriented[4] *i.e.*, all the agents agree on what is the neighbor in the clockwise orientation and what is the neighbor in the counterclockwise orientation. All the agents start at the same node called *the home base (HB)*. We consider that the agents have unbounded memory, in particular, they have access to an internal clock (which may not be synchronized in the asynchronous setting).

Each agent has a unique identifier in the interval $[1, k]$ and executes an algorithm that takes as input the current state of the agent (an arbitrary number of variables used by the algorithm and in particular its internal clock), as well as the states of any collocated agents (located on the same node). In some cases, the agents are allowed to use pebbles which can be dropped on a node, picked up, and carried by the agents. We assume that all the pebbles are identical and hence, there is no way to identify which agent dropped a pebble on a given node.

**Scheduler** We consider two kinds of schedulers. Under the *synchronous* scheduler, the internal clocks of all the agents are synchronized. In other words, time is discretized into rounds and at each round, all agents execute their algorithms synchronously. In this setting, it is meaningful for an agent to "wait" for another agent at a given node. Then, all agents that have decided to move do so simultaneously.

Under the *asynchronous* scheduler, the internal clocks of the agents have arbitrary speeds. Nevertheless, we still consider that the reading of the states of collocated agents is performed atomically, *i.e.*, an agent retrieves the states of any collocated agents at a given point in time in order to execute its algorithm. Then, the execution takes an arbitrary (but finite) amount of time. Finally, if the agent decides to move, the movement also takes an arbitrary amount of time (during which the agent is "invisible" on the edge). We make no assumption on how agents move along an edge, so we even allow an agent to "overtake" another agent (the link is not necessarily FIFO). In this setting, we can also discretize time into a sequence of

[3] `https://bramas.fr/static/emerging-BHS/`. Asynchronous executions are simulated by adding random idle rounds

[4] In our setting, this is equivalent to assume that the agents know the input port.

time instants representing the moments when events occur (an agent reads the states of collocated agents, leaves a node, or arrives at a node). *An asynchronous round* is the time required for all agents to perform at least one complete cycle (from reading states to arriving at the destination node, if applicable).

**Eventually-emerging black hole** A *black hole* is a node that destroys every agent located on it. When an agent is destroyed, it disappears from the system. We say that a black hole appears in node $u$ at time $t$ if, starting from time $t$, node $u$ becomes a black hole. Agents that are not on $u$ are not impacted; agents in node $u$ at time $t-1$ that do not leave $u$, or those arriving on $u$ at time $t' \geq t$, are killed. We denote by $\mathcal{A}_t$ the set of agents that have not been killed by time $t$.

We say a black hole is initial if it is present in the initial configuration, at time $t = 0$. In this paper, we consider an eventually-emerging black hole, *i.e.*, there exists a time $t \geq 0$ and a node $u$ such that a black hole appears on $u$ at time $t$. We assume that the black hole cannot appear at the home base at time $t = 0$ (otherwise all agents would be killed before starting their execution). However, we consider, unless stated otherwise, that a black hole can appear at the home base at any time $t \geq 1$ in the synchronous case, and at any time after one asynchronous round in the asynchronous case. That is, we assume the home base is not necessarily safe (it is only guaranteed to be safe in the first round).

In the remainder, we consider that a single black hole eventually appears and remains for the rest of the execution. We consider that each agent $a$ has a variable $\texttt{bh}_a$, which contains either $\perp$ or the relative location of a node. It represents the node where the agent thinks a black hole is located, encoded as a distance and direction relative to its own position. $\texttt{bh}_a(t)$ denotes the value of the variable at time $t$. In the online simulation, a negative value means the distance is in the counterclockwise direction. For simplicity, in the remainder, $\texttt{bh}_a(t)$ represents the node (in $V$) suspected by the agent.

**The Eventually-emerging black hole search (EBHS) problem** We say an algorithm solves the EBHS problem with explicit termination if, in any execution with an eventually-emerging black hole, eventually there exists an agent that identifies correctly the location of the black hole and terminates. In other words, we require three properties: an agent must only identify a node where a black hole is located; if an agent $a$ identifies the location of the black hole by setting the value $\texttt{bh}_a$, then it cannot change the value of $\texttt{bh}_a$ afterward; and eventually one agent correctly identifies the location of the black hole.

Formally, **an algorithm solves the EBHS problem with explicit termination** if, in any execution $E$ where a black hole appears on $u$, the three conditions are satisfied:

$$\forall t \geq 0, \forall a \in \mathcal{A}_t, \texttt{bh}_a(t) \in \{\perp, u\}$$
$$\forall t \geq 0, \forall a \in \mathcal{A}_t \cap \mathcal{A}_{t+1}, (\texttt{bh}_a(t) = u \Rightarrow \texttt{bh}_a(t+1) = u)$$
$$\exists t \geq 0, \exists a \in \mathcal{A}_t, \text{ s.t. } \texttt{bh}_a(t) = u$$

As we show in Theorem 6, EBHS with explicit termination is not solvable with asynchronous agents, so we define a new version of the problem where termination is implicit. We say an algorithm solves the EBHS problem with implicit termination if, in any execution with an eventually-emerging black hole, eventually all the agents that suspect a node, correctly suspect the location of the black hole for infinity (and at least one agent suspects the location of the black hole).

Formally, **an algorithm solves the EBHS problem with implicit termination** if, in any execution $E$ where a black hole appears on $u$, the two conditions are satisfied:

$$\forall a \in \bigcap_{t \in \mathbb{N}} \mathcal{A}_t, \exists t \geq 0, \text{ such that } (\forall t' > t, \texttt{bh}_a(t') = u) \vee (\forall t' > t, \texttt{bh}_a(t') = \perp)$$
$$\exists a \in \bigcap_{t \in \mathbb{N}} \mathcal{A}_t, \exists t \geq 0, \text{ such that } (\forall t' > t, \texttt{bh}_a(t') = u)$$

In the literature, finding the location of the black hole, *i.e.*, identifying a path to the black hole, is sometimes referred to as the weak black hole search problem, and marking all the edges leading to the

black hole is the strong version of the problem. Here, we do not consider that edges can be marked and we solve the weak version. Also, one can notice that when $n$ is known, the two versions are equivalent.

**Complexity** Since the black hole can appear after an arbitrarily long time, the time complexity is always computed starting from the time at which the BH appears. In other words, the time complexity (or simply complexity) of an algorithm is the number of (synchronous/asynchronous) rounds between the appearance of the black hole and the time $t$ from which an agent correctly identifies the location of the black hole. When considering implicit termination, we consider the number of asynchronous rounds until all the agents suspecting a black hole suspect the correct location and never change their suspicion after that.

## 3 Synchronous agents

In this section we address the EBHS problem assuming synchronous agents. We present four deterministic algorithms that solve the EBHS problem under different hypotheses. The various algorithms highlight the fact that it is very challenging to find an algorithm that is optimal for all the assumptions (number of agents, knowledge of $n$, storage capability, and time complexity). The four algorithms use different techniques to be optimal regarding several assumptions, but each of them requires one possibly non-optimal assumption (the underlined values in Table 1). Our four algorithms are different and use new techniques that are not found in the literature. While they may seem simple at first glance, they are in fact the result of a very careful construction to make them correct. It is still open whether there exists an algorithm that is optimal regarding all the assumptions.

The first algorithm uses four agents that are not aware of the size of the ring and do not have any additional external mechanisms to communicate (like pebbles or whiteboards). In the second and third algorithms, the number of agents is reduced to three, assuming the size of the ring $n$ is known or assuming the agents can use pebbles, respectively. The fourth algorithm uses three agents without the knowledge of $n$ and without using pebbles, but its complexity is quadratic. A summary of our results for synchronous agents is presented in Table 1.

**Handling small rings ($n \in \{2, 3, 4\}$)** To simplify the presentation of Algorithms $\mathcal{A}_1$–$\mathcal{A}_4$, we assume throughout the remainder of this section that the ring size satisfies $n \geq 5$. When we assume $n \geq 2$, the problem can be handled by a short pre-processing phase executed before running any of our four algorithms.

Algorithm 3 is such a pre-processing algorithm. It uses only three agents (agents $a_1, a_2, a_3$); a fourth agent, if present, is ignored (it simply waits). First, the agents detect the case $n = 2$ in a constant number of rounds. Otherwise, they start executing $\mathcal{A}_4$ (which is correct already for $n \geq 3$) while progressively ruling out the remaining small sizes. Importantly, this pre-processing phase can be stopped as soon as the agents have certified that $n \geq 5$; the algorithm is designed so that in this case all agents stop in the same round, and we can then start any of $\mathcal{A}_1$–$\mathcal{A}_4$ from a clean synchronized configuration.

The pseudo-code is given in Algorithm 3 in the appendix.

**Cautious walk** In some of our algorithms, we use the cautious walk strategy that was introduced in [17,20]. The cautious walk is a strategy used when the agents are sure about the existence of the black hole. It ensures that no two or more agents enter the black hole from the same edge. This is done in the following manner: assume that two agents are on node $u$ and one moves from $u$ to a neighboring node $v$. If the agent comes back to $u$, then $v$ is identified as safe and both agents can move toward it; otherwise, it is identified as a black hole.

**Procedure 1:** CautiousWalk($leader, follower, direction$)

```
1  if agent is leader then
2      while true do
3          MOVE towards direction
4          MOVE towards the opposite of direction
5          MOVE towards direction
6  if agent is follower then
7      while true do
8          WAIT 1 round
9          WAIT 1 round
10         if not collocated with leader then
11             bh ← node at direction
12             TERMINATE
13         MOVE towards direction
```

Of course, a cautious walk works if we know that the black hole has already emerged; otherwise, both agents can be killed when they are collocated.

Procedure 1 presents the algorithm pseudo-code, where *agent* refers to the current agent executing the algorithm. This procedure can be executed given three arguments: the two agents performing the cautious walk and the direction. The red part of the algorithm shows the code executed when the presence of the black hole is detected. In the cautious walk, the variable `bh` is set and the agent terminates.

**Impossibility with two agents** We can prove that the EBHS problem is not solvable with two agents, as is the case when the black hole exists initially. This result holds even with whiteboards, safe home base, and the knowledge of $n$. Interestingly, it also holds assuming implicit termination. Due to space constraints, the proof is omitted but can be found in the appendix.

**Theorem 1.** *There is no algorithm solving the EBHS problem with only two synchronized agents, even with a safe home base, whiteboards, the knowledge of $n$, and implicit termination.*

### 3.1 Algorithm $\mathcal{A}_1$: Four Agents

We present in the following a deterministic algorithm $\mathcal{A}_1$ that solves the EBHS problem using four agents using only face-to-face communications. The size of the ring $n$ is unknown and the HB is assumed unsafe. This first algorithm consists of running an algorithm that works when the black hole is initially present, here simply the cautious walk. The idea is to create two groups of agents running such an algorithm. Since the two groups never meet, the appearance of the black hole can only impact one of the two groups. Hence, the correctness of the algorithm follows from the correctness of the cautious walk strategy. The complete algorithm is given in Algorithm 1.

**Algorithm 1:** $\mathcal{A}_1$: Four-Agent EBS Solution

```
   Input:4 agents a1, a2, a3, a4
1  if agent is in {a1, a2} then
2  |   MOVE clockwise;
3  |   execute CautiousWalk(a1, a2, clockwise);
4  else
5  |   MOVE counterclockwise;
6  |   execute CautiousWalk(a3, a4, clockwise);
```

Observe that when the black hole appears, the worst time complexity is the same as for the cautious walk with initial black hole, which is $3n$ rounds. We hence have the following theorem.

**Theorem 2.** *Assuming $n \geq 5$ (otherwise, run Algorithm $\mathcal{A}_0$ first), Algorithm $\mathcal{A}_1$ solves the weak EBHS problem with 4 synchronous agents, with a worst-case time complexity of $3n$.*

### 3.2 Algorithm $\mathcal{A}_2$: with one pebble

We now solve the problem with Algorithm $\mathcal{A}_2$ using three agents having a single pebble with a completely new algorithm. Initially, two agents move in one direction (say clockwise, right on the figures), and one

in the opposite direction. Then, if no black hole is detected, the agents performs three moves repeatedly. After the three moves, the configuration is similar but translated one node clockwise.

Figure 1 shows the first 5 configurations of the execution of Algorithm $\mathcal{A}_2$. As shown, configuration $C_4$ is just the translated version of configuration $C_1$ so the same sequence of moves is repeated until a black hole is detected.

A black hole is detected when an agent that is supposed to be collocated with another agent is alone. Then, this agent knows that a black hole appeared. The pseudo-code of the algorithm is given in Algorithm 2. The red parts correspond to the actions when a black hole is detected.

**Algorithm 2:** $\mathcal{A}_2$: Three-Agent EBS Solution with one pebble

```
   Input:3 agents a1, a2, a3, and one pebble
 1 if agent is a1 then
 2     MOVE counter-clockwise;
 3     while true do
 4         DROP-PEBBLE and MOVE clockwise;
 5         if not collocated with a2 then
 6             bh ← node in the clockwise direction;
 7             TERMINATE;
 8         MOVE counter-clockwise;
 9         GET-PEBBLE and MOVE clockwise;
10 else if agent is a2 then
11     MOVE clockwise;
12     while true do
13         MOVE counter-clockwise;
14         if not collocated with a1 then
15             MOVE clockwise;
16             detection ← true;
17             MOVE clockwise;
18             execute CautiousWalk(a2, a3, counter-clockwise);
19         MOVE clockwise;
20         MOVE clockwise;
21 else
22     MOVE clockwise;
23     while true do
24         if not collocated with a2 then
25             while there is no pebble do
26                 MOVE clockwise
27             bh ← node in the clockwise direction;
28             TERMINATE;
29         else if a2.detection = true then
30             execute CautiousWalk(a2, a3, counter-clockwise);
31         MOVE clockwise;
32         WAIT 2 rounds;
```

To illustrate how Algorithm $\mathcal{A}_2$ works, consider the case where the detection occurs in a configuration similar to $C_1$ by agent $a_3$ on $u_2$. The detecting agent does not know where the black hole is located but it knows it is either one or two hop(s) counterclockwise, so it moves in the clockwise direction until it sees a pebble. When it finds the pebble, it knows the black hole is adjacent to it.

If the detection occurs in a configuration similar to $C_2$ by agent $a_1$ (resp. $a_2$) located on $u_1$, then it knows the black hole is on the clockwise direction and both other agents are killed (resp. on the anticlockwise direction, and the agent continues its movement to tell agent $a_3$ about it and to start a cautious walk).

The following Theorem proves the correctness regardless of where the black hole appears.

**Theorem 3.** *Assuming $n \geq 5$ (otherwise, run Algorithm $\mathcal{A}_0$ first), Algorithm $\mathcal{A}_2$ solves the EBHS using three synchronous agents having a single pebble, with a worst-case time complexity of $4n$.*

*Proof.* Let $a_1$, $a_2$ and $a_3$ denote the three agents (when illustrated by a figure, we consider them ordered from left to right).

While no agent is killed by a black hole, the three agents moves such that every three rounds they are translated by one node, so when a black hole appears, at least one agent is eventually killed.

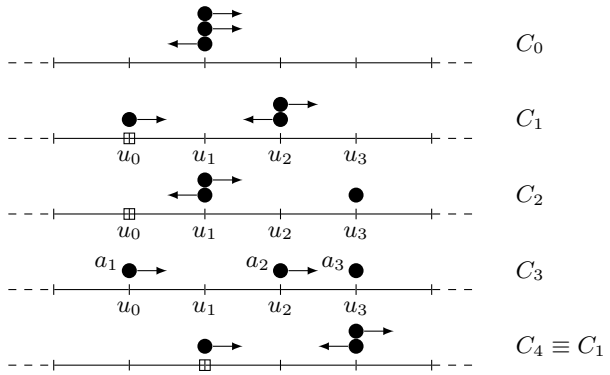


Fig. 1: First 5 configurations of the execution of Algorithm $\mathcal{A}_2$. The square added to a position represents the pebble.

Without loss of generality, we can assume that the first time an agent is destroyed is in configuration $C_1, C_2$, or $C_3$, in node $u_0, u_1, u_2$, or $u_3$ that contains at least an agent. We now show that in all the 7 possible cases, at least one agent remains and identify the black hole correctly.

- **If the black hole appears during $C_1$ on node $u_0$.** Then $a_1$ is killed and agent $a_2$ detects it in the next round, since she is not collocated with $a_1$. This corresponds to Line 14. $a_2$ then meets with $a_3$ and they start a cautious walk to eventually detect the black hole (after 2 iterations of the cautious walk).
- **If the black hole appears during $C_1$ on node $u_2$.** Then $a_2$ and $a_3$ are killed and the third agent $a_1$ detects it in the next configuration since no other agent is collocated with her (in $C_2$, $a_1$ should be collocated with $a_2$). This corresponds to Line 5 of the algorithm. $a_1$, located on $u_1$, correctly identifies that the black hole is located on its clockwise adjacent node.
- **If the black hole appears during $C_2$ on node $u_1$.** Then agents $a_1$ and $a_2$ are killed and the third agent $a_3$ detects it 2 rounds after, when no other agent is collocated with her (in $C_4$, $a_3$ should be collocated with $a_2$). This corresponds to Line 24. $a_3$ moves clockwise until she sees the pebble in $u_0$, and she knows that the black hole is adjacent to the pebble.
- **If the black hole appears during $C_2$ on node $u_3$.** Then agent $a_3$ is killed. Two rounds after, $a_2$ is also killed and we are in a configuration similar to $C_1$ where the black hole appears in $u_2$.
- **If the black hole appears during $C_3$ on node $u_0$.** Then $a_1$ is killed and agent $a_2$ detects two rounds after, since she is not collocated with $a_1$ (in a configuration similar to $C_2$ in the next phase). This corresponds to Line 14. $a_2$ then meets with $a_3$ and they starts a cautious walk and eventually detects the black hole (after 3 iterations of the cautious walk).
- **If the black hole appears during $C_3$ on node $u_2$.** Then $a_2$ is killed ($a_1$ is also killed two rounds after). Agent $a_3$ moves clockwise until she sees the pebble in $u_1$, and she knows that the black hole is adjacent to the pebble.
- **If the black hole appears during $C_3$ on node $u_3$.** Then agent $a_3$ is killed and one round after $a_2$ is also killed and we are in a configuration similar to $C_1$ with a black hole on $u_2$.

When the black hole appears, one agent is killed after at most $3n$ rounds and the worst time complexity is obtained in the penultimate case, as it requires $n$ additional rounds for the last agent to find the pebble. □

### 3.3 Algorithm $\mathcal{A}_3$: when $n$ is known

We now present Algorithm $\mathcal{A}_3$ assuming $n$ is known, with three synchronous agents, unsafe home base and explicit termination. The algorithm uses a different technique from the previous one.

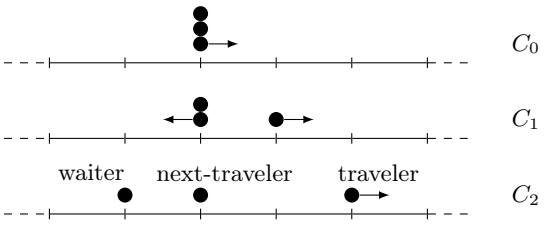


Fig. 2: The first three configuration of Algorithm $\mathcal{A}_3$

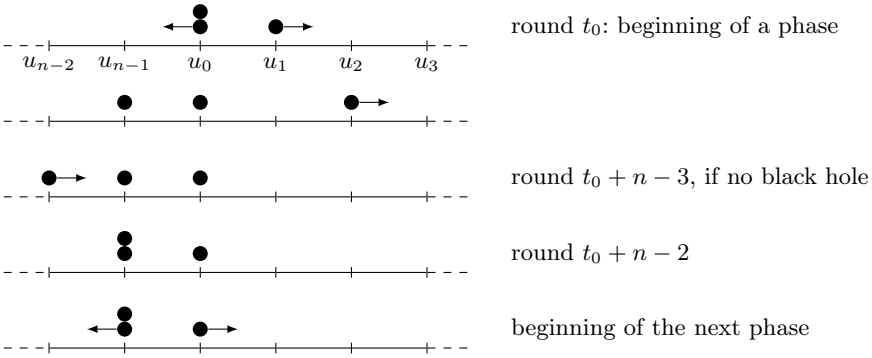


Fig. 3: Execution of Algorithm $\mathcal{A}_3$ when no black hole is detected

The algorithm works as follows: Initially, one agent has a role denoted *traveler* and moves clockwise and the two others stay idle. Then, the agents repeatedly perform the same phase. In a phase, the traveler moves $n-2$ times clockwise and then stays idle for one round. Another agent takes the role *next traveler* and stays idle for $n-1$ rounds. The last agent takes the role *waiter*, moves once counterclockwise, and then stays idle for $n-2$ rounds.

The first three configurations $C_0, C_1, C_2$ are shown in Figure 2. Observe that the first phases starts in configuration $C_1$.

After $n-2$ rounds in the phase, the traveler agent reaches the waiter agent. Then each agent stays idle for one round and then they repeat the phase after exchanging their role: the traveler becomes the waiter, the waiter becomes the next-traveler, and the next-traveler becomes the traveler, as shown in Figure 3.

Observe that at each phase the configuration is similar but rotated by one node counterclockwise.

This phase is repeated until a black hole appears. If a black hole kills the traveler agent while she performs the tour of the ring, then, after $n-2$ round, the waiter agent detects that (Line 28) and moves clockwise to warn the third agent and starts a cautious walk (Line 13) to find the black hole, as shown in Figure 4.

If a black hole appears and kills two collocated agents, then this happens at the beginning of the phase and the traveler is still alive. When the traveler agent finishes its tour, it detects that the waiter agent is not here waiting for her (Line 16), hence knows that the adjacent node contains a black hole, as shown in Figure 5. We obtain the following theorem.

**Theorem 4.** *Assuming $n \geq 5$ (otherwise, run Algorithm $\mathcal{A}_0$ first), Algorithm $\mathcal{A}_3$ solves the EBHS with three synchronous agents that know the size $n$ of the ring, with a worst-case time complexity of $4n$.*

*Proof.* Consider the phase wen the black hole first kills an agent. Let $u_0$ denotes the node where the waiter and the next-traveler are collocated, and $u_1, u_2, \ldots, u_{n-1}$ the nodes clockwise. We have four cases to consider. If the black hole appears in a node $u_i$, $i \in [1, n-2]$ killing only the traveler. Then the traveler never meets with the waiter and the waiter detects the problem and starts a cautious walk with the third agent. One agent correctly identifies the black hole by the correctness of the cautious walk.

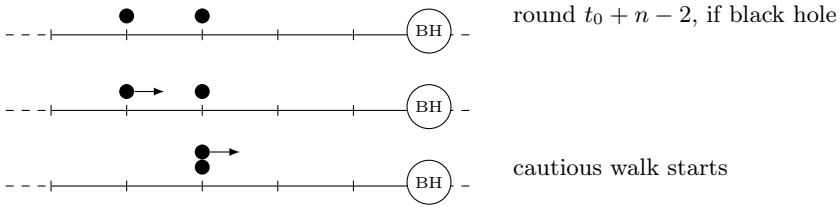


Fig. 4: Execution of Algorithm $\mathcal{A}_3$ when the traveler agent is killed


Fig. 5: Execution of Algorithm $\mathcal{A}_3$ when two agents are killed.

If the black hole appears in $u_0$ killing both the waiter and the next-traveler. Then the traveler detects it when reaching $u_{n-1}$ and the waiter is not present. According to Line 16, the traveler correctly identifies the black hole and terminates.

If the black hole appears in $u_0$ but kills only the next-traveler, then the phase terminates without the two other agents realizing it and a new phase starts. Now we are exactly in the case where the traveler is killed in $u_1$.

If the black hole appears in node $u_{n-1}$ killing first the waiter and then the traveler. Again a new phase starts for the remaining agent, which is now the traveler, and the situation is exactly like when the waiter and the next-traveler are killed in $u_0$.

The worst time complexity is obtained when the traveler is killed and, after waiting for $n$ rounds, the two other agents require at most $3n$ rounds to find the black hole. □

### 3.4 Algorithm $\mathcal{A}_4$ : with quadratic complexity

In the following, we address the case in which $n \geq 3$ is unknown and the number of agents is 3. To make our strategy easier to understand, we first describe the algorithm assuming that the HB is safe.

This simplified algorithm consists in repeating the following for $i = 1, 2, \ldots$: one agent moves to the $i$-th node on one side, and then comes back. Eventually, either $(a)$ the agent does not come back and we know there is a black hole or $(b)$ for a given $i$, the agent reaches the home base from the other side. In case $(a)$ the two remaining agents execute the cautious walk to locate the black hole. In case $(b)$ the three agents apply the algorithm where $n$ is known. The quadratic complexity is obtained when the black hole appears initially adjacent to the HB in the opposite direction of the exploration.

Let us now describe the algorithm assuming an unsafe home base. The pseudo-code of the algorithm is given in Algorithm 5. As before, the red parts correspond to the actions when the presence of the black hole is detected. In the first round, we spread the three agents in three consecutive nodes. Let $u_0, u_1$, and $u_2$ be the three consecutive occupied nodes from left to right, occupied by $a_1$, $a_2$, and $a_3$ respectively. The algorithm repeats a sequence of moves while increasing the value of $i = 1, 2, 3, \ldots$. A given phase $i$ is split into 4 sub-phases (see Figure 6 for an illustration of the agents' moves). The first one lasts $2i$ rounds, and phases (2), (3), and (4) last one round each:

(1) In this sub-phase, $a_3$ performs $i-1$ moves clockwise, waits two rounds, and then performs $i-1$ moves counter-clockwise;
    At the same time, $a_1$ moves one node counter-clockwise, stays there for $2i-2$ rounds, and then moves back one node clockwise; $a_2$ simply stays idle for $2i$ rounds.
(2) both $a_2$ and $a_3$ move counter-clockwise ($a_1$ and $a_2$ becomes collocated on $u_0$);
(3) $a_2$ moves clockwise and becomes collocated with $a_3$ on $u_1$;

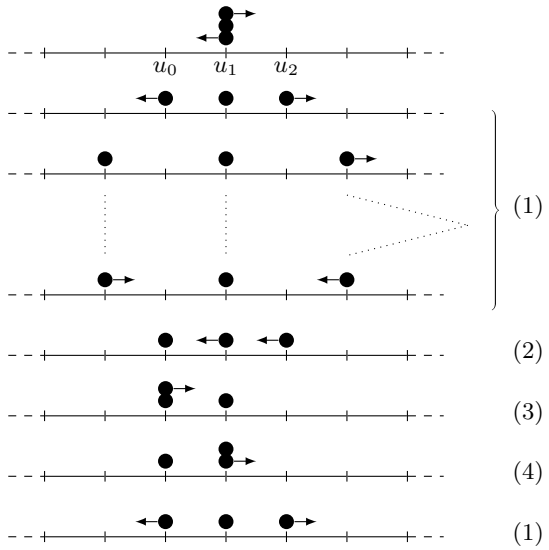


Fig. 6: Illustration of agents' moves by Algorithm $\mathcal{A}_4$

(4) Finally, $a_3$ moves clockwise on $u_2$. After this move, the agents are again spread on $u_0, u_1$, and $u_2$. A new phase starts after incrementing $i$ ($i$ is not incremented if the size of the ring $n$ has been discovered during the phase).

In phase (1), it is possible that $a_3$ meets $a_1$ after performing its $i-1$ moves clockwise. If this happens, then both agents get to know the value of $n$. The agents continue to execute the same algorithm keeping the same value of $i$ ($i = n-2$) for all the next iterations. Observe that in the smallest ring of size $n = 3$, the value of $i$ is already equal to $n-2$ at the first phase, so the algorithm works as well for this case. Recall that for the case $n = 2$, we can run the first part of Algorithm $\mathcal{A}_0$ to detect the case $n = 2$ and solve the problem.

By repeating these actions, the agents can detect a black hole when an agent is missing (is not collocated where she is supposed to be). Informally, if $a_3$ does not come back because it was killed in a black hole during the exploration of the ring, agent $a_2$ eventually knows it (Line 34) and informs $a_1$ to start a cautious walk. If the black hole appears on $u_1$, then $a_2$ is eventually killed and $a_1$ identifies $u_1$ as a black hole because $a_2$ never comes back to $u_0$ (Line 16). If $a_1$ is killed on $u_{n-1}$, then $a_2$ detects it (Line 26) and can tell $a_3$ about the presence of the black hole. Finally, if the black hole kills $a_1$ and $a_2$ on $u_0$, then $a_3$ eventually identifies the node when it sees that $a_2$ is not collocated on $u_1$ (Line 49). The formal case by case analysis is given in the proof of Theorem 5.

Here, the worst case complexity occurs when the black hole is initially on node $u_{n-2}$ as it is discovered when $i = n-3$ (recall that $i = n-2$ corresponds to the case where agent $a_3$ meets with agent $a_1$ on node $u_{n-1}$). The number of round before reaching $i = n-3$ is asymptotically quadratic.

**Theorem 5.** *Assuming $n \geq 3$ (otherwise, run Algorithm $\mathcal{A}_0$ first), Algorithm $\mathcal{A}_4$ solves the EBHS problem with 3 synchronous agents with a quadratic worst-case complexity.*

## 4 Asynchronous agents

We start by giving a simple impossibility result that explains why we only consider algorithms with implicit termination.

**Theorem 6.** *There is no eventually-emerging black hole search algorithm with explicit termination with asynchronous agents.*

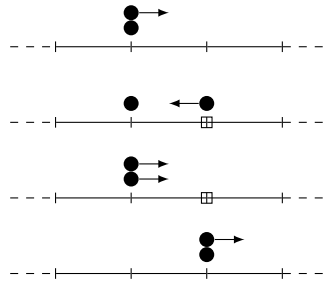

Fig. 7: A cautious move with a pebble.

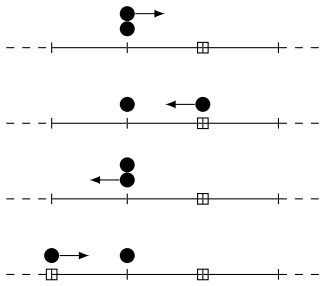

Fig. 8: An agent discovers a pebble when arriving on a node.

One can observe that, with eventual termination, knowing $n$ is not required. This does not contradict previous results, for instance [21,23]. For example, even with an initial black hole, it is possible to find it with implicit termination by using a simple cautious walk with two agents.

In this section, we present two algorithms: one that requires $n$ agents and one that only uses four agents with two pebbles. In both algorithms, the home base is unsafe, and in the second algorithm, $n$ is not known. The number of asynchronous rounds required for all agents to have correct suspicions is asymptotically optimal in $O(n)$.

### 4.1 Algorithm $\mathcal{A}_5$: $n$ asynchronous agents

We propose a simple algorithm $\mathcal{A}_5$ that uses $n$ agents but does not require communication capabilities. The algorithm works as follows: assign one node to each agent *i.e.*, agent $a_i$ is assigned to node $u_i$, for $i \in [0, n-1]$[5]. Then, each agent $a_i$ explores the ring back and forth by avoiding node $u_i$. Also, agent $a_i$ always suspects node $u_i$ to be the black hole. Eventually, after the emergence of the black hole, all agents except one disappear. The surviving agent correctly suspects black hole's location. We have the trivial following theorem.

**Theorem 7.** *For all $n \geq 2$, Algorithm $\mathcal{A}_5$ solves the EBHS with $n$ asynchronous agents without communication, with a worst-case time complexity of $2n$.*

### 4.2 Algorithm $\mathcal{A}_6$: four agents with two pebbles

Algorithm $\mathcal{A}_6$ solves the EBHS problem with four asynchronous agents using only two pebbles. We split the agents into two groups of two agents, each group doing a variant of the cautious walk using pebbles. The first step is to order agents $a_1$ and $a_2$ to move clockwise once; agents $a_3$ and $a_4$ stay idle until they do not see $a_1$ and $a_2$ anymore. Then, each group of agents performs a cautious walk with a pebble, as follows. One of the agents of the group, called the *leader* of the pair, moves in the direction of the walk and puts down a pebble to warn the other pair that they are moving in. Then the leader returns to the other agent, called the *follower*, so that they both move to the target node, and the leader picks up her pebble.

The two pairs of agents may be moving in the same direction or in opposite directions. Figure 7 shows the cautious move with a pebble performed by each pair of agents.

If a leader sees that there is already a pebble or another agent, when arriving on a node, then it moves back to the other agent and the pair of agents starts exploring in the opposite direction, as shown in Figure 8 and 9. The pseudo-code of the algorithm is given in Algorithm 6.

While the follower of a pair waits for the leader to return, the target node is suspected, as shown in Figure 10. Observe that face-to-face communication (*i.e.*, reading the state of the collocated agents) is still possible even in the asynchronous setting, but requires a syncing mechanism. In our variant of the

[5] if the identifiers were in the interval $[1, k^c]$, face-to-face communication would be required to assign the node

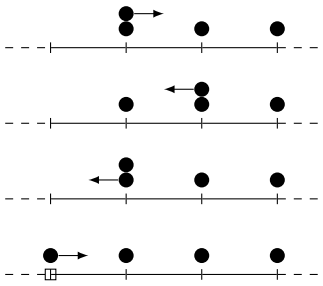

Fig. 9: An agent discovers another agent from the other pair when arriving on a node.

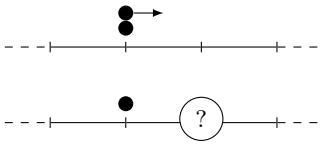


Fig. 10: An agent suspects the adjacent node while waiting for the other agent to return.

cautious walk (Procedure 2) the agents use a `sync` variable. An agent can set a particular value when she sees a collocated agent that she was waiting for to indicate to the other agent that she can continue.

**Theorem 8.** *For all $n \geq 2$, Algorithm $\mathcal{A}_6$ solves the EBHS problem with 4 asynchronous agents and two pebbles, with a worst-case time complexity of $5n$.*

## 5 Conclusion

In this work, we explored the problem of searching for an eventually-emerging black hole in a ring topology using multiple agents under various conditions. We proposed multiple deterministic algorithms for both synchronous and asynchronous settings, each making different assumptions about agent capabilities, knowledge of the network size, and available resources such as pebbles. Our results demonstrate that it is possible to efficiently locate an eventually-emerging black hole with minimal agent assumptions, though trade-offs exist between agent count, memory, and complexity.

Future work could focus on further reducing agent assumptions while maintaining optimal search complexity. Additionally, investigating strategies for handling multiple black holes or dynamic network structures could extend the applicability of these findings to more complex real-world scenarios.

## References


1. B. BALAMOHAN, P. FLOCCHINI, A. MIRI, and N. SANTORO. Time optimal algorithms for black hole search in rings. *Discrete Mathematics, Algorithms and Applications*, 03(04):457–471, December 2011. URL: http://dx.doi.org/10.1142/S1793830911001346, doi:10.1142/s1793830911001346.
2. Balasingham Balamohan, Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Exploring an unknown dangerous graph with a constant number of tokens. *Theor. Comput. Sci.*, 610:169–181, 2016. URL: https://doi.org/10.1016/j.tcs.2014.07.013.
3. Adri Bhattacharya, Giuseppe F. Italiano, and Partha Sarathi Mandal. Black hole search in dynamic tori, 2024. URL: https://arxiv.org/abs/2402.04746, doi:10.48550/ARXIV.2402.04746.
4. Adri Bhattacharya, Giuseppe F. Italiano, and Partha Sarathi Mandal. Searching for a black hole in a dynamic cactus. *Journal of Graph Algorithms and Applications*, 29(2):127–166, May 2025. URL: http://dx.doi.org/10.7155/jgaa.v29i2.3042, doi:10.7155/jgaa.v29i2.3042.
5. Jie Cai, Paola Flocchini, and Nicola Santoro. Decontaminating a network from a black virus. *International Journal of Networking and Computing*, 4(1):151–173, 2014. URL: http://dx.doi.org/10.15803/ijnc.4.1_151, doi:10.15803/ijnc.4.1_151.
6. Jérémie Chalopin, Shantanu Das, Arnaud Labourel, and Euripides Markou. Black hole search with finite automata scattered in a synchronous torus. In *Distributed Computing - 25th International Symposium, DISC*, volume 6950 of *Lecture Notes in Computer Science*, pages 432–446. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-24100-0_41.
7. Jérémie Chalopin, Shantanu Das, Arnaud Labourel, and Euripides Markou. Tight bounds for scattered black hole search in a ring. In *Structural Information and Communication Complexity - 18th International Colloquium, SIROCCO*, volume 6796 of *Lecture Notes in Computer Science*, pages 186–197. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-22212-2_17.
8. Jérémie Chalopin, Shantanu Das, Arnaud Labourel, and Euripides Markou. Tight bounds for black hole search with scattered agents in synchronous rings. *Theoretical Computer Science*, 509:70–85, October 2013. URL: http://dx.doi.org/10.1016/j.tcs.2013.02.010, doi:10.1016/j.tcs.2013.02.010.
9. Colin Cooper, Ralf Klasing, and Tomasz Radzik. *Searching for Black-Hole Faults in a Network Using Multiple Agents*, page 320–332. Springer Berlin Heidelberg, 2006. URL: http://dx.doi.org/10.1007/11945529_23, doi:10.1007/11945529_23.
10. Jurek Czyzowicz, Stefan Dobrev, Rastislav Královic, Stanislav Miklík, and Dana Pardubská. Black hole search in directed graphs. In *Structural Information and Communication Complexity, 16th International Colloquium, SIROCCO*, volume 5869 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2009. URL: https://doi.org/10.1007/978-3-642-11476-2_15.
11. Jurek Czyzowicz, Dariusz R. Kowalski, Euripides Markou, and Andrzej Pelc. Searching for a black hole in tree networks. In *Principles of Distributed Systems, 8th International Conference, OPODIS*, volume 3544 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2004. URL: https://doi.org/10.1007/11516798_5.
12. Jurek Czyzowicz, Dariusz R. Kowalski, Euripides Markou, and Andrzej Pelc. Searching for a black hole in synchronous tree networks. *Comb. Probab. Comput.*, 16(4):595–619, 2007. URL: https://doi.org/10.1017/S0963548306008133.
13. S. Dobrev, P. Flocchini, R. Královič, P. Ružička, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, January 2006. URL: http://dx.doi.org/10.1002/net.20095, doi:10.1002/net.20095.
14. Stefan Dobrev, Paola Flocchini, Rastislav Kralovic, Giuseppe Prencipe, Peter Ruzicka, and Nicola Santoro. Black hole search by mobile agents in hypercubes and related networks. In *Procedings of the 6th International Conference on Principles of Distributed Systems. OPODIS*, volume 3 of *Studia Informatica Universalis*, pages 169–180, 2002.
15. Stefan Dobrev, Paola Flocchini, Rastislav Královič, and Nicola Santoro. Exploring an unknown dangerous graph using tokens. *Theoretical Computer Science*, 472:28–45, February 2013. URL: http://dx.doi.org/10.1016/j.tcs.2012.11.022, doi:10.1016/j.tcs.2012.11.022.
16. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. In *Distributed Computing, 15th International Conference, DISC 2001, Lisbon, Portugal, October 3-5, 2001, Proceedings*, volume 2180 of *Lecture Notes in Computer Science*, pages 166–179. Springer, 2001. URL: https://doi.org/10.1007/3-540-45414-4_12.
17. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: optimal mobile agent protocols. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC*, pages 153–161. ACM, 2002. URL: https://doi.org/10.1145/571825.571853.
18. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In Marina Papatriantafilou and Philippe Hunel, editors, *Principles of Distributed Systems, 7th International Conference, OPODIS*, volume 3144 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2003. URL: https://doi.org/10.1007/978-3-540-27860-3_6.
19. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocols. *Distributed Comput.*, 19(1):1–35, 2006. URL: https://doi.org/10.1007/s00446-006-0154-y.

20. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocols. *Distributed Comput.*, 19(1):1–35, 2006. doi:10.1007/S00446-006-0154-Y.
21. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48(1):67–90, 2007. URL: https://doi.org/10.1007/s00453-006-1232-z.
22. Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Asynchronous gathering in a dangerous ring. *Algorithms*, 16(5):222, April 2023. URL: http://dx.doi.org/10.3390/a16050222, doi:10.3390/a16050222.
23. Stefan Dobrev, Rastislav Kralovic, Nicola Santoro, and Wei Shi. Black hole search in asynchronous rings using tokens. In *Algorithms and Complexity, 6th Italian Conference, CIAC 2006, Rome, Italy, May 29-31, 2006, Proceedings*, volume 3998 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2006. URL: https://doi.org/10.1007/11758471_16.
24. Stefan Dobrev, Nicola Santoro, and Wei Shi. *Locating a Black Hole in an Un-oriented Ring Using Tokens: The Case of Scattered Agents*, page 608–617. Springer Berlin Heidelberg, 2007. URL: http://dx.doi.org/10.1007/978-3-540-74466-5_64, doi:10.1007/978-3-540-74466-5_64.
25. Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. *Exploring and Making Safe Dangerous Networks Using Mobile Entities*, page 136–147. Springer Berlin Heidelberg, 2013. URL: http://dx.doi.org/10.1007/978-3-642-39247-4_12, doi:10.1007/978-3-642-39247-4_12.
26. Paola Flocchini, Matthew Kellett, Peter C. Mason, and Nicola Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, July 2011. URL: http://dx.doi.org/10.1007/s00224-011-9341-8, doi:10.1007/s00224-011-9341-8.
27. Paola Flocchini, Matthew Kellett, Peter C. Mason, and Nicola Santoro. *Fault-Tolerant Exploration of an Unknown Dangerous Graph by Scattered Agents*, page 299–313. Springer Berlin Heidelberg, 2012. URL: http://dx.doi.org/10.1007/978-3-642-33536-5_30, doi:10.1007/978-3-642-33536-5_30.
28. Paola Flocchini, Matthew Kellett, Peter C. Mason, and Nicola Santoro. *Finding Good Coffee in Paris*, page 154–165. Springer Berlin Heidelberg, 2012. URL: http://dx.doi.org/10.1007/978-3-642-30347-0_17, doi:10.1007/978-3-642-30347-0_17.
29. Pritam Goswami, Adri Bhattacharya, Raja Das, and Partha Sarathi Mandal. Perpetual Exploration of a Ring in Presence of Byzantine Black Hole. In Silvia Bonomi, Letterio Galletta, Etienne Rivière, and Valerio Schiavoni, editors, *28th International Conference on Principles of Distributed Systems (OPODIS 2024)*, volume 324 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:17, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.OPODIS.2024.17, doi:10.4230/LIPIcs.OPODIS.2024.17.
30. Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. *Hardness and Approximation Results for Black Hole Search in Arbitrary Graphs*, page 200–215. Springer Berlin Heidelberg, 2005. URL: http://dx.doi.org/10.1007/11429647_17, doi:10.1007/11429647_17.
31. Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Hardness and approximation results for black hole search in arbitrary networks. *Theoretical Computer Science*, 384(2–3):201–221, October 2007. URL: http://dx.doi.org/10.1016/j.tcs.2007.04.024, doi:10.1016/j.tcs.2007.04.024.
32. Adrian Kosowski, Alfredo Navarra, and Cristina M. Pinotti. *Synchronization Helps Robots to Detect Black Holes in Directed Graphs*, page 86–98. Springer Berlin Heidelberg, 2009. URL: http://dx.doi.org/10.1007/978-3-642-10877-8_9, doi:10.1007/978-3-642-10877-8_9.
33. Giuseppe Antonio Di Luna, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Black hole search in dynamic rings: The scattered case. In Alysson Bessani, Xavier Défago, Junya Nakamura, Koichi Wada, and Yukiko Yamauchi, editors, *27th International Conference on Principles of Distributed Systems, OPODIS 2023, December 6-8, 2023, Tokyo, Japan*, volume 286 of *LIPIcs*, pages 33:1–33:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2023.33, doi:10.4230/LIPICS.OPODIS.2023.33.
34. Giuseppe Antonio Di Luna, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Locating a black hole in a dynamic ring. *J. Parallel Distributed Comput.*, 196:104998, 2025. URL: https://doi.org/10.1016/j.jpdc.2024.104998, doi:10.1016/J.JPDC.2024.104998.
35. Euripides Markou and Michel Paquette. *Black Hole Search and Exploration in Unoriented Tori with Synchronous Scattered Finite Automata*, page 239–253. Springer Berlin Heidelberg, 2012. URL: http://dx.doi.org/10.1007/978-3-642-35476-2_17, doi:10.1007/978-3-642-35476-2_17.
36. Jaydip Sen, M. Girish Chandra, S.G. Harihara, Harish Reddy, and P. Balamuralidhar. A mechanism for detection of gray hole attack in mobile ad hoc networks. In *2007 6th International Conference on Information, Communications & Signal Processing*, page 1–5. IEEE, 2007. URL: http://dx.doi.org/10.1109/ICICS.2007.4449664, doi:10.1109/icics.2007.4449664.

## A Omitted Proofs

**Theorem 1.** *There is no algorithm solving the EBHS problem with only two synchronized agents, even with a safe home base, whiteboards, the knowledge of n, and implicit termination.*

*Proof (Proof of Theorem 1).* Assume, for the sake of contradiction, that such an algorithm exists.

We denote by $1 = t_0, t_1, t_2, \ldots$ all the rounds, in increasing order, where the two agents are collocated in an execution where no black hole appears (this sequence contains 1 because the agents are initially collocated, but can be finite or infinite). It is clear that the agents are always collocated on the home base, otherwise a black hole could appear where they are both located, killing them both.

We denote by $P_i$, resp. $Q_i$, the set of nodes visited by the first agent, resp. the second agent, between round $t_i$ and $t_{i+1}$ (or in the infinite remaining execution if $t_{i+1}$ does not exist, these sets are still well-defined because the ring is finite).

If $u_0$ denotes the home base, we saw that $u_0 \in P_i \cap Q_i$, for all $i \geq 0$. If there exists another node $v \neq u_0$ such that $v \in P_i \cap Q_i$ for some $i$, then if the black hole appears on node $v$ at round $t_i$, since the agents do not meet before $t_{i+1}$, they will perform the same movements as if the black hole was not there, hence they will explore $P_i$ and $Q_i$ respectively, and both end up in the black hole.

So we know that $P_i \cap Q_i = \{u_0\}$ for all $i \geq 0$.

If there are two nodes $v_0$ and $v_1$ that are never visited, then the algorithm cannot distinguish the two executions where the black hole appears in $v_0$ and in $v_1$, hence will fail in one of the two executions, a contradiction.

So now consider a ring of size $n \geq 5$, there exists a $i$ such that, without loss of generality $u_2 \in P_i$ or $u_3 \in P_i$ (*i.e.*, one of the two farthest node to the home base is visited by an agent). Consider without loss of generality that $u_0, u_1, u_2 \in P_i$, and $u_1, u_2 \notin Q_i$. Hence, the agent exploring $Q_i$ cannot distinguish the two executions where the first agent is killed on $u_1$ and on $u_2$. If the agent suspects one of the nodes, she will be wrong in the execution where the black hole appears in the other node, which is a contradiction. □

**Theorem 5.** *Assuming $n \geq 3$ (otherwise, run Algorithm $\mathcal{A}_0$ first), Algorithm $\mathcal{A}_4$ solves the EBHS problem with 3 synchronous agents with a quadratic worst-case complexity.*

*Proof (Proof of Theorem 5).* Denote by $u_0, u_1$, and $u_2$ the nodes where agent $a_1$, $a_2$, and $a_3$ are respectively located at round 2 (round 1 being the initial configuration). We can assume without loss of generality that the black hole appears at a node exactly when an agent arrives at this node. So we now consider all the possible cases where a black hole can appear and check if the algorithm succeeds at making the agents detect its position. Note that during one iteration (for a given value of $i$), Agents $a_1$ and $a_2$ perform only a finite number of moves (that do not depend on $i$).

1. The black hole appears when $a_1$ arrives at $u_0$. This eventually also kills agent $a_2$, when it moves to $u_0$ in sub-phase (3). Then, $a_3$ detects that $a_2$ is not collocated on node $u_1$ on sub-phase (4) (Line 49). Hence, $a_3$ identifies correctly $u_0$ as the black hole (the adjacent node on the counterclockwise direction). It is important here to notice that agent $a_3$ cannot be killed by moving on $u_0$ as well, because the value of $i$ cannot be larger than $n-2$. Since $a_1$ is killed in the current iteration, this means that, either the value of $n$ is already known, and $a_3$ uses the value $i = n-2$ to avoid $u_0$, or $n$ is not known, which means the previous iteration was $i' < n-2$ and the current iteration is $i \leq n-2$, avoiding $u_0$ as well.
2. The black hole appears when $a_1$ arrives at $u_{n-1}$. In this case, $a_2$ detects that $a_1$ is not collocated on $u_0$ in sub-phase (3) (Line 26) and knows that the black hole is on $u_{n-1}$. $a_2$ can still move to $u_1$ to see if $a_3$ is still alive so that they both stops and identify the black hole. If $a_3$ is not there, the $a_2$ knows that both $a_1$ and $a_3$ are killed on $u_{n-1}$.
3. The black hole appears when $a_2$ arrives at $u_0$ then, both $a_1$ and $a_2$ are killed in the black hole and the first case 1 applies.

4. The black hole appears when $a_2$ arrives at $u_1$ (after sub-phase (3)) then, both $a_2$ and $a_3$ are killed in the black hole. The next iteration starts for agent $a_1$, which moves to node $u_{n-1}$ and then moves back to $u_0$ after $2i$ rounds. When $a_1$ detects that it is not collocated with $a_2$ on node $u_0$ (Line 16), in sub-phase (4), it knows that the black hole is on $u_1$ (because it is the only other node $a_2$ is traveling to).
5. The black hole appears when $a_3$ arrives at $u_1$ then, in the next round, $a_2$ is also killed and we retrieve Case 4.
6. The black hole appears when $a_3$ arrives at any node $u_r$,

$$r \in \{2, \ldots, n-2\}.$$

After $2i$ rounds from the beginning of the current iteration, $a_2$ detects that $a_3$ has been killed as it is not collocated on node $u_1$ (in sub-phase (4)) (Line 34). Then agent $a_2$ can move directly counterclockwise to meet $a_1$ on $u_0$, informs her that $a_3$ has been killed and both of them execute a cautious walk. Hence, the black hole is correctly identified by the property of the cautious walk.

Observe that when the size of the ring becomes known, the same algorithm is executed but the value of $i$ is not incremented, so that agent $a_3$ never goes to node $u_0$. Thus the previous cases apply in this case as well.

For the time complexity, the worst case is obtained when the black hole appears initially in a node that is reached by $a_3$, the traveling agent, during an iteration $i$ close to $n-2$, which kills the agent after a quadratic number of rounds. □

**Theorem 6.** *There is no eventually-emerging black hole search algorithm with explicit termination with asynchronous agents.*

*Proof (Proof of Theorem 6).* Assume for the sake of contradiction that such an algorithm exists. Consider an execution with an initial black hole at some node $u$. Consider the time $t$ where a surviving agent $a$ detects the black hole and explicitly terminates. Consider another execution (indistinguishable from the previous one) where node $u$ is not a black hole but all agents entering $u$ are slowed down until $t$. In this scenario, agent $a$ wrongly detects $u$ as the black hole. Observe that this proof does not work in the case of an initial black hole, since a particular node can be isolated after all the other nodes have been checked, so that the position of the black hole can be known for sure. In our case, checking a node does not mean that this node is safe forever. □

**Theorem 7.** *For all $n \geq 2$, Algorithm $\mathcal{A}_5$ solves the EBHS with $n$ asynchronous agents without communication, with a worst-case time complexity of $2n$.*

*Proof (Proof of Theorem 7).* When the black hole appears in $u_i$, all the agents that are assigned to $u_j$ with $j \neq i$ are eventually killed. In the works case, $2n$ asynchronous rounds is required for all such agents to be killed, so that that only the agent correctly suspecting $u_i$ remains. □

**Theorem 8.** *For all $n \geq 2$, Algorithm $\mathcal{A}_6$ solves the EBHS problem with 4 asynchronous agents and two pebbles, with a worst-case time complexity of $5n$.*

*Proof (Proof of Theorem 8).* If a black hole appears and kills a single agent, then the other pair eventually finds the black hole, either directly thanks to the cautious walk, or after turning around because they find a pebble, hence finding the black hole after traversing the ring in the other direction.

If a black hole appears while killing two agents of the same pair. Either they did not leave a pebble, and the other pair eventually finds the black hole (one agent of the pair ends up in the black hole, and the other waits infinitely suspecting the black hole), or they left a pebble and the other pair either find the black hole as before, or they first find the pebble, turn back and find the black hole after traversing the ring in the other direction.

If the black hole kills two agents of different pairs (or even three agents), it is important to notice that the two followers are never on the same node, so at least one follower is still alive and suspects the black hole infinitely. Indeed, the followers of both groups cannot be on the same node (except initially, which we assumed to be safe) because the follower moves only when the leader knows that the other pair has not yet arrived on the target node. More formally, a follower only moves towards a node that contains the pebble of the corresponding leader, and the leaders never put their pebble on the same node at the same time.

When the black hole appears, the worst-case time complexity is similar to the cautious walk ($3n$) but with two additional rounds of synchronization per iteration, hence $5n$ □

## B Omitted Algorithms

**Algorithm 3:** $\mathcal{A}_0$: Small Rings

```
   Input:3 or 4 agents a_1, a_2, a_3, a_4
 1 if agent is a_4 then
 2     WAIT until termination;
 3 else if agent is a_1 then
 4     MOVE counter-clockwise;
 5     if collocated with a_2 then
 6         n ← 2;
 7         while true do
 8             MOVE counter-clockwise;
 9     else
10         WAIT 1 round;
11         execute A_4 as agent a_1;
12 else if agent is a_2 then
13     WAIT 2 rounds;
14     if collocated with a_1 then
15         n ← 2; WAIT 1 round;
16         while true do
17             WAIT 1 round;
18             if not collocated with a_1 then
19                 bh ← none;
20                 TERMINATE;
21             WAIT 1 round;
22     else
23         execute A_4 as agent a_2;
24 else if agent is a_3 then
25     MOVE clockwise;
26     if collocated with a_1 then
27         n ← 2;
28         WAIT 1 round;
29         while true do
30             WAIT 1 round;
31             if not collocated with a_1 then
32                 bh ← none;
33                 TERMINATE;
34             WAIT 1 round;
35     else
36         WAIT 1 round;
37         execute A_4 as agent a_3;
```

**Algorithm 4:** $\mathcal{A}_3$: Three-Agent EBS Solution with the knowledge of $n$

```
Input:3 agents a_1, a_2, a_3 /*** Assign initial role to the agents ***/
if agent is a_1 then
    role ← traveler;
    MOVE clockwise;
else if agent is a_2 then
    role ← next-traveler;
    WAIT 1 round;
else if agent is a_3 then
    role ← waiter;
    WAIT 1 round;
/*** Main loop: after each iteration the roles are exchanged, and the configuration is rotated one node counter-clockwise ***/
while true do
    if role = traveler then
        if collocated with waiter then
            execute CautiousWalk(waiter, traveler, clockwise);
        MOVE clockwise (n − 2 times);
        if not collocated with waiter then
            bh ← node in the clockwise direction;
            TERMINATE;
        role ← waiter;
        WAIT 1 round;
    else if role = next-traveler then
        WAIT n − 2 rounds;
        role ← traveler;
        WAIT 1 round;
    else if role = waiter then
        MOVE counter-clockwise;
        WAIT n − 3 rounds;
        if not collocated with traveler then
            MOVE clockwise;
            execute CautiousWalk(waiter, traveler, clockwise);
        role ← next-traveler;
        WAIT 1 round;
```

**Algorithm 5:** $\mathcal{A}_4$: Three-Agent EBS Solution with quadratic complexity

```
   Input:3 agents a_1, a_2, a_3
 1 n ← +∞;
 2 i ← 1;
 3 detection ← false;
 4 if agent is a_1 then
 5     MOVE counter-clockwise;
 6     while true do
 7         if collocated with a_2 then
 8             execute CautiousWalk(a_1, a_2, clockwise);
 9         MOVE counter-clockwise;
10         WAIT i − 1 rounds;
11         if collocated with a_1 then
12             n ← i + 3;
13         WAIT i − 1 rounds;
14         MOVE clockwise;
15         WAIT 1 round;
16         if not collocated with a_2 then
17             bh ← node in the clockwise direction;
18             TERMINATE;
19         WAIT 2 rounds;
20         i ← min(n − 3, i + 1);
21 else if agent is a_2 then
22     WAIT 1 round;
23     while true do
24         WAIT 2i rounds;
25         MOVE counter-clockwise;
26         if not collocated with a_1 then
27             BH is adjacent in the counter-clockwise direction;
28             detection ← true;
29             MOVE clockwise;
30             TERMINATE;
31         if a_1.n < +∞ then
32             n ← a_1.n;
33         MOVE clockwise;
34         if not collocated with a_3 then
35             MOVE counter-clockwise;
36             execute CautiousWalk(a_1, a_2, clockwise);
37         WAIT 1 round;
38         i ← min(n, i + 1);
39 else
40     MOVE clockwise;
41     while true do
42         MOVE clockwise (i − 1 times);
43         WAIT 1 round;
44         if collocated with a_1 then
45             n ← i + 3;
46         WAIT 1 round;
47         MOVE counter-clockwise (i times);
48         WAIT 1 round;
49         if not collocated with a_2 then
50             BH is adjacent in the counter-clockwise direction;
51             TERMINATE;
52         if a_2.detection = true then
53             BH is two hop in the counter-clockwise direction;
54             TERMINATE;
55         MOVE clockwise;
56         i ← min(n, i + 1);
```

**Algorithm 6:** $\mathcal{A}_5$: $n$-agent EBS Solution (knowledge of $n$ or face-to-face communication)

**Input:**$n$ agents $a_1, a_2, \dots a_n$
*id*: id of the agent executing the algorithm, in $[0, n-1]$
`direction` $\leftarrow 1$ (meaning clockwise);
`current_node` $\leftarrow 0$;
`bh` $\leftarrow id$ // global id of the suspected node
**if** $id = 0$ **then**
    `MOVE` clockwise;
    `current_node` $\leftarrow 1$;
**loop**
    `next_node` $\leftarrow$ current_node + `direction` mod $n$;
    **if** `next_node` $= id$ **then**
        `direction` $\leftarrow$ ;
        `direction` $\times(-1)$;
    `current_node` $\leftarrow$ current_node + `direction` mod $n$;
    `MOVE` direction;

**Procedure 2:** CautiousWalkPebble($leader, follower$)

*direction* $\leftarrow 1$ (clockwise)
*sync* $\leftarrow 0$
**if** *agent is leader* **then**
    **while** *true* **do**
        `PICKUP` pebble (if any) and `MOVE` towards *direction*
        **if** *collocated or there is pebble* **then**
            *direction* $\leftarrow$ opposite of *direction*
            `MOVE` towards *direction*
            `WAIT` $follower.direction = direction$ // when the other sees me
            **continue**
        *sync* $\leftarrow 1$
        `DROP` pebble and `MOVE` towards the opposite of *direction*
        `WAIT` not collocated with $follower$ // when the other saw me and moved
        *sync* $\leftarrow 2$
        `MOVE` towards *direction*
        `WAIT` collocated with $follower$
        // when I see the other
        *sync* $\leftarrow 0$
        `WAIT` $followe.sync = 0$// when the other sees me
**if** *agent is* $follower$ **then**
    **while** *true* **do**
        `WAIT` collocated with *leader* and (*leader.direction* is opposite or $leader.sync = 1$)
        **if** *leader.direction is opposite* **then**
            *direction* $\leftarrow$ opposite of *direction*
            **continue**
        *sync* $\leftarrow 2$
        `MOVE` towards *direction*
        `WAIT` collocated with *leader*// when I see the other
        *sync* $\leftarrow 0$
The agents use the `WAIT` instruction to stay in sync. An agent executing this instruction checks if the condition is true. If so, she then execute the next instruction, otherwise stays idle and remains on the same instruction.

**Algorithm 6:** $\mathcal{A}_6$: EBS Solution with 4 asynchronous agent and 2 pebbles

**Input:**4 agents $a_1, a_2, a_3, a_4$
**if** *agent is in* $\{a_1, a_2\}$ **then**
    `MOVE` clockwise;
    execute CautiousWalkPebble($a_1$, $a_2$);
**else**
    `WAIT` until $a_1$ and $a_2$ have moved;
    execute CautiousWalkPebble($a_3$, $a_4$);