Searching for a Black Hole in Arbitrary Networks: Optimal Mobile Agent Protocols

Stefan Dobrev
Paola Flocchini
University of Ottawa
sdobrev@site.uottawa.ca
flocchin@site.uottawa.ca

Giuseppe Prencipe
University of Pisa
prencipe@di.unipi.it.

Nicola Santoro
Carleton University
santoro@scs.carleton.ca.

ABSTRACT

Protecting agents from host attacks is a pressing security concern in networked environments supporting mobile agents. In this paper, we consider a *black hole*: a highly harmful host that disposes of visiting agents upon their arrival, leaving no observable trace of such a destruction. The task to identify the location of the harmful host is clearly dangerous for the searching agents. We study under what conditions and at what cost a team of autonomous asynchronous mobile agents can successfully accomplish this task; we are concerned with solutions that are *generic* (i.e., topology-independent). We study the *size* of the optimal solution (i.e., the minimum number of agents needed to locate the black hole), and the *cost* of the minimal solution (i.e., the number of moves performed by the agents executing a size-optimal solution protocol). We establish tight bounds on size and cost depending on the a priori knowledge the agents have about the network, and on the consistency of the local labellings. In particular, we prove that: with *topological ignorance* $\Delta + 1$ agents are needed and suffice, and the cost is $\Theta(n^2)$, where Δ is the maximal degree of a node and n is the number of the nodes in the network; with topological ignorance but in presence of *sense of direction* only *two* agents suffice and the cost is $\Theta(n^2)$; and with *complete topological knowledge* only *two* agents suffice and the cost is $\Theta(n \log n)$. All the upper-bound proofs are constructive.

Keywords

Mobile Agents, Black Holes, Distributed Computing

1. INTRODUCTION

In a networked environment that makes use of mobile agents, *security* is a pressing concern, and possibly the most difficult one to address. Actually, even the most basic security issues, in spite of their practical urgency and of the amount of effort, must still be effectively addressed. The causes of this situation are not lack of interest and effort, but rather the (unexpected) difficulties found when developing solutions; the nature of these obstacles is varied, most are technological, some computational. Witness, for example, the large effort to determine how to protect a network site (a *host*) from malicious agents, as well as to protect agents from *host attacks* (e.g., see [1, 5, 8, 10, 12]).

In this paper we consider the issue of host attacks; that is, the presence in a site of processes that harms incoming agents (e.g., see [6, 7, 9, 11, 13]). Obviously, the first step in any solution to such a problem must be to *identify*, if possible, the harmful host; i.e., to determine and report its location; following this phase, a "rescue" activity would conceivably be initiated to deal with the destructive process resident there. The task to identify the harmful host is clearly dangerous for the searching agents and, depending on the nature of the harm, might be impossible to perform.

In this paper, we consider a highly harmful process: a stationary process that *disposes* of visiting agents upon their arrival, leaving *no observable trace* of such a destruction. Due its nature, the site where such an item is located is called a *black hole* [2]. The task is to unambiguously determine and report the location of the black hole, and will be called *black hole search*. The research concern is to determine under what conditions and at what cost mobile agents can successfully accomplish this task. The searching agents start from the same safe site and follow the same set of rules; the task is successfully completed if, within finite time, at least one agent survives and knows the location of the black hole.

Black hole search is a non trivial problem, its difficulty is aggravated by the simultaneous presence of asynchrony of the agents and absence of any trace of destruction (outside the black hole). The problem has been investigated when the network is an anonymous *ring* (i.e., a loop network of identical nodes sites) [2]. For such networks, an in-depth characterization of the problem was given. The results established there are however topology-dependent; in fact, in their solution protocols, the agents exploited the properties of the ring to derive the location of the black hole.

In this paper we are interested in the black hole search problem in its more general setting, and in particular we are concerned with *generic* solutions, i.e. solutions that are topology-independent.

We ask computational questions regarding the *size* of the optimal solution (i.e., the minimum number of agents needed to locate the black hole), and the conditions for their ex-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PODC '02 Monterrey, California, USA
Copyright 2002 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

istence. Some answers are immediate. For example, the problem is clearly unsolvable if the graph G representing the network topology is not 2-connected, so we will only consider 2-connected graphs. Similarly, a single agent is incapable of performing the task, so the size is at least two; how realistic is this bound? how many agents suffice? We are also interested in the *cost* of the minimal solution (i.e., the number of moves performed by the agents executing a size-optimal solution protocol).

In this paper we provide specific answers to these questions. We show that the answers vary depending on the a priori knowledge the agents have about the network, and on the consistency of the local labellings.

We consider first the situation of *topological ignorance*; that is when the agents have no a priori knowledge of the topological structure of G. We show that any generic solution needs at least $\Delta + 1$ agents, where Δ is the maximal degree of G, even if the agents know Δ and the size n of G. We further prove that, in any *minimal* generic solution, the agents will perform $\Omega(n^2)$ moves in the worst case. Both these bounds are *tight*. In fact we construct a protocol that correctly locates the black hole in $O(n^2)$ moves using $\Delta+1$ agents that know Δ and n.

We then consider the case of topological ignorance in systems where there is *sense of direction*[3]; informally, sense of direction is a labelling of the ports that allows the nodes to determine whether two paths starting from a node lead to the same node, using only the labels of the ports along these paths. We show the surprising result that, if this is the case, *two agents* suffice to locate the black hole, regardless of the (unknown) topological structure of G. The proof is constructive, and the proposed algorithm has a $O(n^2)$ cost. We further show that this cost is optimal; in fact we prove that any two agents algorithm for locating a black hole in arbitrary networks with neighbourhood SD (a particular type of sense of direction) has a $\Omega(n^2)$ cost in the worst-case.

Finally, we consider the case of *complete topological knowledge* of the network; that is, the agents have a complete knowledge of the edge-labelled graph G, the correspondence between port labels and the link labels of G, *and* the location of the source node (from where the agents start the search). In this case, not surprisingly, two agents suffice. We constructively prove that, in this case, the cost of a minimal protocol can be reduced to $O(n \log n)$, and furthermore this is optimal.

2. DEFINITIONS AND BASIC PROPERTIES

Framework

Let $G = (V, E)$ be a simple 2-connected graph; let $n = |V|$ be the size of G, $E(x)$ be the links incident on $x \in V$, $d(x) = |E(x)|$ denote the degree of x, and Δ denote the maximum degree in G. If $(x, y) \in E$ then x and y are said to be neighbours. The nodes of G can be *anonymous* (i.e., without unique names). At each node x there is a distinct label (called port number) associated to each of its incident links (or ports); let $\lambda_x(x, z)$ denote the label associated at x to the link $(x, z) \in E(x)$, and λ_x denote the overall injective mapping at x. The set $\lambda = \{\lambda_x | x \in V\}$ of those mappings is called a *labelling* and we shall denote by (G, λ) the resulting edge-labelled graph.

Let $P[x]$ denote the set of all paths with x as a starting point, and let $P[x, y]$ denote the set of paths starting from node x and ending in node y. Let Λ be the extension of the labelling function λ from edges to paths. A *coding function* $\mathbf{c}$ of a system (G, λ) is a function such that: $\forall x, y, z \in V$, $\forall \pi_1 \in P[x, y]$, $\pi_2 \in P[x, z]$ $\mathbf{c}(\Lambda_x(\pi_1)) = \mathbf{c}(\Lambda_x(\pi_2))$ iff $y = z$. Thus, for any two paths π_1 and π_2 from x to y, $\mathbf{c}(\Lambda_x(\pi_1)) = \mathbf{c}(\Lambda_x(\pi_2))$; we shall denote this value $\beta_x(y)$ and call it the *local name* of y at x. Given a coding function $\mathbf{c}$, a *decoding function* $\mathbf{d}$ for $\mathbf{c}$ is such that $\forall x, y, z \in V$, such that $(x, y) \in E(x)$ and $\pi \in P[y, z]$, $\mathbf{d}(\lambda_x(x, y), \mathbf{c}(\Lambda_y(\pi))) = \mathbf{c}(\lambda_x(x, y) \circ \Lambda_y(\pi))$, where $\circ$ is the concatenation operator. In other words, by definition of decoding function we have that $\mathbf{d}(\lambda(x, y), \beta_y(z)) = \beta_x(z)$. Given a coding function $\mathbf{c}$ of (G, λ) and a decoding function $\mathbf{d}$ for $\mathbf{c}$, the couple $(\mathbf{c}, \mathbf{d})$ is called a *sense of direction* for (G, λ) [3].

Operating in (G, λ) is a set $\mathcal{A}$ of r distinct autonomous mobile agents. The agents can move from node to neighbouring node in G, have computing capabilities and bounded computational storage ($O(logn)$ bits suffice for all our algorithms), obey the same set of behavioral rules (the "protocol"). The agents are *asynchronous* in the sense that every action they perform (computing, moving, etc) takes a finite but otherwise unpredictable amount of time. Initially, all agents are in the same node h, called *home base*.

Each node has a bounded amount of storage, called *whiteboard*; $O(logn)$ bits suffice for all our algorithms. Agents communicate by reading from and writing on the whiteboards; access to a whiteboard is gained fairly in mutual exclusion.

Black Hole Search

A *black hole* is a node where resides a stationary process that 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* (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.

A solution protocol is *generic* if it solves BHS regardless of G; in this paper we will only consider generic protocols. The main measure of complexity of a solution protocol $\mathcal{P}$ is the *size*, that is the number of agents used by $\mathcal{P}$. We will also consider the total number of moves performed by the agents, and call it the *cost* of $\mathcal{P}$.

We will study generic solutions and their complexity depending on the type of topological information the agents might have a priori available; we always assume they all know n. If no additional topological information is available, the agents operate with *topological ignorance*. If the system (G, λ) has sense of direction $(\mathbf{c}, \mathbf{d})$ that is known to the agents, we say the agents operate with *sense of direction*. Finally, the agents have *complete topological knowledge* of (G, λ) if the following information is available to all agents: (1) Knowledge of the labelled graph (G, λ); (2) Correspondence between port labels and the link labels of (G, λ); (3) Location of the home base in (G, λ).

Basic Limits

Due to the asynchrony of the agents and to the nature of the black hole, the following basic properties simply hold.

LEMMA 2.1. *[2]*

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

2. *It is impossible to determine the location of the black hole if the size of G is not known.*
3. *It is impossible to verify whether or not there is a black hole.*

Thus, we assume that there are at least two agents; furthermore, the existence of the black hole and the size of G is common knowledge to the agents.

Cautious Walk

At any moment of the execution of a protocol, the ports will be classified as *unexplored* – no agent has been sent/received via this port, *explored* – an agent has been received via this port or *active* – an agent has been sent via this port, but no agent has been received via it. Obviously, an explored port does not lead to a black hole; on the other hand, both unexplored and active ports might lead to it. To minimize the number of casualties (i.e., agents entering the black hole), we will not allow any agent to leave through an active port. To prevent the execution from stalling, we will require any active port not leading to the black hole, to be made explored as soon as possible.

This is accomplished as follows: Whenever an agent a leaves a node u through an unexplored port p (transforming it into active), upon its arrival to the node v, and before proceeding somewhere else, a returns to u (transforming that port into explored). We call this technique *Cautious Walk*.

3. COMPUTING WITH TOPOLOGICAL IGNORANCE

3.1 Lower Bounds

The lower bounds we are going to prove are for algorithms that know n and work on all 2-connected graphs of maximal degree at most Δ. (The algorithm knows Δ, but the graph does not necessarily has a vertex of degree Δ.)

We follow an approach common in lower bound proofs by viewing the execution as a game between an algorithm and an adversary. The goal of the algorithm is to locate the black hole and to terminate. The adversary tries to either force the algorithm to behave incorrectly, or make it costly. The adversary has the power (1) to choose the graph (with port labels), (2) to place the black hole and (3) to set (navigational) delays. The first two points are due to the fact that the algorithm does not know the network, neither the location of the black hole. The fact that agents are asynchronous gives the adversary the third power.

Because of the asynchrony, we can limit ourselves to reactive algorithms: If an algorithm uses timeouts, the adversary can make all timeouts expire without allowing any agent to arrive to its destination. This means that an agent at a node must either wait for the arrival of an agent, or to depart via one of the incident ports.

We further simplify matters by turning an active port p into explored at the moment the agent that departed via p arrives to its destination. This essentially gives any algorithm the power of cautious walk, without requiring that an agent must return to mark an active port as explored. Clearly, this only strengthens the algorithm and makes the lower bound result stronger. Since for every algorithm there is an equivalent one using cautious walk, we can limit ourselves to algorithms that avoid sending an agent via an active port.

The above discussion narrows the possible actions the algorithm can specify for available (not in transit and not waiting) agents to (1) departure via an unexplored port, (2) departure via an explored port and (3) waiting until another agent arrives, or a port status changes.

The adversary applies its power to set delays by being able to specify at any moment of the execution which of the agents in transit arrive to their destination, and which of them remain travelling.

We say a configuration is *stable* if there are no agents in transit over explored links. Since an explored link does not lead to a black hole, whenever there are some agents in transit over such links, the adversary must eventually allow all these agents to arrive to their destinations. This suggest the following rule used by our adversary:

- *If there are agents travelling via explored links, block all active and unexplored ports until a stable configuration is reached.*

Applying this rule allows us to focus on the behaviour of the adversary in stable configurations only.

At any moment of the execution the maximal information the algorithm can have about the underlying graph can be described by a graph G_e induced by the explored nodes, together with the information about their degree and the labels of the incident ports in the underlying graph. (We will denote this graph together with the additional information by G_e^*.)

The power of the adversary to choose the graph and the port labelling comes into play in stable configurations, when the adversary allows some agents to arrive to their destination. At that moment, the adversary chooses a *witness graph* $G = (V, E)$ (together with its port labelling) in which these freed agents arrive to their destination. The graph G must be consistent with the previous execution of the algorithm. This consistency means that G_e is an induced subgraph of G and for all nodes in G_e the information (degree and port labels) contained in G_e^* matches these values in G. Note that G must be 2-connected, of order n and should have maximal degree at most Δ. In the rest of this section we restrict ourselves only to such graphs, without explicitly stating these properties. Observe that there are usually many possible witness graphs consistent with the current knowledge G_e^*, and the adversary's choices of the witness graph can vary during the execution.

Since the adversary can permanently block only links leading to the black hole, we immediately get the following observation:

OBSERVATION 3.1. *If, for a stable configuration, there does not exist a graph G and a node u from G such that G is consistent with G_e^* and all active ports lead to u, then the adversary must allow at least one agent to arrive to its destination.*

The following corollary describes in detail some conditions under which such witness graph and node do not exist:

COROLLARY 3.2. *If any of the following conditions hold in a stable configuration, then the adversary must unblock at least one agent.*

a) A node has two incident active ports.

b) There are at least $\Delta + 1$ active ports.

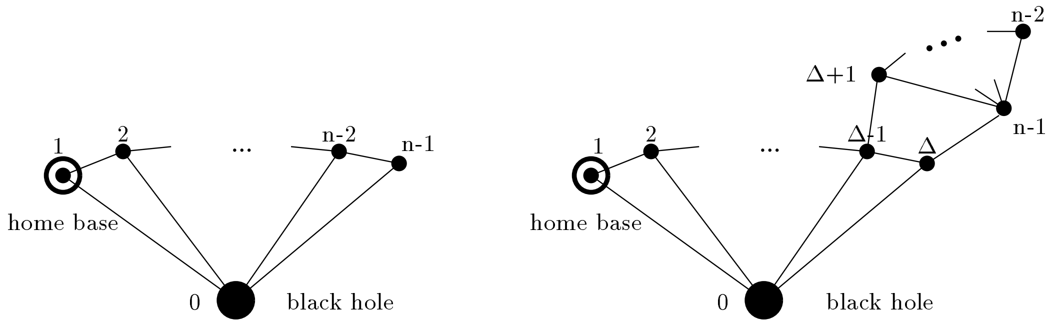

Figure 1: The lower bound graphs for $\Delta \geq n-4$(left) and $\Delta \leq n-4$ (right) .

c) *There are Δ active ports, at most one open node (explored node with an unexplored port) and at most $n-2$ explored nodes.*

d) *There are some active ports, no open node and at most $n-2$ explored nodes.*

PROOF. For each case we prove that there is no witness graph G and a node u from G, such that all active ports lead to u.

a) Since G does not contain multiple edges, there can be at most one link from any node to u.

b) The maximal degree of u is Δ.

c) Suppose all active ports lead to u. Because there are Δ active ports, all Δ links of the node u are used. Hence the unexplored part of G must be connected directly to the explored part (via unexplored ports). Since there is at most one open node where it can be connected, there is no way G to be 2-connected.

d) The same as c), but now the unexplored part can be connected only to the node u, which again contradicts the 2-connectedness of G.

□

Note that although avoiding conditions from Corollary 3.2 is necessary for the adversary to be consistent, it is not sufficient: Consider the following G_e^*: a) G_e contains $n-2$ nodes, b) $\Delta = 3$, c) there are three explored nodes v_1, v_2, v_3, each one having one active and zero unexplored ports d) there is one open node v_4 with two unexplored links. Clearly, it is not possible to have all active ports leading to the same node u, because v_4 also has to be connected to u, violating $\Delta = 3$.

Another technique often used is to apply adversary's power to 'direct' the agents leaving via unexplored ports: When the algorithm specifies that an agent leaves a node via an unexplored port, it chooses the port based on the port labels. Since the adversary can permute the port labels of the unexplored ports, this means that the adversary can choose via which port the agent will leave.

The following theorem gives us the main lower bound, binding the number of agent needed to the maximal degree of the network:

THEOREM 3.3. *There is an n node graph G with the highest degree $\Delta \leq n-4$ such that any algorithm for locating the black hole in arbitrary networks needs at least $\Delta+1$ agents in G. In addition, if $n-4 < \Delta < n$ then any such algorithm needs at least Δ agents.*

PROOF. We first prove the second part for $\Delta = n-1$. Consider the left graph from Figure 1, with the black hole at node 0. The first agent departing the home base 1 is sent via link leading to 0 (and blocked). Therefore there must be an agent moving to the node 2. Again, the first agent departing node 2 via unexplored link is sent to 0; the same is applied for nodes $3, 4, \ldots, \Delta-1$, resulting in $\Delta-1$ agents being blocked before the algorithm can reach the node $n-1$ and terminate. If $\Delta = n-2$, the adversary chooses the same graph, except that node 2 is not connected to 0. If $\Delta = n-3$, also 3 is not connected to 0. It is easy to see that the previous approach can be applied also in these cases.

The case $\Delta \leq n-4$ is played on the right graph from Figure 1. The same arguments as before apply until a node $d-1$ is reached. Again, the first agent leaving $\Delta-1$ via unexplored link is directed towards 0 and blocked. The second such agent is sent to Δ. There are two possibilities: Either the agent departs Δ via unexplored link (and is sent to 0), or it returns back to $\Delta-1$, explores the last unexplored link, arrives to $\Delta+1$ and explores an unexplored link from there. In the first case there are Δ blocked agents and the theorem is proven, in the second case the adversary switches Δ and $\Delta+1$ (as they look the same to the algorithm) and the Δ-th agent is sent to 0 as well. □

In the rest of this subsection we show a lower bound of $\Omega(n^2)$ moves, if the optimal number $\Delta+1$ of agents is used.

Let us call *basic cell* the graph obtained from a mesh $(\Delta+2)\times 2$ by collapsing the outermost pairs of nodes. The game is played on an $\lfloor n/(2\Delta+2)\rfloor$ node ring having each node replaced by a basic cell (see Figure 2). If n is not divisible by $2\Delta+2$, an appropriate number of nodes is connected to the basic cell opposite to the starting node to yield an n node graph G.

Overview description. At any moment of an execution the explored part of the network can be divided into three parts (see Figure 3):

$\mathcal{E}$: The middle, fully explored part, consisting of fully explored basic cells.

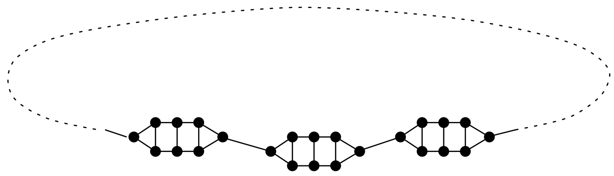

Figure 2: The graph G for $\Delta = 3$.

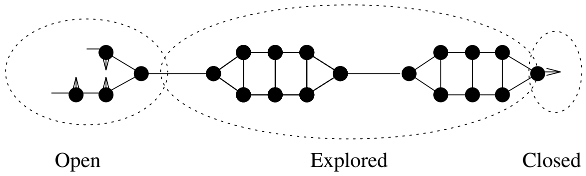

Figure 3: An example of explored part of the network. The arrows correspond to agents in transit.

- $\mathcal{O}$: Open border, containing a single partially explored basic cell.
- $\mathcal{C}$: Closed border, consisting of a single blocked link leading to the next (yet unexplored) basic cell.

The idea is to force the agents to repeatedly migrate between the opposite ends of the explored subgraph, crossing the ever increasing middle part. This migration is forced by "closing" the open border part by blocking the horizontal link to the next basic cell, and simultaneously "opening" the closed border part. Eventually, all unexplored ports in the newly closed part will be explored and there will be no exploration work left there. If the agents do not move to the open part (i.e. they start waiting for the blocked link to become unblocked), the adversary chooses a witness graph in which all active ports lead to the black hole. Once all agents (except the agent blocked at the horizontal link) have migrated, the closed part is opened, the open part is closed and the process is repeated until (almost) the whole network is explored.

Detailed description. In order to reduce the cases we must consider, we give the algorithm the following power: Whenever an agent arrives to a node with an active port, all remaining unexplored ports are revealed (explored) at no cost. Clearly, this modification only strengthens our lower bound.

From the top level perspective, the only action an available agent can do between two stable configurations is to (possibly) move over explored links to some other node and either depart via an unexplored port or start waiting. We describe the adversary's actions as responses to such meta-actions of the algorithm. In particular, we specify how the adversary acts (1) at the beginning of the execution, until a closed and open border parts are formed, (2) in an open border part, until it is closed and (3) in a closed border part, until it is opened. The adversary actions are described as reaction to arrival of an agent to the particular part. Note that although there could be several available agents (e.g. at the beginning of the execution), the adversary can handle them sequentially by allowing only one of them to move, while blocking all others. The adversary reacts to awakening of an agent in the same way as if the agent has just arrived.

Initialization. The adversary chooses a graph with the start node at the right border of a basic cell. The initialization is achieved by sending the first agent to the right and blocking it there. In addition, the two left links are revealed to the algorithm without any cost. (This is done to simplify presentation, clearly it only strengthens our lower bound.) This terminates initialization, with the right border part being closed and the left border part being open. (Consult Figure 4.)

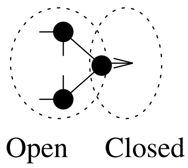

Figure 4: The initial configuration.

Open part. Whenever an agent arrives (either from the closed segment, or being awaken) to an open part and departs anode v via an unexplored port, the adversary directs it to the vertical line incident to v. This ensures that, in general, an open part looks like in Figure 3; the only freedom the algorithm has is to choose whether to explore from the top line or from the bottom one.

When the number of agents traversing the active links in the open part reaches Δ, the adversary switches the open and closed parts as follows.

The open part is closed by blocking the link leading to the next basic cell. Moreover, all agents that have been blocked in this part are unblocked and all remaining unexplored links in this basic cell are revealed at no cost to the algorithm. (This again only strengthens our lower bound.) The resulting fully explored basic cell is then added to the middle (explored) part.

The closed part is opened by unblocking the bridge link and revealing the two incident links of the neighbouring unexplored basic cell.

Closed part. The adversary keeps the bridge link blocked. Since there are no unexplored ports in this part, all agents will either start waiting or leave for the open part.

LEMMA 3.4. *If an open part does not switch to closed, and the number of explored nodes is at most $n-2$, then the algorithm does not locate the black hole.*

PROOF. We show that, in such a case, the adversary can construct a witness graph in which all active ports lead to a single node. This means the algorithm can no longer progress nor locate the black hole among the remaining unexplored nodes.

First note that an open part has two open nodes – the last ones in each horizontal line. Since the open part does not switch to closed, there are at most $\Delta - 1$ active ports there. However, there is at most one active port in the closed part. Since there are no active ports in the middle segment, at most Δ agents are blocked in active ports. The witness graph is constructed from G_e (the currently explored graph – the subgraph of G the algorithm has seen so far) by adding a new node u connected to all active ports; in addition a loop of nodes is connected to G_e at the two open nodes. (Some additional edges may be necessary, see Figure 5.) If there is a single active port, u is also connected to this loop, to ensure 2-connectivity. $\square$

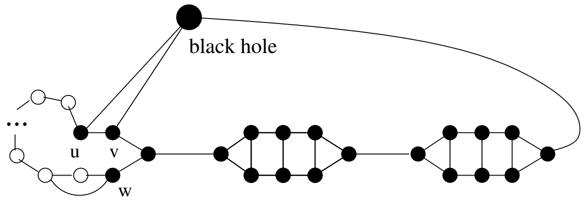

Figure 5: The adversary's response to a stable configuration with 3 agents on active links (one at the closed part, and two on the vertical links from u and v; the vertical and left port of w were unexplored), and others waiting. The white nodes are added in order to make the graph have n vertices.

Now we are ready for the theorem:

THEOREM 3.5. *There exists a graph G such that any $\Delta+1$ agent algorithm working on all 2-connected n-node networks of maximal degree at most $\Delta \geq 3$ needs $\Omega(n^2)$ moves to locate the black hole in G.*

PROOF. During each change (or, flip) of open/closed parts, $\Delta - 1$ agents cross the middle explored part and this part grows by one basic cell. Since $\Delta + 3$ moves are needed to cross a basic cell, the number of moves between the i-th and $i+1$-th flip is at least $(\Delta-1)(i-1)(\Delta+3)$. Summing until there is only the last basic cell left yields the result of the theorem:

$$\sum_{i=1}^{\lfloor n/(2\Delta+2)\rfloor - 1} (\Delta-1)(i-1)(\Delta+3) = n^2/8 + o(n^2)$$

□

Note that the graph we used is of maximal degree 3. The possibility that the graph can have a node of degree Δ is sufficient to drive the cost of a $\Delta + 1$ agent algorithms to $\Omega(n^2)$. Observe also that this lower bound does not hold for $\Delta = 2$ (i.e. ring networks), as there is a black hole location algorithm for rings of complexity $O(n \log n)$ [2].

3.2 Optimal Protocols

In this section we will show that the established lower bounds are tight. Consider the case when $\Delta \leq n-3$; by Theorem 3.3, $\Delta + 1$ agents are necessary. The following protocol shows that they are also sufficient.

Call a node v *expanded* if all its ports are explored or active. Starting from the home base, each node (except the black hole) will be visited and subsequently expanded. $|E(h)| = out_h$ agents are assigned to expand the home base, and (at most) two to each other node; this limitation is imposed so to reduce the total number of movements.

Also to reduce the number of movements, during the expansion process, the agents construct a spanning tree T of G rooted in h; this tree will be used for their travel to and from the home base.

To ensure these properties, the protocol uses a counter $c_a(u)$ ("asking") at each node u; it also uses two other counters at the home base: c_n ("agents-needed") and c_e ("nodes-expanded"). T will be a tree rooted at the home base, spanning all visited nodes, distributively stored at its nodes (at a node, ports corresponding to the tree edges will be marked, as well as the port leading to the parent node). Initially, at the home base, $c_a(h) := c_n := out_h$, $c_e := 0$; h is marked as visited; all agents are waiting at h and T contains just h.

At any time, an agent will be either expanding a node, or searching for a node to expand, or waiting at the home base for an assignment, or destroyed by the black hole.

ALGORITHM 1 (IGNORANCE: $\Delta \leq n-4$).

Waiting for Assignment - Let agent a be waiting at h.

1. If $c_n > 0$ then a sets $c_n := c_n - 1$ and starts searching for a node to expand.

Searching - Let agent a be searching for a node to expand.

1. a traverses[1] T until a node u with $c_a(u) > 0$ is found.
2. It sets $c_a(u) := c_a(u) - 1$.
3. a starts expanding u.

Node Expansion - Let agent a be at v to expand it.

1. It leaves through an unexplored port p of u, making it active. If the incident node v had already been visited, a returns to u to continue its expansion. Otherwise
   (1) a sets $c_a(v) := 2$, updates T by adding v and (u, v), and returns to u (making p explored).
   (2) a goes to h where it sets $c_n := c_n + 2$ (to register that another node has been visited and two agents must be assigned to its expansion);
   (3) It returns [2] to u to continue its expansion.
2. If no port of u is unexplored (a has finished its expansion of u): if u is already marked as expanded, a returns to h to wait; otherwise, a marks u as expanded, returns to h, sets $c_e := c_e + 1$, and waits.

Termination - When $c_e = n-1$, then T includes every node except the black hole; furthermore, at that time, every port p that is still active leads to the black hole. ⊲⊳

THEOREM 3.6. *Algorithm 1 correctly locates the black hole, in $O(n^2)$ moves using $\Delta + 1$ agents.*

[1]A "right-hand-on-the-wall" rule for traversing mazes is used (the ports at a node are cyclically ordered, after arriving by port p, an agent leaves using the next port in the tree), as it does not require additional memory at the agent and/or nodes.

[2]It is sufficient for a to remember the identifier of u and traverse T until u is found. If the nodes did not have unique identifiers at the beginning, the algorithm can be modified to assign to each node a unique identifier when it is first visited.

PROOF. *Correctness.* Since Algorithm 1 uses cautious walk, at most one agent will enter each link incident to the black hole. As the maximum degree of G is at most Δ, at least one agent, say r, never enters a links leading to the black hole. Moreover, since the graph is 2-connected, there is a path from h to every node v in G not passing through the black hole. That means that every node except the black hole will eventually be visited (because at least r survives), and the black hole located.

Cost. The $O(n^2)$ overall cost follows from the fact that the cost of expanding one vertex v is $O(n)$: When v is visited for the first time, the cost ($O(n)$) of returning to the home base to update the "nodes-visited" and "agents-needed" counters is charged not to the node being expanded, but directly to v. At most two agents will then come from the home base to v to expand it, at a cost of $O(n)$. Finally, the cost of expanding v is $O(deg(v))$. □

The Algorithm 1 is not optimal for $n-4 < \Delta \leq n-1$, when Δ agents are sufficient. The main idea of the algorithm for $n-3 \leq \Delta \leq n-1$ is to look at a configuration with exactly $\Delta - 1$ expanded nodes and carefully choose which of the remaining nodes to expand first. The details are quite technical due to the distributed nature of the algorithm and can be found in the full paper.

THEOREM 3.7. *If $n-3 \leq \Delta \leq n-1$, Δ agents can locate the black hole with cost $O(n^2)$.*

The proof of Theorem 3.7 is omitted; it is a quite tedious case analysis of how the remaining $n - \Delta + 1$ unexpanded nodes can be connected to each other and to the graph induced by the already expanded nodes.

As a final remark, we note that the amount of space required by Algorithm 1 is $O(|E(x)|)$ at each node x (to store its portion of the tree and the "asking" counter), plus an additional $O(\log n)$ at h (to store the counters "agents-needed", and "nodes-expanded"). For each agent, $O(\log n)$ bits suffice.

4. COMPUTING WITH SENSE OF DIRECTION

Sense of direction is a well known property of labelled graphs that has been extensively studied in the context of distributed computing and has been shown to have an impact in reducing the communication complexity of several problems (for a survey, see [4]).

In this section we show that, even in a situation of topological ignorance, if there is sense of direction, *two agents* suffice to locate the black hole, regardless of the topology of G. We do so constructively, by designing a solution protocol that requires only two agents. We further prove that the proposed solution is also cost-optimal.

The idea of the algorithm is similar to the one of Algorithm 1: the graph is traversed, expanding the encountered nodes and constructing a spanning tree T of G rooted at h. Unlike the previous solution, only two agents a and b will be employed.

Using only two agents, the crucial and difficult task is to prevent both of them entering the black hole. Clearly, if they are expanding the same node v, no such a danger exists (as only one of the v's neighbours can be the black hole). The problem exists only while expanding different nodes, since both of them might be neighbouring the black hole. Our algorithm makes the agents expand the same node while it is possible. The only situation when this is not the case is when one agent (say b) is exploring the last port p of v (leading to a node w), while the other agent a has already left v searching to expand other nodes (as it had no more work to do at v). In this situation, we must prevent a from entering a port leading to w, since w may contain the black hole. This is achieved using sense of direction: a maintains a variable *danger*, containing the local name of w. Initially, *danger* is set to $\beta_v(w) = \mathbf{c}(p)$. Whenever a moves from node x to node y, *danger* is updated to $\beta_y(w) = \mathbf{d}(\lambda(x,y), \beta_x(w))$. This allows a to check at y whether a port p' leads to w by simply comparing $\beta_y(w)$ with $\mathbf{d}(\mathbf{c}(p'))$. When (if) b explores w, it follows the agent a to inform it that w is no longer dangerous and to join it in expanding the node it is currently expanding. This following is easily accomplished by having a leave a trail (marked ports).

At any time, an agent will be either expanding a node, or searching for a node to expand, or following the other agent, or destroyed by the black hole. In the first two cases, the agent might carry with it information, the variable *danger*, about the dangerous node to be avoided. As before, we will use a spanning tree T of all visited nodes; initially, T contains just h.

ALGORITHM 2 (SD).

Node Expansion - Let a be the agent expanding node u; let z be the current dangerous node (if any) and *danger* $= \beta_u(z)$ be the information known to a (if there is no dangerous node, *danger* $= nil$).

1. If there is an unexplored port p of u not leading to the dangerous node: a leaves u through p making it active, and returns to u (making p explored). If the incident node v had not been previously visited, a updates T by adding v and (u,v) to it.
2. If all ports not leading to the dangerous node are explored:
   (1) If there is a note from b indicating that b is searching, a removes the note, becomes a *follower* and moves following the navigational instructions left by b.
   (2) If there are no notes from b, a marks u as expanded and starts a traversal of T searching for another node to expand, leaving a navigational instruction[3] for b.
3. If one port p, leading to node w, is active and all the others are explored (b is currently exploring link (u,w) and the dangerous node is w), a will: set *danger* $= \beta_u(w) = \mathbf{c}(\lambda(u,w)) = \mathbf{c}(l)$, where l is the label of p; mark u as expanded; start a traversal of T searching for another node to expand; and leave a navigational instruction for b.

[3]a pebble marking the port over which a leaves the current node is sufficient

Searching - Let a be moving from node u to node v searching for a new node to expand. Let z be the current dangerous node (if any); thus, $danger = \beta_u(z)$.

Upon arriving at v:

1. It sets $danger = \beta_v(z) = \mathbf{d}(\lambda(v,u), \beta_u(z))$ updating the information about the dangerous node.
2. If v has been already expanded, a continues its traversal of T searching for another node to expand, and leaves a navigational instruction for b.
3. If v is unexpanded, a starts the expansion of v.

Following - Let a arrive at v while following b. If v is not expanded, a joins b in the expansion; otherwise a follows the navigational command left there by b.

Termination - When the searching agent finds that all nodes in T are expanded, the node identified by *danger* is the black hole. ◁▷

THEOREM 4.1. *In an arbitrary network with sense of direction, the black hole can be located by* two *agents with cost* $O(n^2)$.

PROOF. *Correctness:* We start by showing that at most one agent will enter the black hole. First, note that a following agent does not enter an unexplored port. Second, if both agents are expanding the same node, at most one of them will enter the black hole. Finally, the only possibility for the agents to expand different nodes u and v is when one (say b) is exploring the last unexplored port p of a node u, while a has already left u in a search of the next node v to expand. In such a case, a remembers that p leads to a dangerous node w and will not enter port leading to w.

That means that at least one agent survives; it will eventually explore $n-1$ nodes and terminate, correctly identifying the remaining node as the black hole.

Cost: The total cost for expanding the nodes is $O(m)$ (each link is visited at most twice). The cost of one following is bounded by $O(n)$ (the follower will visit each node at most once). An agent becomes follower only when the last port of a node is explored, hence the total cost following is $O(n^2)$. The cost of finding the next node to expand is $O(n)$ (traversal of the tree T). The total cost of finding is again $O(n^2)$, as for a given node at most one agent (the first one to not find a unexplored port leading to non-dangerous node) leaves it in a search for the next node to expand. □

We will now show that the quadratic bound on the cost is tight in the worst case. We prove it for a particular type of sense of direction called *neighbourhood SD* (note that such lower bound applies to all algorithms assuming nothing about SD), which is equivalent to each node knowing for each port the label of the node on the opposite site (we assume here that the nodes have unique labels, this only strengthens the lower bound) [3].

THEOREM 4.2. *Any two agent algorithm for locating black hole in arbitrary networks with neighbourhood SD has* $\Omega(n^2)$ *cost in the worst-case.*

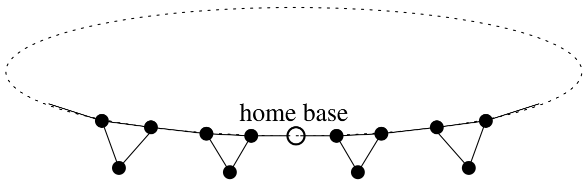

Figure 6: The lower bound graph.

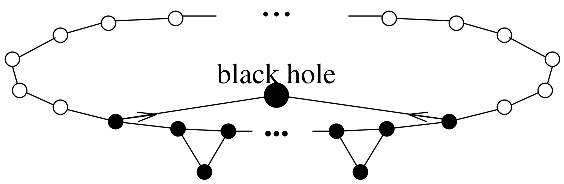

Figure 7: Directing the eagerly exploring agents to the black hole.

PROOF. **(sketch)** The lower bound is based on the observation that, although SD tells an agent what is on the other side of an incident unexplored link, it gives no information about what is deeper inside the unexplored part.

We view the execution as a game between the adversary and the algorithm, played on the graph from Figure 6 (a ring with each vertex except the home base replaced by a triangle). The game proceeds in rounds. At the beginning of each round both agents are leaving the explored part in opposite directions over the "ring edges" (the edges coming from the original ring; the edges of the triangles are called *triangle* edges). The beginning of the first round is reached by the adversary blocking both edges incident to the home base and waiting until both agents depart in the opposite direction (they must do so, otherwise the algorithm does not correctly handle the situation with a black hole incident to the home base).

During a round, the adversary performs two tests: (1) If the left blocked edge is unblocked, will the agent return to the middle node of the explored part before entering a triangle edge of the next unexplored triangle? (2) The same test on the right side. Note that these tests are only virtual, by examining the algorithm. An affirmative answer to both tests means that the agents proceed (without communicating) to explore parts of the graph about which they have no knowledge. In such a case, the adversary forces the algorithm to behave incorrectly by directing both agents to the black hole (see Figure 7).

Hence, at most one agent will enter a triangle edge without returning to the middle of the explored part. The adversary unblocks the agent a that would return there. Then the whole next triangle is revealed and the next unexplored ring edge is blocked (the other agent remains blocked all the time on the ring edge on the opposite side). Eventually, the freed agent will enter the blocked ring edge and the next round will start. (Otherwise the algorithm will not locate the black hole, as there will be more then one unexplored node remaining.)

Note that, since during round p the explored subgraph G_e^p contains $p-1$ triangles, the freed agent performs $\Omega(p)$ moves. Because the arguments above apply for all rounds with at least one unexplored triangle (i.e. $p < n/4 - 1$), summing over all such rounds results in an $\Omega(n^2)$ lower bound. □

5. COMPUTING WITH COMPLETE KNOWLEDGE

In this section we consider the case of an arbitrary system where the agents have complete topological knowledge of (G, λ). In this case, not surprisingly, two agents suffice, even if there is no sense of direction. In fact, with complete topological knowledge, each agent can unambiguously determine the node being visited by the other agent, and thus avoid that node. In other words, we can employ the solution strategy for systems with sense of direction, even if there is no sense of direction. This will yield, by Theorem 4.1, a two-agents solution with $O(n^2)$ cost.

We will now show how, using a different approach and making full use of the available knowledge, the two agents can locate the black hole with at most $O(n \log n)$ moves, which is optimal. The proposed algorithm does not use collaborative "expansion" of the nodes, employed by all previous protocols. Instead, it is based on the notion of individual "working sets" and of "safe" nodes, using an approach similar to the one for ring networks [2]. The *safe* nodes are those in the subgraph induced by the explored links.

Informally, the protocol is as follows. Let G_e be the explored part of the network (i.e., the set of safe nodes); initially it consists only of the home base h. The two agents, a and b, will partition the unexplored area into disjoint subgraphs G_a and G_b, such that there is a link from G_e to G_a and from G_e to G_b (we will show that this is always possible); G_a will be the working set of a, and G_b that of b. Each agent will traverse its working set using *cautious walk* on a tree spanning it, and avoiding the working set of the other agent.

Let b be the first agent to terminate the exploration of its working set; when this happens, b will go to find a. It will do so by: first going to the node u_a from which a departed towards its working set, using optimal path and avoiding G_a; then following the explored links of the tree spanning of G_a used by a; finally reaching the last safe node w_a reached by a.

It will then compute the new subgraph G_u containing all non-safe nodes. If G_u contains a single node, that node is the black hole. Otherwise b computes the new working sets for itself and a; it leaves a note for a at w_a indicating the new working set G_a for a, and goes to explore its new assigned area avoiding the (new) working set of a. When (if) a returns to w_a, it finds the note and starts exploring its new working set.

ALGORITHM 3 (COMPLETE KNOWLEDGE).

Computing the Working Sets - Let a be the agent computing the working sets, and let w be the node where this is done (initially $w = h$).

1. a partitions G_u into disjoint subgraphs, G_a and G_b, such that there is a link (u_a, v_a) from G_e to G_a as well as a link (u_b, v_b) from G_e to G_b. If this is not the first assignment of working sets, then (u_b, v_b) is fixed to be the link from which b last left w.
2. a leaves a note for b informing it of the new working set G_b, and leaves to explore its working set G_a.

Exploring the Working Set -

1. a goes to u_a using the shortest possible route in G_e.
2. Starting from u_a, a explores G_a using *cautious walk* on a tree spanning G_a and avoiding G_b.
3. During the cautious walk, if a finds at node w a note from b informing it of the new working sets G_a and G_b, a will start the exploration of the (new) G_a starting from the (new) u_a (notice that, in this case, $u_a = w$).
4. If a completes the exploration of its working set, it will search for b to recompute the working sets.

Searching for the other agent - Let a be searching for b.

1. a goes to u_b (the node from which b departed towards its working set) using an optimal path avoiding G_b.
2. Starting from u_b, a follows the safe links of the tree spanning G_b used by b, until it reaches the last safe node w reached by b.
3. a computes the new working sets.

Termination - When computing the working set, if G_u contains a single node, that node is the black hole. ◁▷

The following lemma shows that the agents can indeed perform the actions specified in step 1. of part *Computing the Working Sets* of Algorithm 3.

LEMMA 5.1. *Let $G = (V, E)$ be a graph of vertex connectivity at least two and let $\{V_e, V_u\}$ be a partition of V such that G_e (the graph induced by V_e) is connected. Let $(u, v) \in E$, where $u \in V_e$ and $v \in V_u$, and let $1 \leq k < |V_u|$. Then V_u can be partitioned into V_a and V_b such that $|V_b| = k$, $v \in V_a$ and there is a link between V_e and V_a. (See figure 8)*

Existence of a link between V_e and V_a means that a can safely reach the part (subgraph G_a induced by V_a) it has to explore. The lemma is more general then strictly needed, as only the case $k = |V_u|/2$ is used by the algorithm.

The proof of Lemma 5.1 is omitted; it is technical and pure graph theory.

THEOREM 5.2. *The black hole can be located by two agents with full topological knowledge in arbitrary networks of vertex connectivity 2 with cost $O(n \log n)$, and this is optimal.*

PROOF. **(sketch)** *Correctness:* Since G_a and G_b are disjoint, one agent always survives. This, together with the fact that agents never wait for each other, ensures progress. The Lemma 5.1 used with $k = |V_u|/2$ allows to halve in each round (iteration of the main loop) the size of the unexplored area. This means after $O(\log n)$ rounds the algorithm terminates.

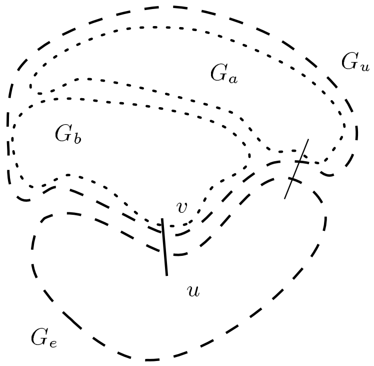

Figure 8: Splitting the unexplored subgraph G_u into G_a and G_b.

Cost: Note that the each action an agent performs within one round is either (1) local computation (2) moving to another node of G_e or (3) traversing a spanning tree of some subgraph of G_e. Clearly, the cost of each of these actions is $O(n)$. Since the number of such actions per round is constant, and there are only two agents, the total cost of one round is $O(n)$. As the number of rounds is $o(\log n)$, the total cost of Algorithm 3 is $O(n \log n)$.

The cost optimality follows from the $\Omega(n \log n)$ lower bound for ring networks by [2]. $\square$

Note that the analysis above uses very weak arguments to prove that the cost of one round is $O(n)$. Nevertheless, they can not be improved in general – the cost of communication (an agent finding the other agent, and leaving a message) in the ring network is $O(|G_e|)$, which over all rounds sums to $O(n \log n)$.

6. CONCLUDING REMARKS

We have studied conditions under which a team of autonomous asynchronous mobile agents can identify the location of the harmful host. We have shown that the size and the cost of an optimal solution depend on the a priori knowledge the agents have about the network, and on the consistency of the local port labellings. In particular, we have provided tight bounds when computing with topological ignorance, in presence of sense of direction, and with complete topological knowledge.

Topological ignorance and complete topological knowledge represent the two extreme levels of possible knowledge. Interesting natural questions are about the intermediate levels. For example, can two agents locate the black hole with $O(n \log n)$ cost with less than complete knowledge?

We have shown that the presence of sense of direction allows two agents to overcome any ignorance about the topology. Are there other consistency properties capable of similar results?

7. REFERENCES

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

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

[3] P. Flocchini, B. Mans, and N. Santoro. Sense of direction: definition, properties and classes. *Networks*, 32(3):165–180, 1998.

[4] P. Flocchini, B. Mans, and N. Santoro. Sense of direction in distributed computing. *Theoretical Computer Science*, (to appear), 2002.

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

[6] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In *Proc. of Conf on Mobile Agent Security*, LNCS 1419, pages 92–113, 1998.

[7] F. Hohl. A framework to protect mobile agents by using reference states. In *Proc. of the 20th Int. Conf. on Distr. Computing Systems (ICDCS 2000)*, 2000.

[8] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.

[9] S. Ng and K. Cheung. Protecting mobile agents against malicious hosts by intention spreading. In *Proc. 1999 Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 725–729, 1999.

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

[11] T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. In *Proc. of Conf on Mobile Agent Security*, LNCS 1419, pages 44–60, 1998.

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

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