Mobile Search for a Black Hole in an Anonymous Ring *

Stefan Dobrev † Paola Flocchini‡ Giuseppe Prencipe§
Nicola Santoro¶

Abstract

In this paper we address the problem of mobile agents searching for a highly harmful item (called *black hole*) in a ring network. The black hole is a stationary process that destroys visiting agents upon their arrival without leaving any observable trace of such a destruction. The task is to have at least one surviving agent able to unambiguously report the location of the black hole.

We consider different scenarios and in each situation we answer some computational as well as complexity questions. We first consider agents that start from the same homebase (co-located). We prove that *two* such agents are necessary and sufficient to locate the black hole; in our algorithm the agents perform $O(nlogn)$ moves (where n is the size of the ring) and we show that such a bound is optimal. We also consider time complexity and we show how to achieve the optimal bound of $2n-4$ units of time using $n-1$ agents. We generalize our technique to establish a *trade-off* between time and number of agents. We then consider the case of agents that start from different homebases (dispersed) and we show that, if the ring is *oriented*, *two* dispersed agents

*A preliminary version of this paper appeared in the Proceedings of the 15th International Symposium on Distributed Computing [8].

†University of Ottawa, email:sdobrev@site.uottawa.ca

‡University of Ottawa, email:flocchin@site.uottawa.ca

§contact author: Università di Pisa, Dipartimento di Informatica, Via Buonarroti, 2 – 56100, Pisa, tel. +39 050 2213148, fax. +39 050 2212726, email:prencipe@di.unipi.it

¶Carleton University, email: santoro@scs.carleton.ca

are still necessary and sufficient. Also in this case our algorithm is optimal in terms of number of moves ($\Theta(nlogn)$). We finally show that, if the ring is *unoriented*, *three* agents are necessary and sufficient; an optimal algorithm follows from the oriented case.

Keywords: Mobile Agents, Distributed Computing, Ring Network, Hazardous Search.

1 Introduction

1.1 Mobile Agents and Black Hole

The most widespread use of autonomous mobile agents in network environments, from the World-Wide-Web to the Data Grid, is clearly to *search*, i.e., to locate some required "item" (e.g., information, resource, ...) in the environment. This process is started with the specification of what must be found and ends with the reporting of where it is located.

The proposed solutions integrate their algorithmic strategies with an exploitation of the capabilities of the network environment; so, not surprising, they are varied in nature, style, applicability and performance. They do however share the same assumption about the "item" to be located by the agents: it poses no danger, it is *harmless*.

This assumption unfortunately does not always hold: the item could be a local program which severely damages visiting agents. In fact, protecting an agent from "host attacks" (i.e., harmful items stored at the visited site) has become a pressing security concern (e.g., see [5, 11, 13, 15, 17, 18, 20]). Still, this problem has not been taken into account so far by any of the existing solutions.

In this paper we address the problem of searching for a highly harmful item whose existence we are aware of, but whose whereabouts are unknown. The item is a stationary process which disposes of visiting agents upon their arrival; no observable trace of such a destruction will be evident. Because of its nature, we shall call such an item a *black hole*.

The task is to unambiguously determine and report the location of the black hole (following this phase, a "rescue" activity would conceivably be initiated to deal with such a destructive process).

We are interested in understanding the basic algorithmic limitations and factors.

The setting we consider is the simplest symmetric topology: the anonymous ring (i.e., a loop network of identical nodes). In this setting operate mobile agents: the agents have limited computing capabilities and bounded storage[1], obey the same set of behavioral rules (the "protocol"), and can move from node to neighboring node. We make no assumptions on the amount of time required by an agent's actions (e.g., computation, movement, etc) except that it is finite; thus, the agents are *asynchronous*. Each node has a bounded amount of storage, called *whiteboard*; $O(\log n)$ bits suffice for all our algorithms. Agents communicate by reading from and writing on the whiteboards; access to a whiteboard is done in mutual exclusion. The problem is solved if at least one agent survives, and all surviving agents know the location of the black hole.

Some basic *computational* questions naturally and immediately arise, such as: How many agents are needed to locate the black hole? How many suffice? What a-priori knowledge is required? as well as *complexity* questions, such as: With how many moves can the agents do it? How long does it take? How many agents will disappear in the black hole ?

In this paper, we provide definite answers to each of these questions. Some answers follow from simple facts. For example, if the ring size n is not known, then the black-hole search problem can not be solved[2]. Another fact is that at least one agent will disappear in the black hole, and thus at least two agents are needed to solve the problem.

1.2 Major Contributions

We consider the problem in two settings: when the anonymous agents are *co-located* (i.e., start from the same node), and when they are *dispersed*(i.e., start from different nodes).

We first consider *co-located* agents. We establish a *lower bound* on the number of moves required by any solution protocol: we prove that $(n-1)\log(n-1)+O(n)$ moves are needed *regardless* of the number of co-located agents.

We then show that *two* agents are both *necessary and sufficient* to locate the black hole. Sufficiency is proved constructively. We present a distributed algorithm that allows two

[1] $O(\log n)$ bits suffice for all our algorithms.

[2] i.e., no deterministic protocol exists which always correctly terminates.

agents to locate the black hole. This algorithm is *optimal*, within a factor of two, also in terms of the *amount of moves* performed by the two agents.

These results show that the presence of more than two agents is not helpful in reducing the number of moves. It can however be useful in reducing the time spent by co-located agents to locate the black hole.

We prove that $2n-4$ ideal time units are needed, *regardless* of the number of agents; we then describe how to achieve such a time bound using $n-1$ agents. We generalize this technique and establish a *general trade-off* between time and number of agents.

We then consider *dispersed* agents. We first establish a lower bound, proving that any solution with k anonymous dispersed agents requires $\Omega(n \log(n-k))$ moves, provided k is known; if k is unknown, $\Omega(n \log n)$ moves are always required.

The number of dispersed agents required to solve the problem actually depends on whether the ring is *oriented* or not.

If the ring is *oriented*, *two* anonymous dispersed agents are both *necessary and sufficient* to locate the black hole. In fact, we present a protocol that allows two (or more) anonymous dispersed agents to find the black hole; This distributed algorithm uses $O(n \log n)$ moves, regardless of the number of agents; thus, it is *optimal* also in terms of *number of moves*.

On the other hand, if the ring is *unoriented*, *three* anonymous dispersed *agents* are *both necessary and sufficient*. Sufficiency follows constructively from the result for oriented rings.

For the *general* setting, where the agents are neither all co-located nor fully dispersed, simple modifications to the protocol for the dispersed agent settings provide a solution with the same complexity.

1.3 Related Work

Recently, algorithmic questions have been raised on the use of autonomous mobile agents to compute in networked environments. In particular, a central question is to determine what minimal hypotheses allow a given problem $\mathcal{P}$ to be solved. Examples of this type of investigations are the studies on topology-reconstruction (e.g., see [3, 4]), graph-exploration [6, 16], wake-up [2, 9], rendez-vous[1, 19], etc.

The topic of hazardous search has been recently considered in [14, 12], where a single agent searches a network in presence of probably incorrect "routing" information at each node.

In the distributed computing literature (i.e., with *static* agents), there have been many studies on computing in presence of undetectable faulty components (e.g., [7, 10]). The problem studied here has never been investigated before.

2 Basic Results and Lower Bound

2.1 Notation

The network environment is ring $\mathcal{R}$ of n *anonymous* (i.e., identical nodes. Each node has two ports, labelled *left* and *right*; if this labelling is globally consistent, the ring will be said to be *oriented*, *unoriented* otherwise. Each node has a bounded amount of storage, called *whiteboard*; $O(\log n)$ bits suffice for all our algorithms. Let $0, 1, \ldots n-1$ be the nodes of the ring in clockwise direction ($0, -1, \ldots -(n-1)$) in counter-clockwise direction) and, without loss of generality, let us assume that node 0 is the *home base*.

In this network there is a set A of *anonymous* (i.e., identical) mobile agents. Let $|A| = k$ denote the number of agents. The agents can move from node to neighboring node in $\mathcal{R}$, have computing capabilities and bounded storage, obey the same set of behavioral rules (the "protocol"), and all their actions (e.g., computation, movement, etc) take a finite but otherwise unpredictable amount of time (i.e., they are *asynchronous*). Agents communicate by reading from and writing on the whiteboards; access to a whiteboard is done in mutual exclusion. We will consider two settings: *co-located* agents (i.e., starting from the same node called the *home base*), and *dispersed* agents (i.e., each starting from a different home base).

A *black hole* (Bh) is a stationary process located at a node, which destroys any agent arriving at that node; no observable trace of such a destruction will be evident to the other agents. The location of the black hole is unknown to the agents. The Black Hole Search (shortly Bhs) problem is to *find the location of the black hole*. More precisely, Bhs is solved if at least one agent survives, and all surviving agents know the location of the black hole within finite time (*explicit termination*). Our lower bounds are actually established requiring

only that at least one surviving agent knows the location of the black hole (the difference is only $O(n)$ moves/time).

Let us now introduce the cost measures. Our main measure of complexity is the *number of agents*, called *size*, used by the protocol. The other important cost measure is the total *number of moves* performed by the agents, which we shall call *cost*. We will also consider the amount of *time* elapsed until termination. Since the agents are asynchronous, "real" time cannot be measured. We will use the traditional measure of *ideal time* (i.e., assuming synchronous execution where a move can be made in one time unit); sometimes we will also consider *bounded delay* (i.e., assuming an execution where a move requires at most one time unit), and *causal time* (i.e., the length of the longest, over all possible executions, chain of causally related moves). In the following, unless otherwise specified, "time" complexity is "ideal time" complexity.

2.2 Cautious Walk

Let us introduce a basic tool that we will employ in the protocols.

At any time during the search for the black hole, the ports (corresponding to the incident links) of a node can be classified as (a) *unexplored* – if no agent has moved across this port, (b) *safe* – if an agent arrived via this port or (c) *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*.

The technique we use, called *cautious walk*, is defined by the following two rules:

Cautious Walk

Rule 1. when an agent moves from node u to v via an *unexplored* port (turning it into *active*), it immediately returns to u (making the port *safe*), and only then go back to v to resume its execution;

Rule 2. no agent leaves via an *active* port.

In the following, unless otherwise specified, when an agent moves, it will move using

cautious walk.

2.3 Basic Limitations and Bounds

First of all notice that, if there is only one agent, the BHS problem is obviously unsolvable because the only agent would necessarily disappear in the black hole; that is

Lemma 1 *At least two agents are needed to locate the black hole.*

Thus, we assume that there are at least two agents.

Further observe that, the combined presence of asynchrony and undetectability of the presence of black hole make it impossible to distinguish between a "slow" link and a link leading to the black hole. As a consequence we have that:

Lemma 2 *It is impossible to find the black hole if the size of the ring is not known.*

Proof. By contradiction, assume A is a solution protocol not requiring knowledge of the ring size. Consider a synchronous execution of A on a ring $R = x_0, \ldots x_{n-1}$ where $k \geq 2$ agents are initially colocated in x_0, and x_b is the black hole. After a finite time t all surviving agents know the location of the black hole x_b. Consider now a ring $R' = x_0, \ldots x_{b-1} u, z, v, x_{b+1}, \ldots x_{n-1}$ where x_0 is the home base and z is the black hole. Consider now the same synchronous execution of A on R' except that any movement from x_{b-1} to u and from x_{b+1} to v will take more than t time units. All agents, which moved towards x_b from x_{b-1} or x_{b+1} and were destroyed there in the execution in R, will now move towards u or v. However, they will not arrive there until after time t. Every other agent will experience for the first t time units exactly the same execution in R' as in R; hence, it will incorrectly report that the links incident to the black hole are those from u and v. □

We now consider the minimum number of moves required to solve the problem. We show that, regardless of the setting (i.e., co-location or dispersal) and of the number of agents employed, $(n-1)\log(n-1) + O(n)$ moves are required.

In the following, we will denote by E^t and U^t the sets of explored and unexplored nodes at time t, respectively; clearly, each such set (which we will also call "area") is a contiguous section of the ring. Let x^t and y^t be the two *border nodes* in E^t (i.e., the nodes that connect

E^t to U^t), and let z^t be the central node of E^t; that is, z^t is the node in E^t at distance $\lceil |E^t|/2 \rceil - 1$ from x^t.

We say that a *causal chain* from node v_p to node v_q *has been executed* at time t, if $\exists d \in \mathbb{N}, \exists u_1, u_2, \ldots, u_d \in V$ and times $t < t_1 < t'_1 < t_2 < t'_2 < \ldots < t_d < t'_d$ such that $v_p = u_1$, $v_q = u_d$, u_i is a neighbor of u_{i+1}, and an agent leaves u_i at time t_i and reaches u_{i+1} at time t'_i $(1 \leq i \leq d-1)$.

Lemma 3 *Let $|U^t| > 2$ at a given time $t \geq 0$.*

1. *Within finite time, at time $t' \geq t$, at least two agents will have left the explored area E^t in different directions.*

2. *A finite time after they have left E^t, say at $t'' > t'$, a* causal chain *is executed from one of the two border nodes of $E_{t''}$ to z_t.*

Proof. Let $\mathcal{P}$ be a BH solution protocol using k agents. Consider an execution of $\mathcal{P}$ in $\mathcal{R}$.

1. If no agent ever leaves E^t, then (since $|U^t| > 2$) the black hole is never discovered, contradicting the fact that $\mathcal{P}$ is a solution protocol.

   If all the agents that leave E^t, leave it in the same direction, let the black hole be on the first node of U^t explored by the agents that leave E^t; in this case, all these agents are destroyed by the BH right after leaving E^t, while the other agents keep moving inside E^t (they never leave E^t by hypothesis); hence the black hole is never discovered. Again a contradiction.

   Therefore, there exists a time $t' \geq t$ when at least two agents have left E^t in different directions, and the first part of the lemma follows.

2. Let us assume by contradiction that, for every $t'' > t'$ there exists no *causal chain* from any of the two border nodes of $E_{t''}$ to z_t. Since no *causal chain* is ever executed after t', the agents on the right side of z_t will not communicate with the agents on the left side of z_t through E^t. Because of the presence of the black hole, they cannot communicate through U^t either. However, to terminate, an agent must have information about all nodes in U^t (but one, the black hole). Let us place the black hole on the central node

of U^t. Since $|U^t| > 2$, every agent will not have information about at least two nodes; hence termination is impossible: a contradiction.

□

Theorem 1 *At least* $(n-1)\log(n-1) + O(n)$ *moves are needed to find a black hole in a ring, regardless of the number of agents.*

Proof. We will describe an adversary (scheduler) *Adv* that, given an arbitrary solution protocol $\mathcal{P}$, will force an execution of $\mathcal{P}$ in which the agents will perform $\Omega(n \log n)$ moves.

The adversary has the power to:
(i) *block* a port - the corresponding link becomes very slow, effectively blocking all the agents traversing it;
(ii) *unblock* a blocked port - the agents in transit on the corresponding link will then arrive to their destination;
(iii) choose which agents will move, if there is a choice;
(iv) decide where the black hole is located.

Any agent exiting through or in transit on a blocked port will be said to be *blocked*. The Adversary can block any port at any time; however, within finite time, it will unblock any blocked port not leading to the black hole.

The adversary works in stages.

By Lemma 3.(1), a finite time after the agents start the execution, at least two agents will leave E_0 (constituted only by the starting node h) in different directions; *Adv* blocks both ports leaving the explored area until there is at least one agent blocked in each direction: Stage 1 starts when this happens.

Adv will now choose one of the two blocked ports, and unblock it. If *Adv* unblocks a port (e.g., the one on the left of h), by Lemma 3.(2), within finite time, a *causal chain* between the border node on the left side of the explored area and z_0 is executed. Let $t' > t$ be the time when this happens. If $|U^{t'}| \leq 2$, then *Adv* will allow the algorithm to locate the black hole within $O(n)$ moves and the protocol to terminate. Otherwise, *Adv* blocks both the ports leaving $E_{t'}$ until at least one agent is blocked on each link (by Lemma 3.(1), this must happen within finite time); the next stage will then start.

In general, stage i starts when at least two agents leave the area explored in stage $i-1$ in opposite directions after a *causal chain* between one border node of the explored area and its central node has been executed. In the following, we will indicate the explored and unexplored area at the beginning of stage i by E_i and U_i respectively, and by x_i and y_i the leftmost and rightmost border node of E_i respectively.

At the beginning of each stage i, *Adv* performs two "virtual executions" of $\mathcal{P}$, to decide which port to unblock. By definition of beginning of a stage, at least one agent must be on x_i and at least one other on y_i, heading towards the unexplored area; let l and r be the port connecting x_i to U_i, and y_i to U_i, respectively. Let us indicate by x_{i+1} (resp., y_{i+1}), x_{i+2} (resp., y_{i+2}), ..., the nodes in the chain starting in x_i (resp., y_i) that belong to the unexplored area. In the first virtual execution, *Adv* blocks l, unblocks r, and blocks the port connecting y_{i+1} to y_{i+2}; in other words, *Adv* allows at most one node, y_{i+1}, to be explored. At this point, if no *causal chain* between y_{i+1} and z_t is executed, *Adv* unblocks the port connecting y_{i+1} to y_{i+2}, blocks the port connecting y_{i+2} to y_{i+3}, and checks again if a *causal chain* is executed. The adversary keeps unblocking one port at a time, until a *causal chain* is executed between z_t and the node at the right border between the explored and unexplored area (by Lemma 3.(2), since l is blocked and $\mathcal{P}$ is a deterministic solution, this must happen within finite time). Let R_i be the set of nodes explored after y_i and before the *causal chain* is executed. Analogously, in the second virtual execution, *Adv* blocks r and unblocks l, and reiterates the process of unblocking the ports on the left side of the ring, one at a time; let L_i be the set of new nodes explored before the *causal chain* is executed. *Adv* decides which virtual execution to actually perform, based on the sizes of L_i and R_i. Namely, if $|L_i| \leq |R_i|$, then l is unblocked, while r is kept blocked; otherwise, r is unblocked, and l is kept blocked. Clearly

$$|L_i| + |R_i| \leq |U_i|, \tag{1}$$

otherwise $L_i \bigcap R_i \neq \emptyset$ and *Adv* can put the black hole on a node in the intersection of L_i and R_i. From the definition of L_i and R_i it follows that, for a *causal chain* to be executed, one of the agents must explore the whole L_i (or R_i). If the black hole is in the intersection, no *causal chain* would be executed from any border vertex, contradicting point (2) of Lemma 3.

If l is unblocked in stage i, at least $|L_i|$ moves to explore new nodes are executed; and at least $|L_i| + \lceil |E_i|/2 \rceil - 1$ moves are performed to reach the central node of E_i. Thus, at least $2|L_i| + |E_i|/2 - 1$ moves are executed. The same happens, if r is allowed to move instead. Denote by $W = w_1, \ldots, w_s$ the sizes of the newly explored regions at each stage, where s is the total number of stages performed by Adv until $\mathcal{P}$ terminates; we have that

$$\Sigma_{i=1}^{s} w_i = n - 1, \tag{2}$$

and $|E_i| = \Sigma_{j=0}^{i-1} w_j$, where $w_0 = |E_1| = 1$; hence the number of movements performed in stage i, $1 \leq i \leq s$ is at least

$$2w_i + \Sigma_{j=0}^{i-1} w_j/2 - 1 > w_i + \Sigma_{j=0}^{i} w_j/2 - 1. \tag{3}$$

Since only the agents that explore less are allowed to move by Adv, from (1) we have that

$$w_i \leq |U_i|/2 = (\Sigma_{j \geq i} w_j)/2;$$

hence,

$$w_i \leq \Sigma_{j=i+1}^{s} w_j. \tag{4}$$

From (2) and (3), the total number $Cost(\mathcal{P})$ of moves performed during the execution of $\mathcal{P}$ is at least

$$Cost(\mathcal{P}) \geq \Sigma_{i=1}^{s}(w_i + \Sigma_{j=0}^{i} w_j/2 - 1) = (n - 1 - s) + \Sigma_{i=1}^{s}\Sigma_{j=0}^{i} w_j/2 = (n - 1 - s/2) + c(W)/2$$

where $c(W) = \Sigma_{i=1}^{s} i w_{s-i+1}$.

Let us assume that Adv executed the first $i - 1$ stages. Clearly, from (1), at least $\log(|U_i|) = \log(\Sigma_{j \geq i} w_j)$ stages are still necessary to terminate the execution of $\mathcal{P}$. Moreover, since the number of stages still to come after stage $i-1$ is $s-i+1$, we have that $\log(\Sigma_{j \geq i} w_j) \leq s - i + 1$; hence,

$$\Sigma_{j=i}^{s} w_j \leq 2^{s-i+1}. \tag{5}$$

Therefore, $w_s \leq 2$; from (4), $w_{s-1} \leq w_s \leq 2$, $w_{s-2} \leq w_{s-1} + w_s \leq 4$; and in general $w_{s-i} \leq 2^i$, $1 \leq i \leq s - 1$. Hence, we have (from (2))

$$2 + \Sigma_{i=1}^{s-1} 2^i \geq w_s + \Sigma_{i=1}^{s-1} w_{s-i} = n - 1,$$

and we can conclude that there are at least $s \geq \log(n-1)$ stages.

Let $\overline{s} = \log(n-1)$ and let $\overline{W} = \overline{w}_1, \ldots, \overline{w}_{\overline{s}}$ where $\overline{w}_{\overline{s}-i} = 2^i$, $1 \leq i \leq \overline{s}-1$, and $\overline{w}_{\overline{s}} = 2$. Informally, $\overline{s}$ represents the achievable minimum number of stages and $\overline{w}_{\overline{s}-i}$ is the maximum size of the region explored in round i in the corresponding $\overline{s}$-stage execution. That is, at each stage (looking backwards from the last stage $\overline{s}$), the algorithm lets the agents explore as much as they are allowed to. We will prove that $\overline{W}$ gives the minimum number of moves; that is, there exists no W such that $c(W) < c(\overline{W})$.

We have

$$c(W) = w_s + 2w_{s-1} + \ldots + \overline{s}w_i + (\overline{s}+1)w_{i-1} + \ldots + sw_1 =$$

$$\Sigma_{i=1}^{\overline{s}} i w_{s-i+1} + \Sigma_{i=\overline{s}+1}^{s} i w_{s-i+1},$$

and

$$c(\overline{W}) = \overline{w}_{\overline{s}} + 2\overline{w}_{\overline{s}-1} + \ldots + \overline{s} \cdot \overline{w}_1.$$

Since equation (5) is valid for both W and $\overline{W}$, and from the way we choose $\overline{W}$, it follows that $w_{s-i+1} \leq \overline{w}_{\overline{s}-i+1}$, for all $1 \leq i \leq \overline{s}$. Moreover, since from (2), $\Sigma_{i=1}^{s} w_i = \Sigma_{i=1}^{\overline{s}} \overline{w}_i = n-1$, it must be that

$$\Sigma_{i=\overline{s}+1}^{s} w_{s-i+1} = \Sigma_{i=1}^{\overline{s}} (\overline{w}_{\overline{s}-i+1} - w_{s-i+1}).$$

Hence,

$$c(W) > \Sigma_{i=1}^{\overline{s}} i w_{s-i+1} + \Sigma_{i=1}^{\overline{s}} i (\overline{w}_{\overline{s}-i+1} - w_{s-i+1}) = c(\overline{W}).$$

Therefore, $\overline{W}$ gives the solution with the minimum number of moves; that is

$$Cost(\mathcal{P}) \geq n - 1 - \frac{1}{2}\log(n-1) + c(\overline{W})/2 = n - 1 - \frac{1}{2}\log(n-1) + \frac{1}{2}\Sigma_{i=1}^{\overline{s}} i \overline{w}_{\overline{s}-i+1} =$$

$$= (n-1)\log(n-1) + O(n)$$

□

3 BH Search by Co-located Agents

In this section, we consider the case when the agents are co-located; i.e., they start at the same node h (the *home base*). Since access to the whiteboard is done in mutual exclusion, the anonymous agents can easily acquire distinct identities by the order in which they access the whiteboard (e.g., the first agent will create a counter set to 1 in the whiteboard, and assume that identity; subsequently, an anonymous agent accessing the whiteboard will increase the counter and take its value as its identity). Thus, we can assume w.l.g. that the agents have distinct identities.

The distinct identities of the agents allow any tie to be deterministically broken. As a consequence, even if the ring is unoriented, the agents can agree on the clockwise direction of the ring. Thus, in the rest of this section, we assume w.l.g. that the ring is oriented.

Recall that $0, 1, \ldots n-1$ are the nodes of the ring in clockwise direction ($0, -1, \ldots -(n-1)$) in counter-clockwise direction); furthermore, without loss of generality, let us assume that node 0 is the *home base*.

3.1 Size- and Cost- Optimal Solution

At least two agents are needed to locate the black hole (Lemma 1), and we have seen that, regardless of how many agents are employed, $\Omega(nlogn)$ moves need to be performed.

We will now prove that *two co-located agents suffice* to solve the problem; furthermore, the two agents can locate the back hole with *minimal cost*.

The algorithm proceeds in phases. Let E_i and U_i denote the explored and unexplored nodes in phase i, respectively. Clearly, E_i and U_i partition the ring into two connected subgraphs, with the black hole located somewhere in U_i. The idea is to divide the unexplored part of the ring between the two agents, assigning to each a region of almost equal size. Each agent starts the exploration of the assigned part. Because of the existence of the BH, only one of them will complete the exploration. When this happens, it will go through the explored part, until it reaches the last safe link visited by the other agent. It will then again partition the unexplored area in two parts of almost equal size, leave a message for the other agent (in case it is not in the BH), and go to explore the newly assigned area.

Algorithm 1 DIVIDE

Start with round number $i = 1$, $E_1 = \{0\}$, and $U_1 = \{1, 2, \ldots, n-1\}$.

1. Divide U_i into two continuous disjoint parts U_i^l and U_i^r of almost equal sizes. Since U_i is a path, this is always possible. We may assume U_i^l is to the left of 0 while U_i^r is to the right.

2. Let agents l and r explore (using *Cautious Walk*) U_i^l and U_i^r, respectively. Note that, since both of them are within E_i and since U_i is divided into two continuous parts, the agents can safely reach the parts they have to explore.

3. Since U_i^l and U_i^r are disjoint, at most one of them contains the black hole; hence, one of the agents (w.l.g. assume r) successfully completes Step 2. Agent r then moves across E_i and follows the *safe* ports of U_i^l until it comes to the node v from which there is no *safe* port leading to the left.

4. Denote by U_{i+1} the remaining unexplored area. All nodes to the right of v, up to the last node of U_i^r explored by r, are now explored – they form E_{i+1}. If $|U_{i+1}| = 1$, agent r knows that the black hole is in the single unexplored node and terminates. Otherwise, U_{i+1} is divided into U_{i+1}^l and U_{i+1}^r as in Step 1. Agent r leaves on the whiteboard of v a message for l indicating the two areas U_{i+1}^l and U_{i+1}^r and the new stage number $i+1$ (overwriting any previous message). Note that $O(\log n)$ bits are sufficient to code this message.

5. Agent r proceeds to round $i+1$: traverses E_{i+1} and goes to Step 2.

6. When (if) l returns to v, it finds the message and proceeds to round $i+1$.

Theorem 2 *Algorithm* DIVIDE *lets two co-located agents find the black hole with* $2n \log n + O(n)$ *moves.*

Proof. In each phase, one of the two agents completes the exploration of its part, and will be called the *traveling* agent; the other can not complete the exploration of its part (since it contains the black hole), and will be called *blocked*. Notice that the *blocked* agent might have explored some of its assigned area.

Let e_i be the size of the total explored area up to phase i. Let w_i be the total work (i.e., number of moves) done by the two agents during phase i. Initially, $e_i = 1$ and $w_i = 0$. Let p_i be the number of links traversed by the *blocked* agent of phase i.

From the description of the algorithm, it follows that the size of the explored area during phase i is equal to half of the size of the unexplored area of the previous phase, plus the new area explored by the current *blocked* agent. Thus, the total area explored at the end of phase i is: $e_i = \lceil \frac{1}{2}(n - e_{i-1}) \rceil + p_i + e_{i-1} \leq \frac{1}{2}(n + e_{i-1}) + p_i + 1$. Moreover, we also have that $w_i = 2e_i$.

Combining the two expressions, we obtain that $\frac{1}{2}w_i \leq \frac{1}{2}(n + \frac{1}{2}w_{i-1}) + p_i + 1$, thus $w_i \leq n + \frac{1}{2}w_{i-1} + 2p_i + 2$.

So, we have:

$$\begin{aligned} w_0 &= 0 \\ w_1 &\leq n + \tfrac{1}{2}w_0 + 2p_1 + 2 \\ w_2 &\leq n + \tfrac{1}{2}w_1 + 2p_2 + 2 \\ &\dots \\ w_s &\leq n + \tfrac{1}{2}w_{s-1} + 2p_s + 2. \end{aligned}$$

Summing up, it follows that

$$\sum_{j=0}^{s} w_i \leq sn + \frac{1}{2}\sum_{j=0}^{s-1} w_j + 2\sum_{j=1}^{s}(p_j + 1).$$

Thus,

$$\sum_{j=0}^{s} w_i - \frac{1}{2}\sum_{j=0}^{s-1} w_j \leq sn + 2\sum_{j=1}^{s}(p_j + 1)$$

and, hence,

$$\frac{1}{2}\sum_{j=0}^{s} w_j \leq sn + 2\sum_{j=1}^{s}(p_j + 1) - w_s/2$$

Since the work done at the last phase s is at most $2(n-1)$, we obtain:

$$\sum_{j=0}^{s} w_j \leq 2sn + 4\sum_{j=1}^{s}(p_j + 1)) - 2(n-1)$$

Since, as observed before, $s \leq \lceil \log n \rceil$ and $\sum_{j=1}^{s}(p_j + 1) < \frac{n}{2}$, we have that

$$\sum_{j=0}^{s} w_j \leq 2n \log n + O(n)$$

. □

From Lemma 1, Theorems 1 and 2, it follows that

Theorem 3 *Algorithm* DIVIDE *is size-optimal and cost-optimal.*

It is easy to see that the time complexity of Algorithm DIVIDE is also $2n \log n + O(n)$.

3.2 More Than Two Agents: Improving the Time

In this section we study the effects of having $k > 2$ agents in the *home base*. We know that increasing k will not result in a decrease of the total number of moves. In fact, the lower bound of Theorem 1 is independent of the number of agents, and is already achieved, within a factor of two, by $k = 2$. The availability of more agents can be exploited to improve the *time complexity* of locating the black hole.

The following theorem shows a simple lower bound on the time needed to find the black hole, regardless of the number of agents in the system.

Theorem 4 *In the worst case,* $2n-4$ *time units are needed to find the black hole, regardless of the number of agents available.*

Proof. Let the black hole be positioned on node $n-1$. For the agents to report the position of the black hole to node 0, it is necessary to receive some information from node $n-2$. It takes $n-2$ steps for an agent to reach node $n-2$ and another $n-2$ steps are needed to report that back to node 0, for a total of $2n-4$ steps. □

Algorithm 2 OPTTIME

Let $r_1 \ldots r_{n-1}$ be the $n-1$ agents. Each agent r_i is assigned a location i; its task is to verify whether the black hole is there. It does so in two steps, executed independently of the other agents.

1. r_i first goes to node $i-1$ in clockwise direction and, if successful, returns to the *home base*.
2. It then goes, in counter clockwise direction, to node $-(n-i-1)$ and, if successful, returns to the *home base*: the black hole resides in the assigned location i.

We now show that the lower bound can be achieved employing $n-1$ agents.

Clearly, in Algorithm OPTTIME, only one agent will be able to complete both its steps without being trapped in the black hole, while the other $n-2$ agents will be destroyed there.

Theorem 5 *Algorithm* OPTTIME *lets $n-1$ co-located agents find the black hole in $2n-4$ time.*

Proof. Let the black hole be located at node b ($-(n-b)$ from the left). The agent reports the information about node $b-1$ at time $2(b-1)$, and the information about node $-(n-b)+1$ at time $2(n-b-1)$. Thus, the total time complexity is $2(b-1)+2(n-b-1) = 2n-4$ and is independent of the location of the black hole. □

Thus, by Theorems 4 and 5, it follows that

Theorem 6 *Algorithm* OPTTIME *is time-optimal.*

3.3 Trading Agents for Time

We now show how to obtain a trade-off between the number of agents employed and the time needed to find the black hole. This is achieved by exploiting two key ideas.

The first key idea is to employ the time-optimal strategy over a number of stages, reusing in each stage the agents. The overall algorithm proceeds in q rounds. In each round $n^{1/q}-1$ agents ($r_1 \ldots r_{n^{1/q}-1}$) follow an algorithm similar to Algorithm 2 to reduce the size of the

unexplored area by a factor of $n^{1/q}$. The unexplored area is divided into $n^{1/q}$ segments $S_1, S_2, \ldots, S_{n^{1/q}}$ of almost equal size (e.g., in the first phase the segment S_i is $(i-1)n^{(q-1)/q} + 1, \ldots, in^{(q-1)/q}$). Agent r_j verifies the guess that the black hole belongs to segment S_j by checking the nodes around S_j (first the right one, then the left one). Clearly only one agent, say r_i, will be able to locate the segment containing the black hole. At this point, we would like to start the new phase applying Algorithm 2 only to S_i. To do so, we need *all* the agents to survive the previous stage.

Unfortunately, in Algorithm OptTime optimality in time is achieved at the expenses of a high level of "casualties". In fact, only one agent will be able to complete both steps of the algorithm and survive, while the other $n-2$ will enter the black hole. This fact makes it impossible to reuse the agents.

The second key idea is to reduce the number of agents that enter the black hole, at the expense of a small increase in time and number of moves. This is achieved by technique we call *Exploring Agents Technique*:

> Two agents e_1 and e_2 (the *exploring agents*), starting from h, move in opposite directions with the only purpose of marking the safe links before disappearing in the black hole. They both move using cautious walk, thus performing $3n$ moves; they complete their exploration in time at most $3n$.

We will employ such an exploration as a (pipelined) pre-processing phase when starting the execution of the main algorithm. The use of this technique frees the other agents from using cautious walk: the other agents follow the main algorithm but they move only if the link they have to traverse is already marked as safe. If such a link is not safe, they simply wait.

We now describe how to employ the ideas used for the Algorithm OptTime in conjunction with the exploring agents technique to obtain a trade-off between the number of agents employed and the time needed to find the black hole. The new algorithm, called Algorithm TradeOff, is reported in Algorithm 3.

As mentioned above, we will divide the unexplored area into disjoint segments, assign an agent to verify whether that segment contains the black hole (using the approach of Algorithm OptTime), and employ the two explorers to save the other agents (see Figure 1).

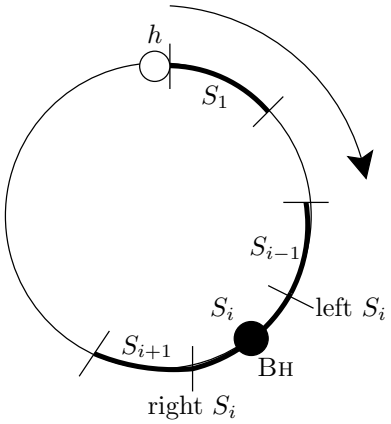

Figure 1: AlgorithmTRADEOFF. At each round, agent r_j verifies the guess that the black hole belongs to segment S_j by checking the nodes immediately outside S_j (first the left one, then the right one.

Clearly only one agent r_i will verify its guess about the assigned region S_i. Observe that, when this happens, all other agents are blocked (possibly inside S_i): the agents r_j with $j < i$ are blocked to the right of the BH, while the agents r_j with $j > i$ are blocked to the left of the BH. To use these agents in the next round, r_i has to notify them.

Notice that, except for the two exploring agents, all agents survive.

Theorem 7 *Let $1 \leq q \leq \log n$. Algorithm* TRADEOFF *let $n^{1/q} + 1$ co-located agents locate the black hole in $2(q+1)n - q - o(n)$ time.*

Proof. The size of the unexplored segment decreases by a factor of $n^{1/q}$ at each round and clearly it becomes equal to $n^{1/q}$ after q rounds.

To simplify the analysis, we will assume that, upon notification, an agent goes to h before starting the next round; notice that this only increases the time and cost, thus the upper bound we derive is a valid one.

Let r^t be the only successful agent in stage t. In round t, each agent $r_j \neq r^t$ moves at most from h to the left border of S_j, then to the right border of S_j, and then back to h. Agent r^t will in addition move also possibly inside its own region to notify the other agents.

Algorithm 3 TradeOff

Let $k = n^{1/q} + 2$ be the number of agents, where q ($1 \leq q \leq \log n$) is the trade-off parameter.

1. Two agents e_1 and e_2 perform the Exploring Agents Technique.

2. The other agents ($r_1 \dots r_{n^{1/q}}$) start their algorithm in pipeline with the two explorers, always leaving from safe ports. The algorithm proceeds in q rounds.

   In each round the unexplored area is divided into $n^{1/q}$ segments $S_1, S_2, \dots, S_{n^{1/q}}$ of almost equal size (e.g., at the first phase the segment S_i is $(i-1)n^{(q-1)/q}+1, \dots, in^{(q-1)/q}$).

   (a) Agent r_j verifies the guess that the black hole belongs to segment S_j by checking the nodes immediately outside S_j (first the left one, then the right one).

   (b) Only one agent, say r_i, will be able to locate the segment containing the black hole. To complete this round, r_i moves left (possibly entering S_i) up to the last safe port.

   (c) At this point, r_i moves right up to the last safe port, passing through h, notifying all other agents to start the computation of the next round; once this is done, it starts the computation of the next round.

   (d) When notified, an agent stops the computation of the current round, and starts the new round.

Hence, the amount of time is at most

$$2\sum_{t=1}^{q}(n - n^{(q-t)/q}) + \sum_{t=1}^{q} N(t),$$

where $N(t)$ is the total amount of time required by the additional moves due to notification in round t.

$N(t)$ is exactly the size of the part of the region assigned to r^t which had been safely explored by the two explorers by the time r^t arrived there. In other words, $N(t) \leq n^{(q-t)/q} - 1$. Hence, the total amount of time is at most

$$2\sum_{t=1}^{q}(n - n^{(q-t)/q}) + \sum_{t=1}^{q}((n^{(q-t)/q}) - 1) = (2n-1)q - o(n).$$

Let us now consider the cost due to the two explorers. The agents are sent in pipeline immediately after the two explorers start to mark the safe ports of the ring. The explorers will then delay the discovery of the black hole of 2 units of time for each unexplored port for a total of additional $2n$ units. Thus, the total time complexity is: $2(q+1)n - q - o(n)$. □

4 Dispersed Agents

In this section we examine the case of dispersed anonymous agents (i.e., initially there is at most one agent at any given location). The number k of agents is not known a priori.

4.1 Lower Bounds and Properties

We now establish a lower bound (that we will prove to be tight in the next section) on the cost for locating the black hole; the lower bound is established for oriented rings and, thus, applies also to the unoriented case.

Theorem 8 *The cost of locating the black hole in oriented rings with dispersed agents is at least* $\Omega(n \log n)$.

Proof. We show that there is an adversary *Adv* such that, for every initial dispersion of k agents and for every black hole solution protocol $\mathcal{P}$, *Adv* can place the black hole and set

link delays in such a way to force $\mathcal{P}$ to make $\Omega(n \log n)$ moves. Let a and b be the agents that start closest to each other. Clearly, the distance between their *home base*s h_a and h_b is at most n/k. Consider an adversary that lets a and b move, but blocks (makes very slow) all the other agents, until they come in contact with a, b or with an already unblocked agent. Since a and b do not know the number of agents, they must behave correctly also in the case there are no other agents. In particular, let optimistically set the initial explored area to be the (shorter) segment between h_a and h_b; note that its size is at most $\lfloor n/k \rfloor + 1$, while the initial unexplored area has size at most $\lfloor n(k-1)/k \rfloor - 1$. In this scenario, Lemma 3 as well the proof of Theorem 1 applies, as if a and b started at the same place and reached their current position in the course of the algorithm. In particular, a blocked agent will become unblocked by *Adv* only when the area explored by a and b is expanded to include its *home base*. Since the proof of Theorem 1 makes no assumptions on the number of agents moving inside the explored area, the lower bound of $\Omega(n \log n)$ follows. □

The proof of the above lemma relies on the fact that the agents do not know their number k. We now show that, even if every agent is endowed with a priori knowledge of k, the cost remains high.

Theorem 9 *If k is known a priori to the agents, the cost of locating the black hole in oriented ring is $\Omega(n \log(n-k))$.*

Proof. The adversary places all agents in a block of k contiguous nodes. Clearly, the technique from Theorem 1 can be applied with the initial explored region set to be the block containing all the agents. □

The proof of Theorem 9 considers a worst-case scenario: an adversarial placement of both the black hole and the agents in the ring. So, one last question is whether, knowing k we could obtain a substantially better performance under a (blind but) favorable placement of the agents in the ring; i.e., assuming that k is known a priori and that *we* can place the agents, leaving to the adversary only the placement of the black hole. Also in this case, the answer is substantially negative. In fact, the application of the proof technique of Theorem 1 (with the initial explored region set to be the smallest connected region containing all agents, which is clearly of size at most $n - n/k$) yields a lower bound of $\Omega(n \log(n/k)) = \Omega(n(\log n - \log k)$, which, for reasonably small k, is still $\Omega(n \log n)$.

A simple but important property is that, although anonymous, the dispersed agents can uniquely identify each other by means of purely local names. This is easily achieved as follows. Each agent a will think of the nodes as numbered with consecutive integers in the clockwise direction, with its starting node (its *home base*) as node 0. Then, when moving, agent a will keep track of the relative distance d_a from the *home base*: adding $+1$ when moving clockwise, and -1 otherwise. Thus, when a finds at the node (at distance) $d_a = -3$ a message written by an agent b that is at distance $d_b = +2$ from its own *home base*, a is able to unambiguously determine the *home base* of b (in a's view of the ring) as follows. If the ring is oriented, b's *home base* is node $d_a - d_b = -5$. If the ring is not oriented, that message must also indicate which of the two ports leads to the "right" in b's view of the ring; if a and b share the same view, then a computes as before; otherwise b's *home base* in a's view is node $d_a + d_b = -1$. In other words,

Lemma 4 *Each agent can distinguish and recognize all the other agents in an oriented ring.*

With dispersed agents, there is a major difference between oriented and unoriented rings. In fact, if the ring is unoriented, two agents no longer suffice to solve the problem: they could be located in the nodes next to the black hole, and both made their first move towards it. In other words,

Lemma 5 *At least three dispersed agents are needed to locate the black hole in an unoriented ring.*

Thus, when dealing with the unoriented ring, we will assume that there are at least three dispersed agents.

4.2 Oriented Rings: Cost-Optimal Solution

In this section we describe a cost-optimal algorithm for the oriented ring where $k \geq 2$ anonymous agents are dispersed. The algorithm is composed of three distinct *phases*: *pairing*, *elimination*, and *resolution*.

In the first phase, agents form pairs, according to the protocol described in AlgorithmPairing.

Algorithm 4 PAIRING

Initially, all agents are not paired. Each agent moves along the ring clockwise, using cautious walk, marking (direction and distance to its own starting node) the visited nodes.

1. If an agent reaches a node visited by another agent b, it becomes *chasing*, and follows b's trace.
2. If an agent arrives at a node visited by two agents, it terminates with status *alone*.
3. If a *chasing* agent reaches the last safe node visited by the chased agent, it leaves a mark *Join me* and terminates with status *paired-left*.
4. If a non *chasing* agent, during its cautious walk, encounters the mark *Join me*, it clears this mark and terminates with status *paired-right*.

The agents with status *paired-* will then execute the elimination phase to locate the black hole. The agents terminating with status *alone* will be passive in the remainder of the computation.

Lemma 6 *At least one pair is formed during the pairing phase. The pairing phase lasts at most* $3n - 6$ *time units, and its cost is at most* $4n - 7$.

Proof. First we prove correctness, i.e. that at least one pair is formed. Let r_1, r_2 and r_3 be the first, second and third (if any) agent to the left of the black hole (refer to Figure 2). Either will r_2 arrive to the *home base* of r_1 or it will be reached by r_3. In the latter case, r_2 and r_3 form a pair. If, instead, r_2 arrives to the *home base* of r_1, it will chase r_1 and a pair will be formed at the last node r_1 was able to explore before either being reached by r_2 or entering the black hole.

Consider now the cost of the pairing phase. Due to rules 2. and 4., no link is traversed by more then two agents. The link to the right of the black hole is not traversed at all, while the link to the left of the black hole is traversed by at most one agent. There are at most 3 moves on all other links due to cautious walk and additional move due to chasing. Summing all these terms yields a bound of at most $4n - 7$ moves in total.

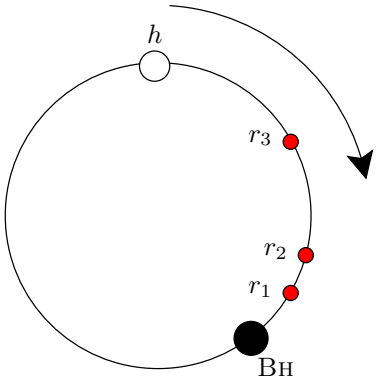

Figure 2: Proof of Lemma 6.

Consider the time complexity. According to the algorithm, at any time, within 3 time steps, each non-terminated agent either terminates or moves at least one node to the right. There are at most $n-2$ such possible moves, resulting in the $3n-6$ bound on time. □

Note that, if the pairing algorithm starts with k agents, any number of pairs between 1 and $\lfloor k/2 \rfloor$ can be formed, depending on the timing. For example $\lfloor k/2 \rfloor$ pairs are formed when the "even" (as counting to the left from the black hole) agents are very slow, and the "odd" agents are fast and reach their right neighbors.

Since agents can distinguish themselves using local names based on their starting nodes (Lemma 4), also the pairs can be given local names, based on the node where the pair was formed (the *home base*). This allows a pair of agents to ignore all other agents. Using this fact, a straightforward solution to the Black Hole Search problem consists of having now each pair independently execute the location algorithm for two agents (AlgorithmDivide). This however will yield an overall $O(n^2 \log n)$ worst-case cost.

To reduce the cost, the number of active pairs must be effectively reduced. The reduction is done in the next phase, called *elimination*, by executing Algorithm Elimination. In this phase, the number of "active" pairs is reduced to at most two. The two agents in the pair formed at node v will be denoted by r_v and l_v, and referred to as the right and the left agent, respectively; v will be their *home base*.

In Algorithm Elimination, the rule of Case (b) renders stronger a *home base* (and, thus, a pair) in a higher logical round; ties are resolved giving priority to the right node (by Case

Algorithm 5 ELIMINATION

The computation proceeds in logical rounds. In each round,

1. the left agent l_v cautiously moves to the left until

   (a) it is destroyed by the black hole, or

   (b) it reaches a *home base* u with higher or

   (c) equal round number.

   In Case (b), l_v returns to v, marks it *Eliminated*, and stops any further execution. In Case (c), l_v marks u as *Eliminated* and returns to v; if v is not marked *Eliminated*, it is promoted to the next round.

2. Similarly, agent r_v cautiously moves to the right until it finds (if it is not destroyed by the black hole) the first *home base* u in equal or higher round; it then returns back to v. If the current level of v (its level could have risen during the travel of r_v) is not higher then the level of u, v is marked *Eliminated* and r_v stops any further execution; otherwise (v is not marked *Eliminated*) r_v travels again to the right (it is now in a higher round).

To prevent both agents of a pair entering the black hole, both l_v and r_v maintain a counter and travel to distance at most $\lfloor (n-1)/2 \rfloor$. If one of them has traveled such a distance without finding another *home base* with the same or higher round, it returns back to v, and v is marked *Selected*.

(c) and by the way the right agent behaves). In the following, we will call *selected* the pair whose *home base* has been marked *Selected* during the elimination phase, and *Eliminated* the pair whose *home base* has been marked *Eliminated*. The following lemma shows that the elimination phase will eventually select either one or two pairs.

Lemma 7 *The elimination phase selects at least one and at most two pairs in* $O(n \log n)$ *moves.*

Proof. First note that, since the distance the agents travel is limited by $\lfloor (n-1)/2 \rfloor$, it cannot happen that both agents of a pair fall into the black hole. Moreover, according to Algorithm ELIMINATION, if a pair is promoted to a higher round, the pair to its left has been eliminated. This means that the number of agents still not eliminated is at least halved in each round; thus there are at most

$$\log(k/2) < \log n \tag{6}$$

rounds.

Consider the rightmost (with respect to the black hole) *home base* v of a pair that reached round i. We show that either there is a pair that reached round $i+1$ or v becomes *Selected*. There are four possible outcomes of the travel done by agent l_v in round i:

1. l_v found a node in a higher round.
2. l_v found a node in the same round. In this case v is promoted to the next round.
3. l_v traveled distance $\lfloor (n-1)/2 \rfloor$ without entering the black hole and without finding a node in the same or higher round. In this case, v is declared *Selected*.
4. l_v entered the black hole. Since v is the rightmost (with respect to the black hole) *home base* of a pair in round i, r_v will not find any node in the same or higher round. Moreover, since the black hole is at the distance at most $\lfloor (n-1)/2 \rfloor$ to the left from v, r_v will not enter the black hole. In other words, r_v will return to v after its counter expired and v will be declared *Selected*.

In the first two cases there is a node in round $i+1$, in the remaining cases v is declared *Selected*. Since the number of rounds is bounded, eventually some node will be *Selected*.

Because of the limited traversal, there could be several *Selected* nodes. However, each of the selected pairs has explored at least $\lfloor (n+1)/2 \rfloor$ nodes (including the *home base*) without encountering a black hole or another *Selected* node; hence there are at most 2 *Selected* nodes.

Since there are at most four moves on each link by agents of a given round (each agent travels only to the closest neighbour of the same round and back home), the cost of each round is $O(n)$. By Equation (6), the lemma follows. □

When a *home base* v has been marked *Selected*, the corresponding agents l_v and r_v will then start the resolution phase of the algorithm, consisting in the executing of Algorithm DIVIDE. Note that, for each of the selected pairs, the execution of Algorithm DIVIDE is started by a single agent; the other agent either has been destroyed by the black hole or will join in the execution upon its return to the *home base*.

The overall algorithm (consisting of the pairing, elimination and resolution phases) that allows a number of Dispersed agents to locate a black hole in an Oriented Ring will be called DOR. By Lemmas 6–7, and from the correctness of Algorithm DIVIDE shown in Theorem 2, we have

Theorem 10 *Algorithm* DOR *lets $k \geq 2$ dispersed anonymous agents locate the black hole in oriented rings in $O(n \log n)$ time and cost.*

Thus, by Theorems 8 and 10, it follows that

Corollary 1 *Algorithm* DOR *is cost-optimal.*

4.3 Oriented Rings: Considerations on Time

In the previous section, we have shown that the lower bound on cost is tight, and can be achieved by two agents. This implies that the presence of multiple agents can not reduce the cost of locating the black hole. The natural question is whether the presence of more agents can be successfully exploited to reduce the time complexity.

In the case of co-located agents, we were able to distribute the workload among them so to reduce time. In the case of dispersed agents, to achieve the same goal, they must first

be able to find each other. Moreover, note that, if the agents are able to quickly *gather* at a node, then they can execute Algorithm TradeOff. This is the approach we are going to take. In the remainder of this section, we focus on the problem of quickly gathering the agents.

If the number k of agents is known, the gathering problem is easily solved by Algorithm Gathering. Eventually, since all agents travel to the right, all but one agent (which

Algorithm 6 Gathering

The number k of agents is known.

1. Each agent moves along the ring clockwise using cautious walk.
2. When arriving at a node already visited by another agent, it proceeds to the right via the safe port. If there is no safe port, it tests how many agents are at this node; if the number of agents at the node is $k-1$, the algorithm terminates.

will reach the black hole) will be at the same node; in the worst case, this node will be the left neighbor of the black hole. Since, using cautious walk, it takes at most three time units to safely move to the right, and since there are at most $n-2$ such possible moves, we have:

Lemma 8 *Let k be the total number of dispersed agents in the ring, and let k be known. Algorithm* Gathering *lets $k-1$ agents gather in an oriented ring with a black hole in time $3n-6$.*

Therefore, we have the following

Theorem 11 *In oriented rings, k agents can locate the* Bh *in $O\left(\frac{n/logn}{\log(k-2)}\right)$ time, with k known.*

Proof. Use Algorithm Gathering to gather $k-1$ agents; the other one acts as one of the two explorers required by Algorithm TradeOff. By Theorem 7 and Lemma 8, with $k-2=n^{\frac{1}{q}}$, the theorem follows. □

Clearly, Algorithm Gathering can not be applied when k is unknown, as the agents have no means to know when to terminate (and, thus, to switch to Algorithm TradeOff).

We do not know whether an improvement in time complexity can be obtained if $k > 2$ is not known.

Unlike cost, there are several measures of time complexity; in addition to the main one, *ideal* time considered so far, there are also *bounded delay* time and *causal* time.

If the *bounded delay* time complexity is considered (i.e., assuming a global clock and that each move takes at most one time unit), the additional agents can indeed help, even if k is not known, as shown in Algorithm BOD.

Algorithm 7 BOD

---

k is unknown, and initially all agents are in state *alone*.

Rules for agent r in state *alone*.

1. Cautiously walk to the right until you meet another agent r'.
2. If r' is in state *alone*, form a group $\mathcal{G}$ (r and r' change state to *grouped*) and start executing the group algorithm.
3. Otherwise (r' is in state *grouped*, belonging to the group $\mathcal{G}'$, formed at the node g') join the group $\mathcal{G}'$: Go to g' and set your state to $Join[g']$.

Rules for group $\mathcal{G}$ formed at node g, consisting of $|\mathcal{G}|$ agents.

Execute Algorithm TRADEOFF using $|\mathcal{G}|$ agents, with the following actions taken after finishing each phase and before starting the next one:

1. If any of your agents have seen agents of another group $\mathcal{G}'$ whose starting node g' is to the right of g, join group $\mathcal{G}'$ by sending all your agents to g', with state $Join[g']$.
2. Otherwise add all the agents waiting at g with state $Join[g]$ to $\mathcal{G}$ and execute the next phase of Algorithm 3 using the updated number of agents.

Theorem 12 *Algorithm* BOD *lets $k = n^{1/q}$ dispersed agents locate the black hole in oriented rings in $O(qn^{1/q})$* bounded delay *time complexity.*

Proof. Observe (similarly to Lemma 6) that, by time $3n-6$, all agents will belong to some group. Note that, by executing the first phase of Algorithm TradeOff, each group explores a segment of size at least $(n+1)/2$ (including its starting node). However, this means that by time $5n-10$ every two groups will come in contact, collapsing into one (the rightmost) group $\mathcal{G}$. Since no group except $\mathcal{G}$ starts the second phase, and since $n-2$ time units are enough to make all the agents of other groups converge to the starting node of $\mathcal{G}$, by time $6n-12$, group $\mathcal{G}$ contains all the agents. This means that the time complexity of Algorithm Bod is at most as the time complexity of the Algorithm TradeOff plus an additional $O(n)$. □

If *causal* time complexity is considered (i.e., the length of the longest chain of causally related moves, over all possible executions of the algorithm), the availability of additional agents can be of little help in the worst case.

Lemma 9 *The* causal time *complexity of locating the black hole in an oriented ring, using k agents is at least $n(\log n - \log k) - O(n)$.*

Proof. Place all but two agents in a block next to the black hole. Consider an adversary that makes all the links leading to nodes within this block very slow. The two remaining agents are placed elsewhere and are free to move. According to the technique used in the proof of Theorem 1 we have that, using 2 agents, the time for locating the region of size k containing the black hole is $n(\log n - \log k)$. Hence, the two free agents will have to do all this work before they can get help from the remaining agents. □

4.4 Unoriented Ring

If the ring is unoriented, at least three dispersed agents are needed to locate the black hole (Lemma 5). Thus, we assume that there are at least three dispersed agents.

It is easy to convert a solution for oriented rings into one for the unoriented ones, at the cost of twice the number of moves and of agents.

Lemma 10 *Let $\mathcal{P}$ be a protocol for oriented ring which, using p agents solves a problem $\mathcal{S}$ in time T and cost C. Then there is a protocol $\mathcal{P}'$ for unoriented ring which, using $2p-1$ agents, solves $\mathcal{S}$ in time T and complexity at most $2C$.*

Proof. The agents in the unoriented ring can be classified into two groups, according to what they assume clockwise to be. The problem is solved by the agents of each groups independently. Each agent remembers the direction it considers right and executes $\mathcal{P}$, ignoring the agents with different opinion. Since there are $2p-1$ agents, one group will have at least p agents. That group will successfully execute $\mathcal{P}$ and solve $\mathcal{S}$. The running time is not influenced; the complexity is at most doubled. □

Lemma 10 can be applied to all previous algorithms presented for scattered agents. Note that, for Algorithm GATHERING, it is sufficient to set $k = p$.

From Theorem 10, Lemma 10, and Lemma 5, it follows that:

Theorem 13 *Three dispersed agents are necessary and sufficient to locate the black hole in an unoriented ring. They can do so with optimal* $\Theta(n \log n)$ *cost.*

5 Extensions

We have provided solutions to the black hole search problem in anonymous rings for two settings: when the anonymous agents are co-located and when they are dispersed.

The protocol for the dispersed agents setting, Algorithm DOR, can be actually employed to solve the black hole search problem in the more general setting, when the anonymous agents are in more than one location in the ring, but some locations contain more than one agent; that is, the agents are neither all co-located nor fully dispersed.

In this general setting, Algorithm DOR can be used as follows.

1. If an agent is the sole occupant of its *home base*, it will execute the algorithm without modifications.

2. In the case of several agents sharing the same *home base*, Algorithm PAIRING (the first step of Algorithm DOR) becomes simpler:

   (a) only two of the colocated agents will be selected (e.g., using the whiteboard to break ties);

   (b) the two selected agents will form a pair and execute the rest of the algorithm, while the others will become *alone*.

It is not difficult to verify that the complexity is not increased.

Acknowledgments This work has been partially supported by NSERC and by VEGA (Slovak Grant Agency).

References

[1] S. Alpern, V. Baston, and S. Essegaier. Rendezvous search on a graph. *Journal of Applied Probability*, 36(1):223–231, 1999.

[2] E. Arkin, M. Bender, S. Fekete, and J. Mitchell. The freeze-tag problem: how to wake up a swarm of robots. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 568–577, 2002.

[3] B. Awerbuch, M. Betke, and M. Singh. Piecemeal graph learning by a mobile robot. *Information and Computation*, 152:155–172, 1999.

[4] M. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proc. of 30th ACM Symp. on Theory of Computing (STOC '98)*, pages 269–278, 1998.

[5] David M. Chess. Security issues in mobile code systems. In *Proc. Conf. on Mobile Agent Security*, LNCS 1419, pages 1–14, 1998.

[6] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. In *Proc. of 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, 2002.

[7] K. Diks, A. Malinowski, and A. Pelc. Reliable token dispersal with random faults. *Parallel Processing Letters*, 4:417–427, 1994.

[8] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile agents searching for a black hole in an anonymous ring. In *Proc. of 15th International Symposium on Distributed Computing (DISC 01)*, pages 166–179, 2001.

[9] P. Fraigniaud, A. Pelc, D. Peleg, and S. Pérennes. Assigning labels in unknown anonymous networks. In *Proc. of 19th ACM Symposium on Principles of Distributed Computing (PODC 2000)*, pages 101–112, 2000.

[10] O. Goldreich and L. Shrira. On the complexity of computation in the presence of link failures: The case of a ring. *Distr. Computing*, 5:121–131, 1991.

[11] M.S. Greenberg, J.C. Byington, and D. G. Harper. Mobile agents and security. *IEEE Commun. Mag.*, 36(7):76 – 85, 1998.

[12] N. Hanusse, D. Kavvadias, E. Kranakis, and D. Krizanc. Memoryless search algorithms in a network with faulty advice. In *Proc. of 2nd IFIP International Conference on Theoretical Computer Science, (TCS '02)*, pages 206–216, 2002.

[13] F. Hohl. A model of attacks of malicious hosts against mobile agents. In *Proc. of the ECOOP Workshop on Distributed Object Security and 4th Workshop on Mobile Object Systems (LNCS 1603)*, pages 105 – 120, 1998.

[14] L.M. Kirousis, E. Kranakis, D. Krizanc, and Y.C. Stamatiou. Locating information with uncertainty in fully interconnected networks. In *Proc. of 14th International Symposium on Distributed Computing, (DISC '00)*, pages 283–296, 2000.

[15] R. Oppliger. Security issues related to mobile code and agent-based systems. *Computer Communications*, 22(12):1165 – 1170, 1999.

[16] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33:281–295, 1999.

[17] T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. In *Mobile Agents and Security*, LNCS 1419, pages 44–60, 1998.

[18] K. Schelderup and J. Ines. Mobile agent security - issues and directions. In *6th Int. Conf. on Intell. and Services in Networks*, LNCS 1597, pages 155–167, 1999.

[19] L.C. Thomas and P.B. Hulme. Searching for targets who want to be found. *Journal of the Operations Research Society*, 48(1):44–50, 1997.

[20] Jan Vitek and Giuseppe Castagna. Mobile computations and hostile hosts. In D. Tsichritzis, editor, *Mobile Objects*, pages 241–261. University of Geneva, 1999.