Time Optimal Algorithms for Black Hole Search in Rings *

B. Balamohan[1], P. Flocchini[1], A. Miri[2], and N. Santoro[3]

[1] University of Ottawa, Ottawa, Canada,
{bbala078,flocchin}@site.uottawa.ca
[2] Ryerson University, Toronto, Canada. samiri@scs.ryerson.ca
[3] Carleton University, Ottawa, Canada,
santoro@scs.carleton.ca.

Abstract. In a network environments supporting mobile entities (called robots or agents), a *black hole* is harmful site that destroys any incoming entity without leaving any visible trace. The *black-hole search* problem is the task of a team of $k > 1$ mobile entities, starting from the same safe location and executing the same algorithm, to determine within finite time the location of the black hole. In this paper we consider the black hole search problem in asynchronous *ring* networks of n nodes, and focus on the *time complexity*.
It is known that any algorithm for black-hole search in a ring requires at least $2(n-2)$ time in the worst case. The best algorithm achieves this bound with a team of $n-1$ agents with an average time cost $2(n-2)$, equal to the worst case. In this paper we first show how the same number of agents using 2 extra time units from optimal in the worst case, can solve the problem in only $\frac{7}{4}n - O(1)$ time on the *average*. We then prove that the optimal average case complexity $\frac{3}{2}n - O(1)$ can be achieved without increasing the worst case using $2(n-1)$ agents Finally we design an algorithm that achieves asymptotically optimal both worst case and average case time complexity employing an optimal team of $k = 2$ agents, thus improving on the earlier results that required $O(n)$ agents.

1 Introduction

1.1 The Problem

Black Hole Search (BHS) is a multi-agents problem set in graph G: a team of (identical) cooperating mobile entities called agents (or robots) must determine the location in G of a *black hole* (BH): a node where any incoming agent is destroyed without leaving any detectable trace. The problem is solved if at least one agent survives and knows the location of the black hole.

A black hole can model several types of faults, both hardware and software, arising in networked systems with code mobility. For example, the crash failure of a site in an asynchronous network turns such a site into a black hole; similarly,

* Research partially supported by NSERC.

the presence at a site of a malicious process (e.g., a virus) that thrashes any incoming message (e.g., by classifying it as spam) also renders that site a black hole. Clearly, in presence of such a harmful host, the first step must be to *identify* it, if possible; i.e., to determine and report its location; following this phase, a "rescue" or "repair" activity would conceivably be initiated [16]. The black hole search problem is also theoretically interesting because it is a generalization of the classical problem of *graph exploration* (e.g., see [1, 7, 17]). In fact, it is easy to see that to locate a black hole the agents have to necessarily "explore" all safe nodes; in this exploration process some agents may disappear in the black hole. In other words, while the existing wide body of literature on exploration assumes that the graph is *safe*, BHS opens the problem of the exploration of *dangerous graphs*.

In this paper we consider the *Black Hole Search* problem in a *ring* network.

1.2 Related Work

This black hole search problem has been originally studied in *ring* networks [12] and has been extensively investigated in various settings since then (e.g., see [4, 6, 8, 10, 13, 14, 19, 24]).

In order to locate the black hole, some of the agents of the team will necessarily have to enter the dangerous site. The goal of all location algorithms studied in the literature is to minimize the *size* of the exploring team, the number of *moves* performed by the agents and the *time* spent in the search.

The main distinctions made in the literature are whether the system is synchronous or asynchronous, and whether the agents communicate through whiteboards or by using tokens.

The majority of the work focuses on the *asynchronous whiteboard model*, which is the one considered in this paper. In this model, there are no assumptions on the time required for each operation or movement other than it is finite. Each network node provides a shared memory area, the *whiteboard*, which visiting agents can access (in fair mutual exclusion) to write on and/or read from. The communication and coordination between agents takes place solely via the whiteboards. Within this model, a complete characterization has been done for the localization of a black hole in ring networks [12], providing protocols that are optimal in size, time, and asymptotically move-optimal number of moves. In [10], arbitrary topologies have been considered and asymptotically optimal location algorithms have been proposed under a variety of assumptions on the agents' knowledge (knowledge of the topology, presence of sense of direction). An improved algorithm when the topology is known has been described in [11], while optimal algorithms for common interconnection networks have been studied in [8]. In [19] the effects of knowledge of incoming link on the optimal team size has been studied and lower bounds provided. The case of black links in arbitrary networks has been studied in [2, 15], respectively for anonymous and non-anonymous nodes. Black hole search in directed graphs has been investigated for the first time in [4], where it is shown that the requirements in number of agents change considerably. A variant of dangerous node behavior has been

studied in [22], where the authors introduce black holes with Byzantine behavior (they do not always destroy a passing agent) and consider the periodic ring exploration problem.

In the *asynchronous token* model, there are no whiteboards, but each agent is provided with *pebbles* that it can place on (and pick up from) a node; the communication and coordination among agents is achieved solely by placing on the nodes. This model has been investigated in [9, 13, 14, 24].

In *synchronous* networks, where movements are synchronized and it is assumed that it takes one unit of time to traverse a link, the techniques and the results are quite different. Tight bounds on the number of moves have been established for some classes of trees [6]. In the case of general networks finding the optimal strategy is shown to be NP-hard [5, 20] and approximation algorithms are given in [20,21]. The case of multiple black holes have been investigated in [3] where a lower bound on the cost and close upper bounds are given.

1.3 Main Contributions

In this paper we turn our attention to the *time complexity* of locating a black hole in a ring of n nodes using a team of $k > 1$ *asynchronous* agents communicating by means of whiteboards (shared memory available at each node). The asynchrony of the computational entities means that the algorithm must work regardless of the time required for each computation or movement, which is finite but a priori unknown (i.e., determined by an adversary); however, the time complexity of the algorithm is measured only over those executions where time delays are unitary (i.e., determined by a synchronous scheduler), as traditional in distributed computing (e.g., [12, 18, 23]).

It has been shown in [12] that any asynchronous black hole search algorithm for rings requires $T_{worst}(n,k) = 2(n-2)$ time in the worst case regardless of the number $k > 1$ of agents. The best algorithm achieves this bound with a team of $k = n-1$ agents with an average time cost $2(n-2)$, equal to the worst case [12].

In this paper we first show how the same number of agents can solve the problem using on the *average* only $\frac{7}{4}n$ time, and in the worst case 2 extra time units from optimal. We also show that any asynchronous black hole search algorithm for rings requires $T_{average}(n,k) = \frac{3}{2}n$ time regardless of the number $k > 1$ of agents, and then prove that, with $2(n-1)$ agents, the optimal average case complexity $\frac{3}{2}n - O(1)$ can be achieved *without increasing the worst case*. Finally, observing that all considered protocols achieve (worst and average) $\Theta(n)$ time using $O(n)$ agents, we prove that it is possible to locate a black hole in asymptotically optimal (worst and average) $\Theta(n)$ time with just $k = 2$ agents. In fact, we design an algorithm that uses $8n + O(1)$ time in the worst case and $\frac{15}{2}n + O(1)$ on the average, employing an optimal team of 2 agents thus improving the earlier result that employed $n-1$ agents. These results are summarized in Table 1. The costs in terms of moves of all these algorithms is $O(n^2)$, the same as that of the algorithm in [12] they improve upon.

.

Algorithm	Agents	Time Complexity	
		Average	Worst
[12]	$n-1$	$2(n-2)$	$2(n-2)$ $(\star)$
GROUP	$n-1$	$\frac{7}{4}\ n - O(1)$	$2(n-1)$
OPTAVGTIME	$2(n-1)$	$\frac{3}{2}\ n - O(1)$ $(\star)$	$2(n-2)$ $(\star)$
OPTTEAMSIZE	2 $(\star)$	$\frac{15}{2}n + O(1)$	$8n + O(1)$

Table 1. Summary of results; $(\star)$ indicates an optimal exact bound.

2 Preliminaries

2.1 Definitions and Notations

The network environment is a ring $\mathcal{R}$ of n anonymous nodes (for simplicity indicated as $0, 1, \cdots, n-1$ in clockwise direction). Each node has two ports, labelled *left* and *right*. Without loss of generality we assume that this labeling is globally consistent and the ring is oriented (if it is not the case, orientation can be easily obtained) . Each node is equipped with a limited amount of storage, called *whiteboard*. For all our algorithms $O(\log n)$ bits of storage are sufficient

In this network there is a set $\mathcal{A}$ of *anonymous* (i.e., identical) mobile agents, which are all initially located on the same node, called the *homebase* (w.l.g. node 0). The topology is known to the agents, as well as the number of nodes (as shown in [12] not knowing the number of nodes make the location process impossible). The agents can move from node to neighboring node in $\mathcal{R}$ and have computing capabilities and bounded storage. The agents obey the same set of behavioral rules, the protocol, and all their actions are performed asynchronously, i.e., they take a finite but unpredictable amount of time. The agents communicate by writing on and reading from the whiteboards. Access to the whiteboards is governed by fair mutual exclusion.

A *black hole* (BH) is a stationary process located at a node, which destroys any agent arriving at the node; no observable trace of such a destruction will be evident external to the node in which black hole is located. The *Black Hole Search* problem is the one of *finding the location of the black hole*. More precisely, the black hole search problem is solved if at least one agent survives, and all surviving agents know the location of the black hole within a finite amount of time.

We evaluate the efficiency of our solutions based on the following measures:

1. *Number of agents* used/needed in the protocol.
2. *Total number of moves* performed by all agents.
3. *The amount of time* between the earliest start time of the protocol by any agent and the time all the agents that started the protocol have terminated the execution of protocol. Since the system is asynchronous, when evaluating the time complexity we will employ *ideal time*; i.e., we will assume that it time delays are unitary (e.g., see [12, 18, 23]).

2.2 Cautious Walk

We first recall the *cautious walk* technique, which is central to the algorithms presented in this paper, and the existing asymptotically optimal algorithm of [12].

At any time during the search for the black hole, the ports (corresponding to the incident links) of a node can be classified as follows:

1. *unexplored*: if no agent has moved across this port.
2. *safe*: if an agent arrived via this port.
3. *active*: if an agent departed via this port, but no agent has arrived via it.

Clearly, both unexplored and active links are dangerous in that they might lead to the black hole; however, active links are being explored, so there is no need for another agent to go there unless it is declared safe. Cautious walk is defined by the following two rules:

1. when an agent moves from node u to v via an unexplored port (turning it into active), if it does not disappears (i.e., v is not the black hole) the agent immediately returns to u (making the port safe), and only then resumes its execution;
2. no agent leaves via an active port.

3 Improved Algorithm

In this section we improve on the average time complexity of [12]. We describe an algorithm that uses only $n-1$ agents, as in [12], but only $\frac{7}{4}n + O(1)$ time on the average (instead of $2(n-2)$). The worst case is $2(n-1)$ (instead of $2(n-2)$).

The idea is to determine the location of the black hole on some node by having a particular pair of agents (witnessing pair) returning successfully to the homebase after exploring a subset of nodes that does not include the black hole. The way subsets of nodes are associated to agents is complicated by the objective of reducing the number of agents entering the black hole.

We recall that node 0 indicates the homebase and the other nodes are indicated as $1, 2, \ldots, n-1$ in clockwise direction. For simplicity of description let $n = 4q + 1$. The $4q$ agents are divided into four groups: *Left*, *Right*, *Middle* and *TieBreakers*.

The groups *Left* and *Right* contain q agents each. The *Middle* group consists of $q+1$ agents and the *TieBreakers* group consists of $q-1$ agents.

The witnessing pairs are chosen in a different way depending on the potential location of the black hole. If the black hole is far from the homebase, it will be witnessed by a pair from *Left* and *Middle* or from *Right* and *Middle*. If instead, the black hole is closer to the homebase the witnessing pair will belong to *TieBreakers* and *Right* or to *TieBreakers* and *Left*.

More precisely, the idea is that the agents of the *Left*, *Right* and *Middle* groups explore each a region of size $3q$ appropriately chosen in such a way that the $2q$ nodes farthest from the homebase are endpoints of complements of explored areas of some agents. In other words, the presence of the black hole in one of those $2q$ nodes would be witnessed by a pair of agents being able to successfully return after their exploration.

The agents of the *Tiebreakers* group are instead used to pair themselves either with an agent from *Right* or with one from $Left$ to locate the black hole when it is within q nodes from the homebase. The details are given in Algorithm 1.

Algorithm 1 Algorithm GROUP

1. The Left group consists of $left_i$ for $1 \leq i \leq q$. An agent $left_i$ in this group explores all node except the nodes $\{i, i+1, i+2, \cdots, i+q-1\}$. It moves left first, then right and then returns to homebase. (See Figure 3).
2. The Right group consists of $right_i$ for $1 \leq i \leq q$. An agent $right_i$ in this group explores all node except the nodes $\{n-i-q, n-i-q+1, \cdots, n-i\}$. It moves right first, then left and then returns to homebase.
3. The Middle group consists of $middle_i$ for $1 \leq i \leq q+1$. An agent $middle_i$ in this group explores all node except the nodes $\{q+i-1, q+i, \cdots, 2q+i-1\}$. It moves left first, then right and then returns to homebase. (See Figure 3).
4. The Tiebreaker group consists of $tiebreaker_i$ for $1 \leq i \leq q-1$. An agent $tiebreaker_i$ in this group explores all nodes except nodes $\{i, i-1, i-2, \cdots, 1\}$ at the right of the homebase starting as soon as either $right_{i+1}$ or $left_{i+1}$ passes through the homebase. (See Figure 3).
5. The black hole is located on node i iff one of the following witnessing pairs return safely: ($left_i$,$middle_i$), ($right_i, middle_{q-i}$), ($tiebreaker_i$,$left_i$), ($tiebreaker_i$,$right_i$).

We have that:

Theorem 1. *Algorithm* GROUP *solves the black hole search problem with average time* $\frac{7n}{4} + O(1)$ *and worst case time* $2(n-1)$.

Proof. Let us first consider the correctness of the algorithm. If the black hole is one of nodes $\{q, q+1, q+2, \cdots, 2q\}$ then it is the unique node in the intersection of the excluded segments of a pair of agents $left_i$ and $middle_i$ from the *Left* and

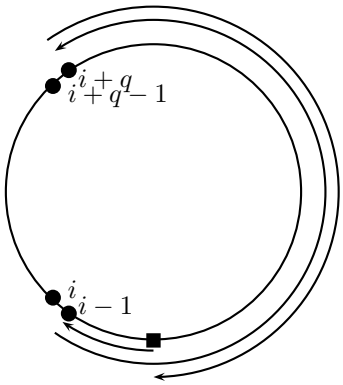

Fig. 1. Algorithm GROUP: Protocol For Left Group

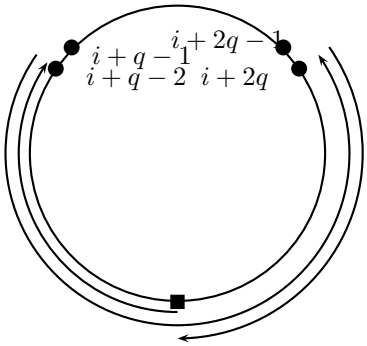

Fig. 2. Algorithm GROUP: Protocol For Middle Group

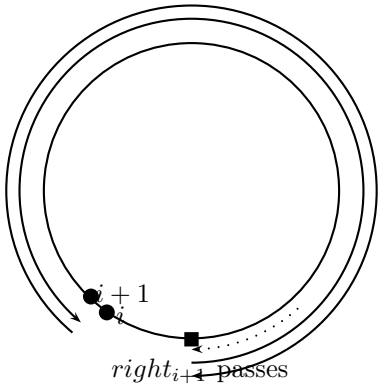

Fig. 3. Algorithm GROUP: Protocol For Tie-Breaker Group

the *Middle* groups. (for example, $q+1$ is the only node not explored by the pair $left_2$, $middle_2$). In such a case the black hole is located by the return of such a witnessing pair and the location takes $\frac{3}{2}(n-2)$ time units as both agents $left_i$ and $middle_i$ explore all but $q = \frac{1}{4}(n-1)$ agents. If the black hole is in node i with $i < q$ then agent $right_{i+1}$, after exploring the i nodes on the right of the homebase, passes back through the homebase. At this moment, by the rules of the algorithm, agent $tiebreaker_i$ starts exploring all the nodes (except node i and the $i-1$ nodes between node i and homebase) moving to the right of the homebase. If also agent $left_i$ returns, it means that node i contains the black hole because it is the only unexplored node. Hence, the return of the pair $tiebreaker_i$,$left_i$ (or $tiebreaker_i$,$right_i$) signals the presence of the black hole in node i.

Consider now the time complexity of the algorithm. Agent $tiebreaker_i$ begins the execution after $2i$ time units, and it explores all but i nodes. Hence it returns back to the homebase $2(n-1)$ time units after the start of the execution of the first agent. So, when the black hole is one of the $q-1$ nodes closest to the homebase, it will be located within $2(n-1)$ units. A similar argument apply when the blackhole is symmetrically placed on the other (right) half of the ring, and the worst case result follows.

As for the average time complexity, we have two situations: when the black hole is within q nodes of the homebase, the time for locating it is $2(n-1)$; otherwise, it is $\frac{3}{2}(n-2)$ time units. Hence the average ideal time complexity is:

$$\frac{2(q-1)(2(n-1)) + 2(q+1)(\frac{3(n-2)}{2})}{4q} = \frac{7(n-1)}{4} - O(1).$$

4 Optimal Average Time

In this section we show that, by using $2(n-1)$ agents, it is possible to achieve simultaneously optimal time both in the average and in the worst case, establishing a lower bound on the average time complexity of black hole search.

The idea of the algorithm is to identify pairs of agents $(left_i, right_i,\ i \leq 1 \leq n-1)$ among the $2(n-1)$ available, and to assign each pair to "check" a node of the ring. To check node i, an agent of the pair would move to node $i-1$ clockwise (thus exploring nodes $1, 2, \ldots, i-1$) and the other would move to node $i+1$ counterclockwise (thus exploring nodes $i+1, i+2, \ldots n-1$). The presence of the black hole in the ring insures that only one pair will come back to the homebase intact while one agents of each of the other pairs will disappear in the black hole. Once the successful pair returns, the black hole is located.

Theorem 2. *Algorithm* OptAvgTime *solves the black hole location problem. in average ideal time complexity* $\frac{3}{2}n + O(1)$ *and worst case ideal time complexity* $2(n-2)$. *Both complexities are optimal.*

Proof. Correctness follows from the fact that for each node i there are two agents, namely $left_i$ and $right_i$ such that the singleton set $\{i\}$ is the intersection of the areas that they do not explore.

Algorithm 2 Algorithm OPTAVGTIME

$2(n-1)$ co-located agent $left_i, right_i$, $i \leq 1 \leq n-1$ at homebase node 0.

1. Agent $left_i$ explores nodes $(0, 1, 2, \cdots, i-1)$ and returns.
2. Agent $right_i$ explores nodes $(n-1, n-2 \cdots, i+2, i+1)$ and returns.
3. Let ($left_j$, $right_j$) be the only full pair safely returning. The black hole is node j.

Let us now consider worst case time complexity: The time spent by $left_i$ and $right_i$ to reach it and come back is $2Max\{i-1, n-i\}$; the worst case clearly occurs when the black hole is located on node 1 (or $n-2$) and the corresponding time complexity is $2(n-2)$, which is optimal [12].

As for the average time. The presence of the black hole at a node i is witnessed by agents reaching nodes $i-1$ and $n-i-1$. Hence, the ideal time delay for the algorithm when the black hole is located at node i is $2Max\{i-1, n-1-i\}$. $2(i-1)$ is greater than or equal to $2(n-1-i)$ whenever $i \geq \frac{n}{2}$. Since all nodes other than the homebase are equally likely to contain the black hole, the average time complexity is:

$$\frac{\sum_{i=1}^{\frac{n}{2}-1} 2(n-1-i) + \sum_{i=\frac{n}{2}}^{n-1} 2(i-1)}{n-1}$$

$$= \frac{(n-1)(n-2) + \sum_{i=1}^{\frac{n}{2}-1} -2i + \sum_{\frac{n}{2}}^{n-1} 2i}{n-1}$$

$$= \frac{(n-1)(n-2) - (\frac{n}{2}-1)n + (n-1)n}{n-1} = \frac{3}{2}n + O(1)$$

Notice that nodes on either side of the black hole have necessarily to be reached by some agents and their visit reported back. Hence the time when the black hole is located at node i must be greater than or equal to both $2(i-1)$ and $2(n-1-i)$, which precisely corresponds to the time complexity of our algorithm for node i. We can then conclude that our bound is optimal.

5 Optimal Team Size

The algorithm of Dobrev et al [12] as well as the improvements presented here have optimal time complexities both in the worst and in the average case; however they all use $O(n)$ agents, which is order of magnitude larger that than the optimal team size $k = 2$. One might think that this large number of agents used by time-optimal solutions is necessary. This is however not true, as we show in this section.

In the following, we present an algorithm that allows $k = 2$ agents to locate the black hole with asymptotically optimal time in both the worst and the average case. The cost in terms of messages of this algorithm is $O(n^2)$, the same as all the others considered here.

The algorithm, called OptTeamSize, is as follows. At each point in time the nodes of the ring are partitioned into an *Explored* area and an *Unexplored* one. The explored area has been already visited by some agent and it is known to be safe, the unexplored area is still to be visited and contains the black hole. Moreover, during the algorithm, the unexplored area is partitioned between the two agents. More precisely, it is always divided into two disjoint areas of different sizes to which agents are assigned: one part containing a single node and the other containing all other unexplored nodes. In each step of the algorithm one of the agents (called *small*) is given the task to explore the area containing a single node, while the other (called *big*) has to explore the other area. The exploration proceeds with cautious walk. Since the two areas are disjoint, one of the agents will certainly succeed in its exploration. If the *big* agent succeeds, the blackhole is obviously located and the algorithm terminates. On the other hand, if the *small* agent returns successfully, it further divides the remaining unexplored area and notifies the *big* agent of the update by leaving a message on the whiteboard of the last node successfully visited by the other agent. The way the update of the unexplored area is performed is such that an agent stays *small* for *two consecutive steps* before switching role. A stage of the algorithm consists of these two consecutive steps and the algorithm is a sequence of stages which terminates when $n-1$ nodes are known to be safe.

This division process is preceded by a preprocessing phase where the two agents divide the ring in two disjoint parts of almost equal size: only when one of the two returns to the homebase the asymmetric workload division starts to take place.

In the following when we say that an agent acts as *big* we mean that it cautiously explores *all but the last* nodes of the unexplored area. When an agent acts as *small* it cautiously explores the *first* node of the unexplored area. The location of the homebase in the various steps of the algorithms is variable and it is always the central node of the current explored area. An *update* message contains the update information about the current unexplored area and the current location of the homebase.

We now prove that the algorithm terminates correctly and we study its complexity.

Theorem 3. *Algorithm* OptTeamSize *solves the black hole search in optimal time* $\Theta(n)$ *using 2 agents and performing* $O(n^2)$ *moves.*

Proof. After the end of the preprocessing phase at least one agent, say *right*, survives and returns. Since the segments of the ring explored by each agent are always disjoint, at least one agent survives every stage. If the *big* agent survives, the algorithm terminates correctly, if the *small* agent survives the size of the unexplored area decreases by one and the algorithm correctly moves to the next stage (or it terminates if the new size is equal to one). Hence, one of the agents eventually discovers the location of the black hole.

To prove that the algorithm has $\Theta(n)$ time complexity, we first observe that when the exploration phase of the algorithm begins the explored area is at least

Algorithm 3 Algorithm OptTeamSize

Two co-located agents l and r . $E = \{v_h\}$. $U = V - E$.

1. Preprocessing Phase: Agent l (resp. r) explores cautiously the leftmost (resp. rightmost) $\lfloor |U|/2 \rfloor$ nodes of the unexplored area and when finished returns to the homebase v_h.
2. Exploration Phase: One of the agents (say l) arrives at the homebase and becomes *small*. Agent l moves to the last explored node on agent r's side, it leaves an update message to r indicating to act as *big*. Agent l then moves to its side and act as *small*. Stage 1 of Phase 2 begins.
3. Stage i of Phase 2:
   (a) If the *big* agent (say r) returns to the homebase (or the *small* agent returns and the size of the unexplored area is one) then the blackhole is located and the algorithm terminates.
   (b) Otherwise, the *small* agent l returns, it moves to the last explored node on agent r's side, it leaves an update message for r indicating to maintain the same role *big*.
   (c) Agent l moves back to its side and it acts as *small*.
   (d) If r returns, then the blackhole is located and the algorithm terminates.
   (e) Otherwise agent l returns, it moves to the last explored node on agent r's side, it leaves an update message for r instructing to reverse role. Agent l then moves to the other side and it changes role acting as *big*.
   (f) If l returns the blackhole is located and the algorithm terminates.
   (g) Otherwise Agent r returns and becomes *small*; it moves to agent l's side, it leaves an update message for agent l instructing it to act as *big*. Agent r then moves back to its side and acts as *small*.
   (h) If l returns then the blackhole is located and the algorithm terminates.
   (i) Otherwise agent r returns, it changes role becoming *big*, it moves to agent l's side and it leaves a message to agent l at the last explored node updating the unexplored area and instructing to reverse roles. Agent l moves to its side and acts as *big*.
   (j) Stage $i + 1$ starts.

of size $\frac{n-1}{2}$. While the *big* agent, say r, is exploring all but one nodes on its side, the other agent (if it did not disappear in the blackhole before) performs two steps as *small* making at least $\frac{3}{2}(n-1)$ moves (and spending the same amount of time). By that time, under the ideal time assumption, agent r would have either i) returned safely determining the location of the black hole, or ii) died in the black hole. In the first case we obviously have a time complexity of $O(n)$. In the other case, when agent l switches role becoming *big* and moves to explore all but one nodes, it necessarily completes its task locating the black hole, again with an overall time complexity of $O(n)$.

To show that the worst case move complexity is $O(n^2)$, it suffices to notice that in the worst possible asynchronous execution it is always the *small* agent that completes a step, while the big agent is slow on a link. Since the *small* agent manages to explore a single node in each step, and the size of the unexplored area when this procedure starts is $\frac{n}{2}$, $O(n)$ steps are necessary to locate the black hole. In each step however $O(n)$ moves are performed by the *small* agent to explore and report the update on the other side of the explored area, for a total of $O(n^2)$ moves.

We now show the exact average and worst case time complexities of the algorithm.

Theorem 4. *Algorithm* OptTeamSize *solves the black hole search in average ideal time* $\frac{15}{2}n + O(1)$ *and worst case* $8n + O(1)$.

Proof. By symmetry of the algorithm we may assume that the black hole is located on the right half of the ring (w.l.g let n be even). We then calculate the ideal time delay when the black hole is located at node $i \leq \frac{n}{2}$ (i.e., $\frac{n}{2}$ nodes to the right of the homebase). We consider different cases.

- *Case 1*: Node i is the border node of the partition between the right and the left agents. In this case the left agent returns after $\frac{3}{2}n + \frac{n}{2}$ time units to the homebase. In the sum, the first addend is for the cautious exploration and the second is for the time taken to return to the homebase. Now the left agents follows the path of the right agent. The right agent must have died at the last node of its partition. So in another n time units the left agent will reach the last safe node explored by the right agent and return. So, in this case the black hole is located in $3n$ total time units.
- *Case 2*: Node i is the neighbor of the border node of the partition between the right and the left agents. Similarly to the previous case, the left agent will take $3n - O(1)$ time units to return to the homebase after exploring all nodes, except the black hole and its neighbor. Now the left agent in the role of *small* explores the last safe node and return in further $n+O(1)$ time units. In this case the black hole is then located in $4n + O(1)$ time units.
- *Case 3*: Node i is the third node from the border of the partition between the right and the left agents. In this case the left agent discovers the black hole after the end of the second round as *small*. The total ideal time delay in this case is $5n + O(1)$.

– *Case 4*: Node i is the fourth node or the node further from the border of the partition between the right and the left agents. In this case the left agent (l) performs two rounds as *small* and a round as *big*. Observe that under ideal conditions the right agent would die in the black hole and would not return, so it suffices to count the time taken by the left agent. Agent l explores $(n-i)+O(1)$ nodes in total and this cautious exploration costs in total $3(n-i)+O(1)$ time units. Let us now compute the time necessary for the other movements of the left agent. Agent l takes $\frac{n}{2}+i+O(1)$ time units for reaching the last safe node explored by the right agent. Moreover, agent l takes $4(\frac{n}{2}+i)+O(1)$ time unit for the two rounds as *small*; after becoming *big*, agent l reaches the last explored node on its side in $\frac{n}{2}+i$ time units. At this point it cautiously explores (the cost of the exploration has been already accounted for earlier). Finally, the agent returns to the homebase in $n-i+O(1)$ time units. Thus the total ideal time delay is $7n+2i+O(1)$ time units.

Hence the average ideal time delay is :

$$\frac{12n+\sum_{i=1}^{i=\frac{n}{2}}(7n+2i)}{\frac{n}{2}} = 7n+\frac{(\frac{n}{2})(\frac{n}{2}+1)}{\frac{n}{2}} = \frac{15}{2}n+O(1)$$

The worst case occurs in correspondence of Case 4, when $i = n/2 - O(1)$, which yields $8n+O(1)$.

References

1. M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176(1):1–21, 2002.
2. J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. In *21st International Symposium on Distributed Computing (DISC)*, pages 108–122, 2007.
3. C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *10th International Conference on Principles of Distributed Systems (OPODIS)*, pages 320–332, 2006.
4. J. Czyzowicz, S. Dobrev, R. Královic, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In *16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 182–194, 2009.
5. J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2–3):229–242, 2006.
6. J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability & Computing*, 16(4):595–619, 2007.
7. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *J. Graph Theory*, 32(3):265–297, 1999.
8. S. Dobrev, P. Flocchini, R. Kralovic, P. Ruzicka, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, 2006.

9. S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring an unknown graph to locate a black hole using tokens. In *5th IFIP Int. Conference on Theoretical Computer Science(TCS)*, pages 131–150, 2006.
10. 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.
11. S. Dobrev, P. Flocchini, and N. Santoro. Improved bounds for optimal black hole search with a network map. In *10th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 111–122, 2004.
12. S. Dobrev, P. Flocchini, and N. Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90, 2007.
13. S. Dobrev, N. Santoro, and W. Shi. Using scattered mobile agents to locate a black hole in an un-oriented ring with tokens. *Int. J. Found. Comput. S.*, 19(6):1355 – 1372, 2008.
14. P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *22nd International Symposium on Distributed Computing (DISC)*, pages 227–241, 2008.
15. P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *24th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–10, 2009.
16. P. Flocchini and N. Santoro. Distributed Security Algorithms For Mobile Agents. In J. Cao and S. K. Das, editors, *Mobile Agents in Networking and Distributed Computing*, chapter 3. Wiley, 2009.
17. P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2–3):331–344, 2005.
18. J. A. Garay, S. Kutten, and D. Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, 27(1):302–316, 1998.
19. P. Glaus. Locating a black hole without the knowledge of incoming link. In *5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 128–138, 2009.
20. R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and approximation results for black hole search in arbitrary networks. *Theoretical Computer Science*, 384(2-3):201–221, 2007.
21. R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Approximation bounds for black hole search problems. *Networks*, 52(4):216–226, 2008.
22. R. Královic and S. Miklík. Periodic data retrieval problem in rings containing a malicious host. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), to appear*, 2010.
23. S. Kutten and D. Peleg. Fast distributed construction of small k-dominating sets and applications. *Journal of Algorithms*, 26(1):40–66, 1998.
24. W. Shi. Black hole search with tokens in interconnected networks. In *11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 670–682, 2009.