Identifying Hostile Nodes in Networks Using Mobile Agents

Euripides Markou*

Abstract

In distributed mobile computing environments one of the most pressing concerns is security. Two are the most important security threats: a malicious mobile process which can move along the network and a stationary harmful process which resides at a host. One of the most studied models for stationary harmful processes is the *black hole* which has been introduced by S. Dobrev, P. Flocchini, G. Prencipe and N. Santoro in 2001 ([25]). A *black hole* is a harmful node in the network that destroys any *mobile agent* visiting that node without leaving any trace. The objective of the *Black Hole Search* problem is to identify the black hole without destroying too many agents and the main effort is to discover the minimal hypotheses under which it can be solved. Another effort has to do with producing fastest black hole search schemes. The problem has been initially investigated in *asynchronous* networks and introduced later in *synchronous* networks. In synchronous networks the most studied issue was that of achieving time optimal black hole search schemes. Recently it has been investigated in synchronous networks under very weak models (e.g., using agents with only constant memory) in which it is unsolvable in asynchronous networks. In this survey we discuss the computational issues and present algorithmic techniques for this problem. Since searching for a black hole in a network is closely related to *exploration*, *rendezvous* and *leader election* problems, the techniques presented here have a much wider range in distributed computing.

Keywords: Distributed Algorithms, Fault Tolerance, Black Hole Search, Networks, Graph Exploration, Mobile Agents, Computational Complexity, Deterministic Finite Automata.

*Department of Computer Science & Biomedical Informatics, University of Central Greece, Lamia, Greece, emarkou@ucg.gr.

1 Introduction

In distributed computing, the research focus is on the computational and complexity issues of systems composed of autonomous computational entities interacting with each other to solve a problem or to perform a task. While traditionally these autonomous computational entities have been assumed to be static, recent advances in a variety of fields, ranging from robotics to networking, have motivated the community to address the situation of *mobile entities*.

1.1 Mobile agents

Mobile agents' ability to adapt dynamically and execute asynchronously and autonomously brings potential advantages in terms of efficiency, fault-tolerance, flexibility and simplicity. However, there are many who believe that opening up your network to the "invasion" of potentially harmful mobile agents is not worth the advantages they may provide. In order to evaluate, at least theoretically, the claimed advantages, especially those concerning efficiency and fault-tolerance, we need an algorithmic model in which we can analyze the behavior of mobile agent algorithms and prove given complexity bounds.

Recently, an increasing number of investigations are being carried out on the computational and complexity issues arising in systems of mobile entities that can move in a spatial universe. Depending on the nature of the spatial universe, there are two basic settings in which mobile entities are being investigated. The first setting, called sometimes continuous, is when the universe is a region of the $2D$ (or $3D$) space. This is for example the case of robotic swarms, mobile sensor networks, mobile robotic sensors, etc. (as in, e.g., [1, 21, 72]). The second setting, sometimes called graph world or discrete universe, is when the universe is a simple graph; this is for example the case of mobile agents in communication networks (as in, e.g., [20, 45]). In both settings, the research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost. In particular, a central question is to determine what minimal hypotheses allow a given problem to be solved.

Mobile agents in networks can be thought of as autonomous, goal-oriented software entities that can transport their state from one computational environment to the next and resume their execution in the new environment, thus remaining active as they migrate between computers. This makes them a powerful tool for implementing distributed applications in computer networks. The agents are generally modeled as automata that move on a network modeled as a graph. The first known algorithm designed for graph exploration by a mobile agent, modeled as a finite automaton, was introduced by Shannon [69] in 1951. Since then, several papers have been dedicated to the feasibility of graph exploration by one or

more agents. Important properties that have been considered by researchers are as follows:

- Whether or not the agents are distinguishable, i.e., if they have distinct identities. Anonymous agents are limited to executing precisely the same algorithm, while agents with distinct identities have the potential to execute different algorithms.

- The size of memory an agent has (e.g., a function of the size of the network or constant memory) and the knowledge an agent has about the network it is on (e.g., a map or the size of the network) and about the other agents.

- The method through which the agents communicate. For example they may have the ability to read the state of other agents residing at the same node. Or they can communicate via a shared memory space provided at each node (usually called *whiteboard*), or via message passing, or by leaving indistinguishable markers at nodes or edges (often called *tokens* or *pebbles*).

- Whether the nodes and/or edges of the network are distinguishable by an agent (as in, e.g., [33, 63]) or not (as in, e.g., [8]). The outgoing edges of a node are usually thought of as distinguishable but an important distinction is made between a *globally consistent* edge-labeling (thus giving the agents the ability to navigate) versus a *locally independent* edge-labeling (in that case, even in restricted topologies like a torus, the agents cannot navigate based only on the edge labeling). If the labeling satisfies certain coding properties it is called a *sense of direction* [41].

- How networks deal with time. In a synchronous network there exists a global clock available to all nodes. This global clock is inherited by the agents. In particular it is usually assumed that in a single step an agent arrives at a node, performs some calculation, and exits the node and that all agents are performing these tasks 'in sync'. In an asynchronous network such a global clock is not available; the speed with which an agent computes or moves between nodes, while guaranteed to be finite, is not a priori determined (as in, e.g., [64]).

- Whether networks and/or agents may be faulty or not. For example crash or omission failures, faulty edges, Byzantine failures (where a faulty agent and/or node behaves arbitrarily and potentially maliciously) have been considered.

All the above properties have turned out to greatly effect the solvability and efficiency of solution of a number of problems in distributed computing and have been shown to be important for the study of mobile agents as well.

For a given choice of agent plus network model there are a number of important resources for which one can define a complexity measure. Measures that reflect the time and bandwidth efficiency of a given algorithm are of paramount concern. The total bandwidth consumed by an agent depends upon its size (memory) as well as the number of moves it makes during an execution of its algorithm. Generally the size (memory) of an agent is identified with the number of bits required to encode its states, i.e., it is proportional to the log base two of the number of possible states (as in, e.g., [22, 46]). Depending on its memory, an agent may be able to make and/or store a map of the network, store the network size, count the number of nodes visited, etc. If the agent sends messages then the size and number of these messages must also count towards any measure of its bandwidth. Other complexity measurements of interest include the size of shared memory required at each node assuming the agents communicate via shared memory, the number of random bits used by a randomized agent and the number and kind of faults an algorithm can successfully deal with.

1.2 Exploration in unsafe networks

One of the main concerns in distributed mobile computing has to do with the security of both agents and hosts [13, 49, 62, 68, 9, 52]. The case of harmful mobile agents (representing mobile viruses that infect any visited network site) has been considered in the literature. A crucial task is to decontaminate the infected network; this task is to be carried out by a team of system agents (the cleaners), able to decontaminate visited sites, preventing any reinfection of cleaned areas. This problem is equivalent to the one of capturing an intruder moving in the network. Results on this and related problems have appeared in [2, 7, 35, 36, 44, 59, 71].

Protecting agents from "host attacks", i.e., harmful items stored at nodes of the network, has become almost as urgent as protecting a host, i.e., a node of the network, from an agent's attack. Various methods of protecting mobile agents against malicious hosts have been discussed (as in, e.g., [50, 51, 61, 67, 68, 73]). In [43] results for both harmful nodes (especially for asynchronous networks) and harmful agents are surveyed.

Recently, the exploration problem has been studied in unsafe networks which contain malicious hosts of a highly harmful nature, called *black holes*. A black hole is a node which contains a stationary process destroying all mobile agents visiting this node, without leaving any trace. In the *Black Hole Search* problem the goal for the agents is to locate the black hole within finite time. In particular, at least one agent has to survive knowing all edges leading to the black hole. The problem has been introduced by S. Dobrev, P. Flocchini, G. Prencipe and N. Santoro in 2001 ([25]). The only way of locating a black hole is to have at least one agent visiting it. However, since any agent visiting a black hole is vanished

without leaving any trace, the location of the black hole must be deducted by some communication mechanism employed by the agents. Four such mechanisms have been proposed in the literature: a) the *whiteboard* model in which there is a whiteboard at each node of the network where the agents can leave messages, b) the *'pure' token* model where the agents carry tokens which they can leave at nodes, c) the *'enhanced' token* model in which the agents can leave tokens at nodes or edges, and d) the time-out mechanism (only for synchronous networks) in which one agent explores a new node and then (after a pre-determined fixed time) informs another agent who waits at a safe node.

The most powerful inter-agent communication mechanism is having whiteboards at all nodes. Since access to a whiteboard is provided in mutual exclusion, this model could also provide the agents a breaking symmetry mechanism: If the agents start at the same node, they can get distinct identities and then the distinct agents can assign different labels to all nodes. Hence in this model, if the agents are initially co-located, both the agents and the nodes can be assumed to be non-anonymous without any loss of generality.

While the whiteboard model is commonly used in unsafe networks, the token model has been mostly used in safe networks' exploration. The 'pure' token model can be implemented with $O(1)$-bit whiteboards, for a constant number of agents and a constant number of tokens, while the 'enhanced' token model can be implemented having a $O(\log \Delta)$-bit whiteboard on each node, where Δ is the maximum degree in the graph. In the whiteboard model, the capacity of each whiteboard is usually assumed to be of at least $\Omega(\log n)$ bits, where n is the number of nodes of the network.

In asynchronous networks and given that all agents have memory and initially start at the same safe node, the Black Hole Search problem has been studied under the whiteboard model (e.g., [23, 27, 28, 29, 48, 4, 6, 5, 40]), the 'enhanced' token model (e.g., [24, 30, 70]) and the 'pure' token model in [37, 38]. It has been proved that the problem can be solved with a minimal number of agents performing a polynomial number of moves. In an asynchronous network the number of the nodes of the network must be known to the agents otherwise the problem is unsolvable ([28]). If the graph topology is unknown, at least $\Delta + 1$ agents are needed, where Δ is the maximum node degree in the graph ([27]). Furthermore the network should be 2-connected. It is also not possible to answer the question of *whether* one black hole exists in the network. If the agents have a map of the network or at least a *sense of direction* ([41, 42]) and can use whiteboards, then two agents with memory suffice to solve the problem.

In asynchronous networks with dispersed agents (not initially located at the same node), the problem has been investigated for the ring topology ([26, 28]) and for arbitrary topologies ([39, 12]) in the whiteboard model while in the 'enhanced' token model it has been studied for rings ([31, 32]) and for some interconnected

networks ([70]).

The consideration of synchronous networks makes a dramatic change to the problem. Now two co-located distinct agents with memory can discover one black hole in any graph for which they have a map by using the time-out mechanism, without the need of whiteboards or tokens. Moreover the network does not have to be 2-connected anymore, as in asynchronous networks, and furthermore it is now possible to answer the question of *whether* a black hole actually exists or not in the network. Hence, with co-located distinct agents, the issue is not the feasibility but the time efficiency of black hole search. The issue of efficient black hole search has been studied in synchronous networks without whiteboards or tokens (only using the time-out mechanism) in [14, 15, 18, 19, 55, 56] under the condition that all distinct agents start at the same node. However when the agents are scattered in the network, the time-out mechanism is not sufficient anymore.

In most papers studying the Black Hole Search problem under a token model apart from [37, 38, 11, 10, 3], the authors have used the 'enhanced' token model with agents having non-constant memory. The weakest 'pure' token model has been used in [37, 38, 3] for co-located agents with non-constant memory in asynchronous networks. The first results for scattered agents with constant memory and 'pure' tokens appeared in [11] for synchronous unoriented rings and [10] for synchronous oriented tori. In [10] it has been proven that three scattered agents with constant memory and with two tokens each can locate the black hole in any synchronous oriented torus.

As in many cases that deal with the analysis of algorithms for certain problems, an approach in proving correctness of an algorithm for the Black Hole Search problem, showing upper bounds on time needed for discovering the black hole or proving infeasibility of the problem under a certain model, often uses the notion of an *adversary*. The analysis of the Black Hole Search problem under a model can then be considered as a game between a proposed algorithm and an *adversary* who uses its power to make the algorithm fail. The weakest the model is, the more powerful the adversary is. The adversary can choose all the parameters of the instance of the problem for which the algorithm has no knowledge. For example, the adversary can decide for the initial positions of the agents, the location of the black hole and, depending on the model, the topology or the size of the network, the number of the agents, the ports' labeling, and (in asynchronous networks) the time that an agent needs to cross an edge. A correct algorithm should work of course for any options of the adversary. The problem is infeasible when the adversary has an option to make any algorithm fail.

1.3 Structure of the survey

The rest of the article has been organized as follows. In Section 2 we present some results for two or more co-located agents with memory in asynchronous ring topologies with whiteboards at nodes. Then we move on to synchronous networks and discuss the problem for two co-located agents with memory using only the time-out mechanism in a synchronous tree topology (Section 3) presenting a time optimal algorithm for special trees and discussing an approximation algorithm for arbitrary trees. In Section 4 we study the problem in synchronous arbitrary graphs. We present NP-hardness results for constructing time optimal *Black Hole Search schemes* (i.e., a pair of sequences of edge traversals for each of the two agents) for two versions of the problem. We also give approximation algorithms for arbitrary graphs and present APX-hardness results which show that an optimal time Black Hole Search scheme is not approximable in polynomial time within more than a constant factor unless $P = NP$. In Section 5 we study the problem for scattered agents having only constant memory (DFAs) and carrying 'pure' tokens. We discuss the results in ring and torus topologies. We end this survey in Section 6 discussing important tools and their impact on the black hole search problem and presenting some other models of hostile-node identification problems.

2 Black-Hole Search in an Asynchronous Ring

The Black-Hole Search (BHS) problem has been introduced in [25] (see also the full version of the paper in [28]) where it was studied for asynchronous ring topologies. We present here some results from this seminal paper. We first discuss the model used and give lower bounds on the number of agents and time-complexity. We then give two algorithms that solve the problem using two or more co-located mobile agents with memory using whiteboards.

2.1 Model and basic lower bounds

We consider two anonymous co-located agents with memory. The ring is anonymous, asynchronous and consists of n nodes. On each node there is a *whiteboard* where the agents can leave messages. The access on a whiteboard is done using mutual exclusion and hence the agents can acquire distinct identities by the order in which they access the whiteboard (e.g., the first agent accessing the whiteboard creates a counter on it, initializes it to 1 and gets this identity, while the next agent increases the counter and take its value as its identity). Although the ring is anonymous, the distinct identities assigned to agents, allow them to agree on the clockwise direction. The goal is that after a finite time at least one agent should

report back to the starting node the exact location of the black hole.

It is trivial to see that, if there is only one agent, the BHS problem is unsolvable since the only agent would necessarily vanish into the black hole. Hence at least two agents are needed to locate the black hole.

Due to network asynchrony, it is impossible to distinguish between a 'slow' link and a link leading to a black hole. This observation gives us the following two lemmas.

Lemma 1. *([28]) The problem of whether there exists or not a black hole in an asynchronous network is unsolvable.*

Lemma 2. *([28]) It is impossible to locate the black hole if the size of the network is unknown.*

The following theorem gives us a lower bound on the number of steps the agents need to locate the black hole in a ring.

Theorem 1. *([28]) Any algorithm needs at least* $2n - 4$ *moves to find the black hole in a ring regardless of the number of agents available.*

Proof. In order to report the position of the black hole back to the starting node h, the agents need to receive information from any other node apart from the node $\mathcal{B}$ containing the black hole. This means that every node apart from $\mathcal{B}$ has to be visited by at least one agent. Suppose that the black hole resides at a distance $n-1$ clockwise. Any agents traveling in counter-clockwise direction would vanish in the black hole but (due to the asynchronous network) the remaining agents cannot decide whether the black hole is at $\mathcal{B}$ or the link to node $\mathcal{B}$ is 'slow'. Hence an agent must travel clockwise to node $n-2$ and then an agent must report back to h. Therefore a total number of $2n-4$ steps must be taken. ■

2.2 A protocol for two agents

We denote with U and E the unexplored and explored area respectively. We also denote with U^L and U^R the continuous unexplored area adjacent counter-clockwise and clockwise respectively from the explored area. A basic tool which is used in the algorithm is the *Cautious Walk* (introduced in [25, 28]):

Consider an agent situated at a node v_0 adjacent to an unexplored node v_1. The agent explores a previously unexplored area $U_k =< v_1, v_2, ..., v_k >$ in the following way:

Cautious Walk:

- before leaving a node v_i going to a node v_{i+1}, the agent marks the port leading from v_i to v_{i+1} as *active*,

- immediately after visiting v_{i+1}, the agent returns to v_i, and marks the port leading from v_i to v_{i+1} as *safe*,
- the agent checks for messages at v_i and (if such a message exists) re-assigns to itself an unexplored area U'_k which has to discover and repeats from the start.

The agents know the size n of the ring. They follow Algorithm 1. A high level description of the algorithm is the following:

Suppose the agents start at node h. Using mutual exclusion they write at the whiteboard of node h and get distinct identities as described in the beginning of the section. They also agree on the clockwise orientation. Then they divide the unexplored area U with $|U| = n - 1$ into two disjoint paths U^L and U^R with $|U^L| = \lceil \frac{n-1}{2} \rceil$ and $|U^R| = \lfloor \frac{n-1}{2} \rfloor$. Agent 1 explores the area U^L and agent 2 explores the area U^R using *Cautious Walk*. Since there is exactly one black hole, and the two sets are continuous and disjoint, after a finite time exactly one of the agents will finish with the exploration. Suppose without loss of generality that agent 1 finishes. Then agent 1 traverses back through the explored area until it meets a node u whose port leading to a node v has not been marked as safe. At that point agent 1 updates explored area and divides the updated unexplored area into new continuous and disjoint paths U^L and U^R having U^L starting at a node situated counter-clockwise from h and U^R starting at node v situated clockwise from h. Assigns U^L to itself and U^R to agent 2 and leaves this message at node u. Now agent 1 traverses the explored area and explores its new assigned path U^L. Agent 2 either travels through a slow link (not having explored its assigned area) or vanished in the black-hole. In the first case agent 2 will first return at node u, will check for messages and update its assigned area for exploration. The agents repeat this procedure until exactly one of them vanishes into the black-hole. Then the other agent which repeats the procedure, will eventually come up with an explored area of size $n - 1$. At that point it knows the exact location of the black-hole.

Theorem 2. *([28]) Algorithm 1 locates the black-hole in an asynchronous ring of n nodes with two co-located agents within $2n \log n + O(n)$ moves.*

A natural question is the following: Can we decrease the time needed for Black Hole Search if we have $k \geq 2$ available co-located agents?

2.3 The case of $n - 1$ agents

Algorithm 2 shows that $n - 1$ co-located agents can locate the black hole in an asynchronous ring of n nodes within $2n - 4$ moves.

Algorithm 1 (2 *agents with memory in an asynchronous ring with whiteboards*)

1: The agents get distinct identities and agree on the clockwise orientation
2: Let $X := h$
3: **while** $|E| < n - 1$ **do**
4: Divide U into two continuous disjoint parts U^L (starting counter-clockwise of node X) and U^R (starting clockwise of node X) of almost equal sizes
5: Agent 1(2) leaves a message at X saying that she will explore $U^L(U^R)$
6: Agent 1(2) explores $U^L(U^R)$ using *Cautious Walk*
7: Agent 1(2) traverses clockwise (counter-clockwise) the explored area until it reaches a node u with a not safe port
8: Update E and U
9: Let $X := u$
10: **end while**
11: Report the black-hole location

Algorithm 2 ($n-1$ *agents with memory in an asynchronous ring with whiteboards*)

1: Agents get distinct identities from the set $\{1, 2, ..., n - 1\}$ and agree on the clockwise orientation
2: Agent i travels $i - 1$ edges in clockwise direction
3: Agent i travels $n - 2$ edges in counter-clockwise direction
4: Agent i returns to homebase traveling in clockwise direction
5: Agent i reports that the black hole resides at a distance i clockwise

In Algorithm 2, it should be clear that among the $n - 1$ agents, exactly one will finish the algorithm, while all the other $n - 2$ agents will vanish into the black hole. Also notice that the $n - 1$ agents use the whiteboard only at the starting node in order to assign distinct identities to themselves.

Theorem 3. *([28]) Algorithm 2 lets* $n - 1$ *co-located agents find the black hole in an asynchronous ring of n nodes within* $2n - 4$ *moves.*

Proof. Suppose that the black hole resides at a node $\mathcal{B}$ which is $\mathcal{B}$ steps clockwise from node 0 (where the agents start). Then agent with label $\mathcal{B}$ travels $\mathcal{B} - 1 + n - 2 + n - (\mathcal{B} + 1) = 2n - 4$. ∎

3 Black Hole Search in a Synchronous Tree

The first results on the Black-Hole Search problem in *synchronous* networks appeared for two agents operating in tree topologies in [17] (see also the full version of the paper in [19]). We present here the model which was used, give lower

bounds on BHS schemes in synchronous trees and describe a fastest algorithm for the problem in some special tree topologies.

3.1 Model and basic lower bounds and tools

We consider a labeled tree T rooted at node s which is the starting node of two agents, and is assumed to be safe (s is not a black hole). Agents are synchronous, have memory, a map of the tree and distinct labels. They can communicate (i.e., reading each-other's states) only when they meet at a node (and not, e.g., by leaving messages at nodes). We assume that there is at most one black hole in the network. The goal is to design an algorithm for the agents which produces a black hole search scheme (*BHS-scheme*) for the input (T, s) (i.e., a pair of sequences of edge traversals (moves) of each of the two agents). Upon completion of the BHS-scheme there should be at least one surviving agent which either knows the exact location of the black hole or knows that there is no black hole in the tree. The surviving agents must return back to s to report this information. An agent can move along the edges of the tree (each move takes one time unit) or can wait at a node.

The time of a black hole search scheme is the number of time units until the completion of the scheme, assuming the worst-case location of the black hole (or its absence, whichever is worse). It is easy to see that the worst case for a given scheme occurs when there is no black hole in the tree or when the black hole is the last unvisited node, both cases yielding the same time. A scheme is called *fastest* for a given input if its time is the shortest possible for this input. We emphasize here that the time of a black hole search scheme should not be confused with the time complexity of an algorithm producing such a scheme.

Since there is at most one black hole in the tree, in any BHS-scheme all nodes of the tree should be eventually visited in the worst case (e.g., when there is no black hole). Hence all edges of the tree should be traversed (*explored*).

When a meeting occurs the agents exchange information about the explored territory (i.e., the set of explored edges). The sequence of steps of a BHS-scheme between two consecutive meetings is called a *phase*. Since an unexplored edge could be incident to a black hole we have:

Lemma 3. *([19]) In a BHS-scheme, an unexplored edge cannot be traversed by both agents.*

If one of the agents a attempts to traverse more than one unexplored edges before meeting with the other agent b then a may vanish in the black hole (placed by an adversary somewhere in a's path) and b does not have enough information to decide for the correct location of the black hole. Thus:

Lemma 4. *([19]) During a phase of a BHS-scheme an agent can traverse at most* one *unexplored edge.*

Hence in a BHS-scheme, an edge can be explored only in the following way: an agent traverses this edge and then a meeting is scheduled (somewhere within the previously known explored territory) where the agents exchange information about the explored territory. Whether the meeting occurs or not (in the latter case the agent vanished in the black hole) the edge becomes explored. Therefore an unexplored edge could be explored in the next phase only if it is adjacent to the explored territory (which stays connected).

Lemma 5. *([19]) At the end of each phase, the explored territory is increased by one or two edges, or the black hole is found.*

We define a *1-phase* to be a phase in which exactly one edge is explored. Similarly, we define a *2-phase* to be a phase in which exactly two edges are explored. In view of Lemma 5, every phase is either a 1-phase or a 2-phase.

Let (u, v) be an unexplored edge with node u being incident to a previously explored edge (or $u \equiv s$). Suppose that the two agents are at u at time t. A way of exploring exactly one edge in a phase is the following:

Procedure `Probe(`v`)`: one agent traverses edge (u, v) and returns to node u to meet the other agent who waits. If they do not meet at step $t + 2$ then the black hole is at v.

We also define a procedure that the two agents could follow to explore two new edges in a phase. Let $(u_1, v_1), (u_2, v_2)$ be two unexplored edges with nodes u_1, u_2 being incident to previously explored edges (or $u_1 \equiv u_2 \equiv s$). Suppose that one of the agents is at u_1 and the other agent is at u_2 at time t. The definition of the procedure is the following:

Procedure `Split(`v_1, v_2`)`: One of the agents traverses edge (u_1, v_1) and then returns to u_1 while the other one traverses edge (u_2, v_2) and then returns to u_2. Then both agents traverse the path $< u_1, u_2 >$ (which should be totally within the explored territory since the explored territory is always connected) from different directions until they meet at a node as soon as possible. Let $dist(u_1, u_2)$ denote the number of edges in the path from node u_1 to node u_2. If the agents do not meet at step $t + \lceil \frac{dist(u_1,u_2)}{2} \rceil + 2$ then the black hole has been found.

Both above procedures resemble the `Cautious-Walk` procedure from Section 2 in the sense that in both procedures the agents explore at most one edge each and then they return to give a message to each-other. Two agents with a map can easily discover the exact location of the black hole (or decide that there is no

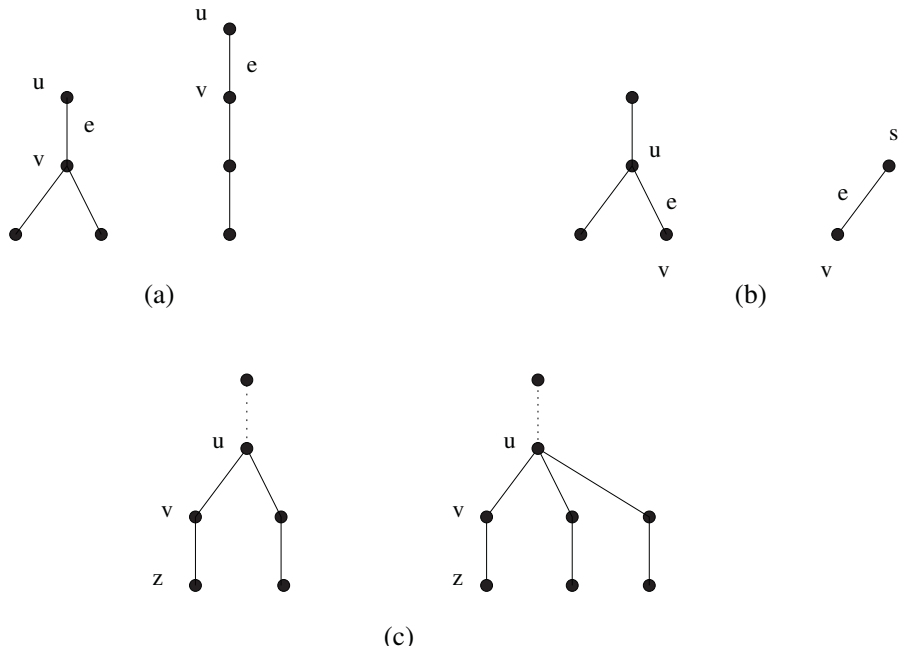

Figure 1: In (a) e is a red edge. In (b) e is a green edge. In (c) all solid edges are blue.

black hole) in any tree within finite time (e.g., exploring the tree in a Depth First Search traversal using Procedure `Probe`). In fact this simple algorithm guarantees a 4 approximation ratio on the fastest BHS scheme for any arbitrary tree (see at the end of the section for details). The authors ([19]) give an algorithm which uses both procedures `Probe` and `Split` and solves the BHS problem for a special family of trees in the fastest possible time. Before describing this algorithm we give a general lower bound on the time needed by any algorithm to solve the problem on any tree.

3.2 A lower bound on the time needed in any tree

Let T be a tree rooted at the starting node s and $e = (u, v)$ be an edge of T (with v being a child of u). Consider the following coloring which creates a partition of the edges of the tree.

- assign red color to edge e if node v has at least two descendants,
- assign green color to edge e if v is a leaf and exactly one of the following holds: $u = s$ or the edge (w, u) is a red edge (where w is the parent of u),
- assign blue color to edge e if it has none of the above properties

Red, green and blue edges are shown in Figure 1.

Let $e = (u, v)$ and $e' = (v, z)$ be two blue edges as shown in Figure 1c, i.e., v is a child of u and z is a leaf and the unique child of v. We call the set of these two edges a *branch*. The set of all branches of blue edges with upper node u is called a *block*. In view of Lemmas 3 and 4 it can be easily proved that:

Lemma 6. *([19]) In any BHS-scheme, the following holds: a green edge has to be traversed by the agents at least* 2 *times, a red edge has to be traversed at least* 6 *times and a branch of blue edges requires a total of at least* 6 *traversals.*

Since in any BHS-scheme, each of the two agents can do at most one traversal in every time unit, the following lemma holds:

Lemma 7. *([19]) Any BHS-scheme requires at least* 3, 1 *and* 3r *time units for the traversals of a red edge, a green edge and a block of r branches of blue edges, respectively.*

3.3 A time optimal algorithm for a family of trees

Consider the family $\mathcal{T}$ of rooted trees with the following property: any internal node of a tree in $\mathcal{T}$ (including the root) has at least 2 children. Trees in $\mathcal{T}$ will be called *bushy trees*. For these trees, searching for a black hole can be efficiently parallelized by an appropriate use of the Procedure `Split`. This enables us to get an optimal algorithm (Algorithm 3) for this class of trees.

Let T be a bushy tree with root s and let u be an internal node of T. The *heaviest child* $v = H(u)$ (resp. *lightest* child $v = L(u)$) of u is defined as a child v of u such that the subtree $T(v)$ rooted at v (which is also a bushy tree) has a maximum (resp. minimum) height among all subtrees rooted at children of u with ties broken arbitrarily. Notice that $H(u)$ and $L(u)$ can be computed by the agents for each node u in linear time.

The high-level description of Algorithm 3 is the following. Let m be the meeting point of the two agents after a phase (initially $m \equiv s$).

- Explore any pair of unexplored edges (m, x), (m, y) with upper node m by executing Procedure `Split(x,y)`, leaving edge $(m, L(m))$ last.

- If there is one unexplored edge with upper node m (which must be $(m, L(m))$) then one of the agents explores this edge while the other one explores another 'close by' unexplored edge (if any) again using `Split`. If edge $(m, L(m))$ is the last unexplored edge in the tree, explore it by executing `Probe(L(m))`.

- If all edges with upper node m are explored, explore similarly as before any unexplored edges incident to the children of m and to ancestors of m.

Function `Explore-only-child(`v`)` takes as input the current node v where both agents reside and returns the new meeting point after the exploration of edge $(v, L(v))$. The description of the procedure is the following:

Algorithm 3 (2 *agents with memory and a map in a synchronous 'bushy' tree*)

```
 1: next := s;
 2: repeat
 3:    v := next;
 4:    for every pair of unexplored edges (v, x), (v, y) with upper node v do
 5:       Split(x, y), so that edge (v, L(v)) is explored last;
 6:    end for
 7:    if there are still unexplored edges in the tree then
 8:       case 1: every edge incident to v has been explored:
 9:          case 1.1: there is an unexplored edge incident to a child w of v:
10:             next := w;
11:          case 1.2: every edge incident to any child of v is explored:
12:             let t be the parent of v;
13:             next := t;
14:          walk to node next;
15:       case 2: there is an unexplored edge (v, z) incident to v:
16:       (* must be z = L(v) *)
17:          next := Explore-only-child(v);
18:    end if
19: until every edge has been explored
20: walk to node s and report the location of the black hole;
```

- If there is an unexplored edge incident to a child w of v, $w \neq L(v)$, then the agents explore edge $(w, H(w))$ together with edge $(v, L(v))$ calling Procedure `Split(H(w),L(v))`. The new meeting point is w.

- If every edge incident to any child w of v, different from $L(v)$, is explored and edge $(v, L(v))$ is not the last unexplored edge in the tree, then find the deepest ancestor a of v having a descendant incident to an unexplored edge (excluding $L(v)$); the agents explore edge $(D(a), H(D(a)))$ (where $D(a)$ is the closest descendant of a with incident unexplored edges), together with edge $(v, L(v))$, by `Split(H(D(a)),L(v))`; the new meeting point is $D(a)$.

- If edge $(v, L(v))$ is the last unexplored edge in the tree then explore it by calling `Probe(L(v))`; the new meeting point is v.

Notice that all edges of the tree (except possibly the last one if the number of edges is odd) are explored by calling Procedure `Split`. Observe that in any bushy tree, there are only *red* and *green* edges.

It is proved that the BHS scheme produced by Algorithm 3 traverses any red edge 6 times and any green edge 2 times. Moreover every phase is a 2-phase (i.e. the two agents traverse edges in parallel), except possibly the last phase, and no agent waits in any 2-phase. Hence, in view of Lemma 7 the following theorem

holds:

Theorem 4. *([19]) Algorithm 3 produces a fastest BHS-scheme for any bushy tree.*

3.4 The case of arbitrary trees

For arbitrary trees there is a simple algorithm which guarantees an approximation ratio strictly less than 4, i.e., an algorithm which produces a BHS-scheme whose time is less than 4 times the cost of the fastest BHS scheme for any arbitrary tree and starting node. The algorithm is the following: both agents traverse the tree together in depth first search order and explore each new node with a *probe* phase. This BHS scheme explores a n-node tree within $4(n-1)-2l$ steps, where l is the number of leafs in the tree. In any BHS scheme of an n-node tree, each edge has to be traversed at least twice by an agent which gives a total of $2(n-1)$ traversals. Hence at least $n-1$ steps are required which implies a strictly less than 4 approximation factor for the above algorithm.

A better (and still simple) approximation algorithm for arbitrary trees was also proposed in [19] achieving a ratio of $\frac{5}{3}$. The high-level description of the algorithm is the following. Let v be the meeting point of the two agents after a phase (initially $v \equiv s$); the edges with upper node v are explored by calling Procedure `Split` until either all such edges are explored or there is at most one remaining unexplored edge incident to v, which is explored by calling Procedure `Probe`; this is repeated for any child of v.

The question of whether it is possible to produce a fastest BHS-scheme for any arbitrary tree (or it is NP-hard) is open until now. The conjecture is that it is possible but such an algorithm would probably need to take care of many special cases that could occur in an arbitrary tree.

4 Black-Hole Search in Synchronous Graphs

Producing a fastest BHS scheme for arbitrary graphs was proved to be NP-hard in [54, 55]. This result extended in some way a reduction which showed the NP-hardness of finding fastest schemes for a more general version of the Black Hole Search problem in which a set of safe nodes is given (instead of just the starting node) and appeared in [18]. Later, in [56], APX-hardness for both versions in arbitrary graphs was proved. In this section we first describe the (less complicated) reduction that shows the NP-hardness of the general BHS problem and later we discuss the original BHS problem. We also discuss the APX-hardness results and approximation algorithms for both versions.

4.1 The general Black Hole Search problem

In [18] the authors studied the BHS problem in synchronous arbitrary graphs adopting a more general scenario in which initially a subset of nodes of the network (instead of just the starting node), containing the starting node, is safe, and the black hole can be located in one of the remaining nodes. Let us call this version of the problem *general Black Hole Search (gBHS)*. It was shown that the problem of finding the fastest possible black hole search scheme by two agents in an arbitrary graph is NP-hard, and a 9.3-approximation algorithm was given for this problem, i.e., a polynomial time algorithm which, given a graph with a subset of safe nodes and a starting node as input, produces, in polynomial time, a black hole search scheme whose time is at most 9.3 times larger than the time of the fastest scheme for this input.

4.1.1 Model and terminology

The model which was used was the same as before with the following differences. Given are:

- a graph G with node s which is the starting node of both agents, and
- a subset S of *safe* nodes containing s (S cannot contain a black hole).

It was again assumed that there is at most one black hole in the network and the goal is to find a gBHS-scheme for the input (G, S, s).

4.1.2 NP-hardness of the general black hole search problem

To prove NP-hardness of the gBHS problem, the authors presented a reduction from the (NP-complete) Hamiltonian Cycle Problem (HC problem) to the decision version of the gBHS problem (called dgBHS).

The HC problem:

Instance: A Graph G.

Question: Does G contain a Hamiltonian cycle?

The dgBHS problem:

Instance: A graph G' with a subset S of safe nodes, starting node $s \in S$, positive integer X.

Question: Does there exist a gBHS scheme for the input (G', S, s), with time at most X?

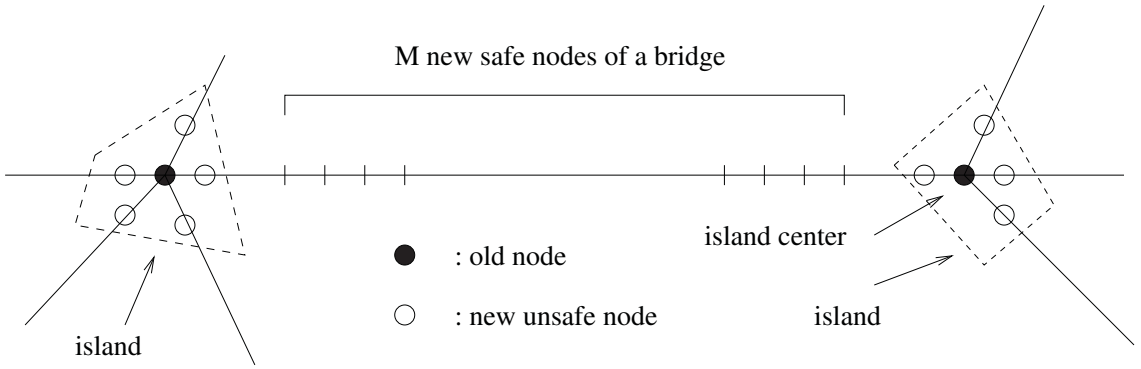

Figure 2: Construction of a dgBHS problem instance

Construction

Let a graph G with n nodes and e edges be an instance of the HC problem. We construct a new graph G' as follows. Call the nodes of graph G old nodes. In each edge of G we add 2 new unsafe nodes adjacent to endpoints of this edge and $M = 4e + 5n - 1$ new safe nodes between them, as in Figure 2. Let s be any node of the old n nodes. All old nodes except s are considered unsafe. Hence the set of unsafe nodes consists of all old nodes except s and all nodes adjacent to old nodes.

The instance of the dgBHS problem is the graph G' with $n' = n + (M + 2)e$ nodes, the set S of $Me + 1$ safe nodes, node s as a starting node, and the integer $X = M(n + 1) - 1$.

The construction of this instance from the graph G can be clearly done in polynomial time. If there is a Hamilton Cycle C in the old graph G then the two agents can follow this cycle starting at node s and identify the black hole (or discover that there is no black hole) in the new graph G' (constructed as above) in at most $M(n + 1) - 1$ time units as follows:

They explore by probing (one agent goes to an adjacent node while the other agent waits) the nodes which are adjacent to the center of an island (see Fig. 2) except the two nodes which are on C and they return to the center of the island. Then they explore by probing the unsafe node of the bridge on C in the chosen direction and they walk along the bridge, till they get to the last safe node of it. Subsequently, they explore the adjacent unsafe node v (in the next island) by probing, walk to it, explore the center of this island by probing and walk to it. They repeat the above procedure in every island on C until they reach again node s.

In the absence of a Hamiltonian Cycle in graph G the agents need at least $M(n + 1)$ time units to identify the black hole (or discover that there is no black hole) in G' (see [18] for more details).

Theorem 5. *([18]) Producing a fastest BHS scheme for the gBHS problem in arbitrary graphs is NP-hard.*

4.1.3 An approximation algorithm for the gBHS problem

An approximation algorithm (Algorithm 4) for the gBHS problem in arbitrary graphs also appeared in [18]. The algorithm is based on the construction of a *Steiner Tree* of the input graph G, where the unsafe nodes of G along with the starting node s are the *required* nodes. Recall that a Steiner Tree for a graph $G = (V, E)$ with the set $R \subseteq V$ of required nodes is any subtree of G containing R. We can construct such a Steiner Tree T in polynomial time with approximation ratio α, where $\alpha = 1 + \frac{\ln 3}{2} < 1.55$ ([53], [65]). More specifically, if x is the number of unsafe nodes in G plus one for node s, and y is the number of safe nodes in T (excluding node s), while y^* is a minimum number of safe nodes (excluding node s) needed for the optimal Steiner Tree, then $(x + y) \leq 1.55(x + y^*)$.

Algorithm 4 (*An approximation algorithm for the gBHS problem in graphs*)

```
 1: construct a minimum Steiner Tree T containing all unsafe nodes and node s;
 2: next := s;
 3: repeat
 4:    v := next;
 5:    for every unexplored node z adjacent to v do
 6:       probe(z);
 7:    end for
 8:    if there are still unexplored nodes then
 9:       case 1: there is an unexplored node adjacent to a child w of v:
10:          next := w;
11:       case 2: every node adjacent to any child of v is explored:
12:          let t be the parent of v;
13:          next := t;
14:       walk to node next;
15:    end if
16: until every node is explored
17: walk to node s and report the location of the black hole;
```

The high-level description of Algorithm 4 is the following. First a Steiner Tree is constructed as described above. Let v be the meeting point of the two agents after a phase (initially $v \equiv s$); the unexplored children of v are explored by calling procedure probe; if there is an unexplored node adjacent to a child of v then the agents go to that child and repeat. Otherwise the two agents go to the parent of v and repeat.

The time-complexity of Algorithm 4 is polynomial on the size of G and is dominated by the time of constructing the Steiner Tree. Algorithm 4 is in fact a Depth First Search type algorithm with the only difference that any unsafe node is visited using a cautious way (probing). The time spent on traversals of any edge (u, v) (v is a child of u) of the tree T is at most 4 units: the worst case is when edge (u, v) leads to an unexplored node v which is not a leaf in T, therefore the agents spend 2 time units for probing v, 1 time unit to walk to v and another time unit to return to node u - after the exploration of the descendants of v. The total time needed by the gBHS-scheme produced by Algorithm 4 is less than $4(x + y)$.

Lemma 8. *([18]) Any gBHS scheme for the graph G requires at least $\frac{4}{3}(x + y^*)$ traversals of edges.*

The lower bound of Lemma 8 together with the upper bound achieved by Algorithm 4 and the approximation factor α (< 1.55) of the Minimum Steiner Tree lead to the following theorem:

Theorem 6. *([18]) Algorithm 4 is an approximation algorithm for the gBHS problem with ratio $6\alpha < 9.3$.*

The approximation ratio of the gBHS problem was improved in [56] where a 6-approximation algorithm was given for the problem, using again the approach of minimum spanning trees.

4.2 The BHS problem in arbitrary graphs

In [54] (see also the full version in [55]) it was proved that the problem of constructing a time optimal Black Hole Search scheme for two agents under the restricted scenario of one safe node, the starting node (i.e., the original BHS problem) is NP-hard even on planar graphs.

4.2.1 NP-Hardness of Black Hole Search in Planar Graphs

The NP-hardness of the BHS problem in planar graphs (BHSp problem) was shown by providing a reduction from a particular version of the Hamiltonian Cycle problem (Hamiltonian Cycle on cubic planar graphs (cpHC problem)) to the decision version of the BHSp problem.
Here is the formalization of the two problems:

The cpHC problem

Instance: a cubic planar 2-connected graph $G = (V, E)$, and an edge $(x, y) \in E$;
Question: does G contain a Hamiltonian cycle that includes edge (x, y) ?

The dBHSp problem

Instance: a planar graph $G' = (V', E')$, with a starting node $s \in V'$, and a positive integer X;

Question: does there exist a BHS scheme for G' starting from s with time at most X?

The cpHC problem above is proved to be NP-complete in [55] by a simple reduction from the Hamilton Cycle problem in cubic planar graphs without the extra requirement that the Hamiltonian cycle passes through a given edge (which was proven NP-complete in [47]). Then with a more technical construction than the one in the previous section the authors reduce the cpHC problem to the dBHSp problem proving that the dBHSp problem is NP-complete:

Theorem 7. *([55]) A graph G with n nodes has a Hamiltonian Cycle passing through an edge (x, y) if and only if there is a BHS scheme for a graph G' and a starting node s (constructed from G in polynomial time) with time at most $5n + 2$ units.*

4.2.2 An approximation algorithm for the BHS problem

For an approximation algorithm for the BHS problem in an arbitrary graph G, a natural approach is the following: First select a spanning tree in G and then explore the graph by traversing the tree edges. Since any BHS scheme of a n-node graph requires at least $n-1$ steps, this approach together with the simple algorithm described at the end of Section 3 guarantees an approximation ratio of 4 for any arbitrary graph. To follow this spanning-tree approach more effectively we need an algorithm for constructing "good" BHS schemes for trees and an algorithm for computing spanning trees which are "good" for those schemes. A linear-time algorithm which extends the construction of the optimal BHS scheme for *bushy* trees (which has been presented in Section 3) to the general rooted trees is proposed in [55]. This algorithm still does not guarantee optimality of computed exploration schemes for trees other than bushy trees: the question of computing in polynomial time optimal exploration schemes for general trees remains open. The cost of the exploration scheme computed by this extended algorithm for an arbitrary tree T is given as a function of the number of nodes of different types in tree T. Then a heuristic algorithm is presented for the problem of computing a rooted spanning tree T of graph G which gives a relatively small value of that formula. This approach guarantees an approximation ratio of at most $3\frac{3}{8}$.

4.3 APX-hardness of the BHS problem in arbitrary graphs

Finally in [56] both versions of the black hole problem discussed in this section were proved to be APX-hard. It was shown that a fastest BHS scheme for the general Black Hole Search problem is not approximable in polynomial time within a $1 + \varepsilon$ factor for any $\varepsilon < \frac{1}{388}$, unless $P = NP$. It was also proved that the original BHS problem (in which only the starting node is initially known to be safe) is also APX-hard.

The authors provide an explicit lower bound on the approximability of the General Black Hole Search problem by showing an approximation-preserving reduction from a particular subcase of the Traveling Salesman Problem (TSP), presented in [34]. In this subcase of TSP, distances between nodes are symmetric and satisfy the triangle inequality while the maximum distance M is a constant.

Lemma 9. *[34] It is NP-hard to approximate* TSP(1,M) *within* $1 + \varepsilon$ *for any* $\varepsilon < \frac{1}{388}$.

The authors' approach to prove the APX-hardness of the gBHS problem is the following. They first provide a reduction from instances (G, d) of TSP to instances (G', S, s) of the gBHS problem and then they show that if the optimal solution of the gBHS, constructed by the given reduction from an instance of TSP, can be approximated within a $(1 + \varepsilon)$ factor, then the optimal solution of the corresponding instance of TSP can be approximated within the same factor.

Theorem 8. *([56]) The gBHS problem is not approximable in polynomial time within a factor of* $1 + \varepsilon$ *for any* $\varepsilon < \frac{1}{388}$, *unless* $P = NP$.

They also provide a reduction from the TSP problem where the distances between the nodes are either 1 or 2 to the original version of the BHS problem (in which only the starting node is known to be safe) and show that:

Theorem 9. *([56]) It is NP-hard to approximate the BHS problem within a factor* $1 + \varepsilon$ *for any* $\varepsilon < \frac{1}{2258}$.

5 The BHS problem with Scattered Finite Automata

In all previous sections of this survey the Black Hole Search problem was studied for co-located agents with memory.

The memory is a critical capability that allows the agents to store (or make) a map of the network, to keep information about (or count) the number of nodes of the graph and generally helps them decisively to explore the network.

In asynchronous networks the co-location of the agents in the whiteboard model gave them the ability to assign different identities to themselves and different labels at nodes while in synchronous networks the co-location gave the agents the ability to *probe* the network in a *cautious way* and discover the black hole.

The question of whether even more weak models allow the solution of the black hole search problem has been recently raised. Although there are results for scattered agents (not initially starting at the same node) using tokens (instead of whiteboards) and even not having a map of the graph for asynchronous networks, the memory capability cannot be dropped since in such networks the agents need to know (and store) the number of nodes (or at least an upper bound). However in synchronous networks no knowledge of such bound is generally needed so an interesting question is whether *deterministic finite automata (DFAs)* (i.e., with only a constant memory), initially scattered in the network, without a map and having only a constant number of 'pure' tokens (which can leave only at nodes) can still solve the black hole search problem.

We notice here that Rollik ([66]) has proved that no finite set of finite automata can cooperatively perform exploration of all cubic planar graphs. Since a finite automaton is more powerful than a token, it means that no finite set of finite automata using any constant number of tokens can explore all cubic planar graphs even when the agents start at the same node and can exchange information when they meet at nodes. It is clear that if the agents cannot even explore the graph (i.e., visiting all nodes) they cannot solve the BHS problem. Hence a challenging question is whether there are synchronous network topologies for which the agents can solve this problem in such a weak model. Even when the agents can communicate when they meet at a node, since they are initially scattered they need to solve the *rendezvous* problem which is far from trivial for identical, anonymous DFAs carrying only indistinguishable tokens.

Very recently the BHS problem in this model (initially *scattered anonymous agents* with *constant memory*, carrying only a constant number of *pure* indistinguishable tokens and having the ability to communicate only when they meet at the same node) has been investigated for ring ([11]) and torus ([10]) topologies.

Since the model now is quite different than before we give below some more details.

5.1 The model

The model consists of an anonymous, synchronous network with $k \geq 2$ identical mobile agents that are initially located at distinct nodes called *homebases*. Each mobile agent owns a constant number t of identical tokens which can be placed at any node visited by the agent. The tokens are indistinguishable. Any token or

agent at a given node is visible to all agents on the same node, but not visible to agents on other nodes. The agents follow the same deterministic algorithm and begin execution at the same time and being in the same initial state.

A token is called *movable* if it can be put on a node and picked up later by any mobile agent visiting the node. Otherwise the token is called *unmovable* in the sense that, once released, it can occupy only the node where it was released.

Formally a mobile agent was considered as a finite Moore automaton.

All computations by the agents are independent (the agents have no knowledge) of the size of the network. The algorithms for the ring topology work even without knowledge of the number of the agents. There is exactly one black hole in the network. An agent can start from any node other than the black hole and no two agents are initially co-located. Once an agent detects a link to the black hole, it marks the link permanently as dangerous. The goal is that at the end of a black hole search scheme, all links incident to the black hole (and only those links) are marked dangerous and that there is at least one surviving agent. Note that this definition of a successful BHS scheme is slightly different from the original definition. Indeed, in the original definition, it is required that there is at least one surviving agent, and this agent knows the location of all edges incident to the black hole. However, this is impossible in this model since the agents have only constant memory.

5.2 The ring topology

For the ring topology four different scenarios were considered in [11] depending on whether the tokens are movable or not, and whether the agents agree on a common orientation. Surprisingly, the agreement on the ring orientation does not influence the number of agents needed in the case of movable tokens but is important in the case of unmovable tokens.

The lower bounds presented in [11] are very strong in the sense that they do not allow any trade-off between the number of agents and the number of tokens for solving the BHS problem. In particular it was shown that:

- Any constant number of agents, even having unlimited memory, cannot solve the BHS problem with less tokens than depicted in all cases of Table 1.

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

Meanwhile the algorithms match the lower bounds on the number of tokens, are asymptotically time-optimal and since they do not require any knowledge of the

size of the ring or the number of agents, they work in any anonymous synchronous ring, for any number of anonymous identical agents (respecting the minimal requirements of Table 1).

		Resources necessary and sufficient	
Tokens are	**Ring is**	**# agents**	**# tokens**
Movable	Oriented	3	1
	Unoriented		
Unmovable	Oriented	4	2
	Unoriented	5	2

Table 1: Results for BHS with DFAs and tokens in synchronous rings

5.2.1 Impossibility results

Oriented rings

Consider that the agents agree on the orientation of the ring (i.e., the ports of the edges are consistently labeled as 'left', 'right'). When the tokens are unmovable, a team of any constant number of agents needs at least two tokens per agent to solve the BHS problem. This is due to the fact that with only one unmovable token the agents are forced to always take the same actions including marking (incorrectly in different sized rings) links as dangerous.

Theorem 10. *([11]) For any constant k, there exists no algorithm that solves BHS in all oriented rings containing one black hole and k or more scattered agents, when each agent is provided with only one unmovable token. The result holds even if the agents have unlimited memory.*

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

Theorem 11. *([11]) Two mobile agents carrying any number of movable (or unmovable) tokens each, cannot solve the BHS problem in an oriented ring, even if the agents have unlimited memory.*

When the tokens are unmovable, even three agents are not sufficient to solve BHS as shown below. This is due to the following facts: a) because of the constant number of unmovable tokens the agents cannot guarantee to leave a token at a node which is adjacent to the black hole, and b) after one of them vanishes in the black hole, the remaining two agents could possibly meet (by exploiting the

asymmetry left by the vanished agent) but they cannot discover both incident links to the black hole.

Theorem 12. *([11]) Three mobile agents carrying a constant number of unmovable tokens each, cannot solve the BHS problem in an oriented ring, even if agents have unlimited memory.*

Unoriented rings

In an unoriented ring (i.e., the clockwise direction perceived by an agent is handled by an adversary), even four agents do not suffice to solve the BHS problem with unmovable tokens, since two of them can be forced by the adversary to vanish in the black hole at the same time leaving their tokens more than a constant distance away from the black hole and the remaining two agents cannot correctly mark both links incident to the black hole.

Theorem 13. *([11]) In an unoriented ring, four agents carrying any constant number of unmovable tokens each, cannot correctly mark any link incident to the black hole, even when the agents have unlimited memory.*

5.2.2 A BHS scheme with movable tokens

If each agent has a movable token it can perform a cautious walk type movement using its token. The `Cautious-Walk` procedure consists of the following actions: Put the token at the current node, move one step in the specified direction, return to pick up the token, and again move one step in the specified direction (carrying the token).

We show that only three agents are sufficient to solve BHS, when they have one movable token each. Algorithm 5 achieves this, both for oriented and unoriented rings.

Algorithm 5 (*Three DFAs with one movable token in synchronous rings*)

```
1: repeat
2:     CautiousWalk(Left)
3: until you see a token and no agent OR next link is marked Dangerous
4: MarkLink(Left)
5: repeat
6:     CautiousWalk(Right)
7: until you see a token and no agent OR next link is marked Dangerous
8: MarkLink(Right)
```

Theorem 14. *([11]) Algorithm 5 solves the BHS problem in an unoriented ring with $k \geq 3$ agents having constant memory and one movable token each.*

5.2.3 BHS schemes with unmovable tokens

For agents having only unmovable tokens, we use the *probing* technique (as discussed in previous sections) for exploring new nodes. Now in order to use this technique, we need to gather two agents at the same node and break the symmetry between them, so that distinct roles can be assigned to each of them. This is the main difficulty that has been taken care by the algorithms. The basic idea of our algorithms is the following. We first identify the two homebases that are closest to the black hole (one on each side). These homebases are called *gates*. The gates divide the ring into two segments: one segment contains the black hole (thus, is dangerous); the other segment contains all other homebases (and is safe). Initially all agents are in the safe part and an agent can move to the dangerous part only when it passes through the gate node. We ensure that any agent reaching a gate node, waits for a partner agent in order to perform the probing procedure. We now present two BHS algorithms, one for oriented rings and the other for unoriented rings.

Oriented rings

In this section, we describe an algorithm (`Algorithm BHS-Ring-2`) using at least four agents with two unmovable tokens.

Description of `Algorithm BHS-Ring-2`:

During the first phase of the algorithm each agent places a token on its homebase, moves left until the next homebase (i.e., next node with a token) and then returns to its homebase to put down the second token. During this phase one agent will fall into the black hole and there will be a unique homebase with a single token (we call this node the *gate* node) and all the other homebases will eventually contain exactly two tokens each. However, the agents may not complete this phase of the algorithm at the same time. Thus during the algorithm, there may be multiple homebases that contain a single token. Whenever an agent reaches any node containing a single token, it waits for a partner agent and then they perform probing in the left direction. One of the agents of a pair eventually falls into the black hole and the other agent marks the edge leading to the black hole and returns to the gate node, waiting for another partner. When another agent arrives at this node, these two agents perform probing in the opposite direction to find the other incident link to the black hole. ■

Theorem 15. *([11]) Algorithm BHS-Ring-2 correctly solves the black hole search problem in any oriented ring with* 4 *or more agents having constant memory and carrying two unmovable tokens each.*

Unoriented rings

For unoriented rings, we need at least 5 agents with two unmovable tokens each. The `Algorithm BHS-Ring-3` for unoriented rings using five agents with two unmovable tokens is similar to the one for oriented rings, except that each agent chooses an orientation. When two agents meet they can agree on the orientation and assign different roles to themselves.

Description of `Algorithm BHS-Ring-3`:

Each agent puts one token on its homebase, goes on *its left* until it sees another token and then returns to its homebase. Now the agent goes on its right until it sees a token and then returns again to the homebase. The agent now puts its second token on its homebase. During this operation exactly two agents will fall into the black hole. Each surviving agent walks to its left until it sees a node u with a single token. At this point the agent has to wait, since either there is a black hole ahead, or u is the homebase of an agent b that has not yet returned to put its second token. Since we assume there are at least five agents, at least two of them will meet at a gate. They identify one of the links incident to the black hole and then the remaining agent can join the other agent who waits and identify together the other link incident to the black hole. ■

Theorem 16. *([11]) Algorithm BHS-Ring-3 correctly solves the black hole search problem in an unoriented ring with 5 or more agents having constant memory and carrying two unmovable tokens each.*

5.3 The torus topology

While in ring topologies the exploration of a safe (even unoriented) ring is feasible by one DFA with one unmovable token, in torus topologies the exploration of a (safe) torus is not always possible by a DFA using tokens.

5.3.1 Impossibility results in an oriented torus

Agents with unmovable tokens

Theorem 17. *([10]) For any constant numbers k, t, there exists no algorithm that solves BHS in all oriented tori containing one black hole and k scattered agents, where each agent has a constant memory and t unmovable tokens.*

The idea of the proof of Theorem 17 is the following: It is shown that an adversary (by looking at the transition function of an agent) can always select a big enough torus and initially place the agents so that no agent visits nodes which contain tokens left by another agent, or meets with another agent. Moreover there are nodes on the torus never visited by any agent. Hence the adversary may place the black hole at a node not visited by any of the agents to make any algorithm fail.

Agents with movable tokens

The situation with movable tokens is quite different than with unmovable ones since now any oriented (safe) torus can be eventually explored even by one DFA carrying one movable token (exploration without stop, also known as *perpetual* exploration). Hence any lower bound using movable tokens can not be based on impossibility of exploration (visiting all nodes of the torus).

A fairly easy observation is that two agents carrying any number of movable tokens cannot solve the BHS problem in an oriented torus even if the agents have unlimited memory due to the fact that the adversary can always place the agents and the black hole in such a way that one of the agents vanishes while changing vertical rings and the other one while changing horizontal rings. Additionally the following lower bound is shown:

Lemma 10. *([10]) There exists no algorithm that could solve the BHS problem in all oriented tori using three agents with constant memory and one movable token each.*

The idea of the proof is that at least two of the three agents are forced to leave their tokens more than a constant number of nodes away (otherwise they are not able to explore the torus) from the black hole before they vanish and then the third agent cannot decide for the correct location of the black hole.

Hence the following theorem holds:

Theorem 18. *([10]) At least three agents are necessary to solve the BHS problem in an oriented torus of arbitrary size. Any algorithm solving this problem using three agents requires at least two movable tokens per agent.*

5.3.2 BHS schemes in oriented tori using movable tokens

Due to the impossibility result from the previous section, any algorithm for the BHS problem by DFAs should use movable tokens. We describe below an algorithm which leads three agents carrying three tokens each to locate the black hole.

Algorithm `BHS-torus-33`

An agent explores one horizontal ring at a time and then moves one step South to the next horizontal ring and so on. When exploring a horizontal ring, the agent leaves one token on the starting node. This node is called the *homebase* of the agent and the token left (called homebase token) will be used by the agent to decide when to proceed to the next horizontal ring. The agent uses the two remaining tokens to repeat Cautious-Walk in the East direction until it has seen twice a node containing one token. Any node containing one token is a homebase either of this agent or of another agent. The agent moves to the next horizontal ring below (using again Cautious-Walk with three tokens) after encountering two homebases. It then repeats the same exploration process for this new ring leaving one token at its new homebase. Whenever the agent sees two or three tokens at the end of a cautious-walk, the agent has detected the location of the black hole: If there are two (resp. three) tokens at the current node, the black hole is the neighboring node w to the East (resp. South). In this case, the agent stops its normal execution and then traverses a cycle around node w, visiting all neighbors of w and marking all the links leading to w as dangerous. ■

Theorem 19. *([10]) Algorithm BHS-torus-33 correctly solves the BHS problem with* 3 *or more agents and three tokens.*

Using a similar technique, the authors present an algorithm that solves the problem using four agents and two tokens. Finally they give a more involved algorithm (using a technique which makes the agents meet when they are 'close' enough) that meets the lower bound, i.e., solves the BHS problem using three agents and two tokens.

Theorem 20. *([10]) The BHS problem can be solved in any oriented torus with exactly three agents carrying two tokens each.*

5.3.3 The case of an unoriented torus

The situation in an unoriented torus is quite different than in an oriented one. As new results in [60] show, any constant number of DFAs with one movable token each cannot explore all (safe) unoriented tori and thus the BHS problem cannot be solved. However, surprisingly enough[1], it is shown in [60] that one agent with two movable tokens can explore any totally unoriented (safe) torus which gives a hope that even in a totally unoriented torus the BHS problem can be solved by a

[1] Given the old result of Rollik in [66] (discussed in the beginning of the section) which proved that no finite set of finite automata can cooperatively perform exploration of all cubic planar graphs.

small number of agents with movable tokens. In a partially unoriented torus the BHS problem can be solved using five agents with constant memory and three movable tokens ([60]).

6 Discussion and Future Directions

In a distributed mobile computing environment the most pressing security concerns one has to deal with are a hostile agent (i.e., a malicious mobile process), and a hostile host (i.e., a harmful process at a network site). In this survey we tried to discuss the second one as it appears in the literature under the model of a highly harmful host called *black hole*. With the exclusion of Section 2 we presented most of the results for synchronous networks. As it appeared in this survey, the black hole search problem is related to the *graph exploration* problem and sometimes (when scattered agents are considered) to the *rendezvous* problem.

6.1 The impact of the *Cautious-Walk* technique

In most of the part of this survey the algorithms for the black hole search problem relied more or less in the *Cautious-Walk* technique. However when the network is *directed* this technique cannot be applied any more. The problem has been studied for co-located agents with memory but without a map of the network in asynchronous and synchronous *directed* graphs with whiteboards in [16, 57]. In [16] it was proved that in directed asynchronous graphs with whiteboards there is an exponential gap on the number of agents needed in order to solve the BHS problem: 2^{Δ} agents are needed in the worst case (where $\Delta + 1$ agents without a map are sufficient in undirected graphs), where Δ is the in-degree of the black hole. This is a consequence of the fact that in directed graphs the Cautious-Walk technique cannot be applied. This lower bound holds even in the case of synchronous graphs. However in planar directed graphs with a planar embedding known to the agents, 2Δ agents are needed and $2\Delta + 1$ agents are sufficient. In synchronous directed graphs with whiteboards it was shown in [57] that $O(\Delta \cdot 2^{\Delta})$ agents are sufficient to solve the problem.

6.2 Other models of hostile-node identification problems

Weaker as well as stronger faults than black holes have also been introduced in the literature. In [15] they study how to fast locate and repair special faults which can destroy only the first visiting agent (thus weaker than black holes) using co-located agents with memory in synchronous known networks.

As already mentioned, black hole is a particular type of a malicious host with a very simple behavior: killing every agent instantly without leaving any trace. In reality, a host has many ways to harm the agents: it may not only kill any agent residing in it at any time, it may also duplicate agents, introduce fake agents, tamper the runtime environment (e.g. changing the contents of the whiteboard), or disobey communication protocols (e.g., do not execute agents in FIFO order). Those are all very interesting and challenging future directions on protecting a network from hostile nodes. In [58] appears the first attempt to study how the abilities of a malicious host affect the solvability of exploration problems. There, a different dangerous behavior is studied for co-located agents: the authors consider an asynchronous ring with whiteboards and assume black-holes with Byzantine behavior, which do not always destroy a visiting agent and can change the whiteboard stored there. Their main result shows that, in the case of rings, it is sufficient to protect the internal state of the agent (i.e., the malicious host cannot change or create the content of agent's memory), and the periodic data retrieval problem is solvable by a constant number of agents.

Aknowledgements

I wish to gratefully thank all my co-authors on the Black-Hole Search problem: Andrzej Pelc, Jurek Czyzowicz, Dariusz Kowalski, Ralf Klasing, Tomasz Radzik, Fabiano Sarracco, Jérémie Chalopin, Shantanu Das, and Arnaud Labourel. I would also like to thank Evangelos Kranakis, Danny Krizanc, Stefan Dobrev, Rastislav Královič, Nicola Santoro and Paola Flocchini for the many discussions we had during the years on this problem and its variants. Finally, I wish to thank Panagiota Fatourou for encouragement.

References

[1] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. on Computing*, 36(1):56–82, 2006.

[2] M. Asaka, S. Okazawa, A. Taguchi, and S. Goto. A method of tracing intruders by use of mobile agents. In *Proc. of 9th Annual Conf. of the Internet Society*, 1999. (http://www.isoc.org/).

[3] B. Balamohan, S. Dobrev, P. Flocchini, and N. Santoro. Asynchronous exploration of an unknown anonymous dangerous graph with o(1) pebbles. In *Proc. of 19th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 7355, pages 279–290, 2012.

[4] B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Time optimal algorithms for black hole search in rings. In *4th Annual Int. Conference on Combinatorial Optimization and Applications*, LNCS 6509, pages 58–71, 2010.

[5] B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Improving the optimal bounds for black hole search in rings. In *Proc. of 18th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 6796, pages 198–209, 2011.

[6] B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Time optimal algorithms for black hole search in rings. *Discrete Mathematics, Algorithms and Applications*, 3(4):1–15, 2011.

[7] L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In *Proc. of 14th ACM Symp. on Parallel Algorithms and Architectures*, pages 200–209, 2002.

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

[9] N. Borselius. Mobile agent security. *Electronics and Communication Engineering Journal*, 14(5):211–218, 2002.

[10] J. Chalopin, S. Das, A. Labourel, and E. Markou. Black hole search with finite automata scattered in a synchronous torus. In *Proc. of 25th Int. Symp. on Distributed Computing*, LNCS 6950, pages 432–446, 2011.

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

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

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

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

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

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

[17] J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in tree networks. In *Proc. of 8th International Conference on Principles of Distributed Systems*, LNCS 3544, pages 67–80, 2004.

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

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

[20] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1-3):34–48, 2007.

[21] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment i: the rectilinear case. *J. of the ACM*, 45:215–245, 1998.

[22] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. *J. of Algorithms*, 51:38–63, 2004.

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

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

[25] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile agents search for a black-hole in an anonymous ring. In *Proc. of 15th Int. Symp. on Distributed Computing*, pages 166–179, 2001.

[26] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *Proc. of 6th Int. Conf. on Principles of Distributed Systems*, pages 34–46, 2003.

[27] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distributed Computing*, 19(1):1–19, 2006.

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

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

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

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

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

[33] C. A. Duncan, S. G. Kobourov, and V. S. Kumar. Optimal constrained graph exploration. In *Proc. of 12th Annual ACM Symp. on Discrete Algorithms*, pages 807–814, 2001.

[34] L. Engebretsen and M. Karpinski. TSP with bounded metrics. *Journal of Computer and System Sciences*, 72(4):509–546, 2006.

[35] P. Flocchini, M. J. Huang, and F. L. Luccio. Contiguous search in the hypercube for capturing an intruder. In *Proc. of 18th IEEE Int. Parallel and Distributed Processing Symp.*, 2005.

[36] P. Flocchini, M. J. Huang, and F. L. Luccio. Decontamination of chordal rings and tori. In *Proc. of 8th Workshop on Advances in Parallel and Distributed Computational Models*, 2006.

[37] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *Proc. of 22nd Int. Symp. on Distributed Computing*, LNCS 5218, pages 227–241, 2008.

[38] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, 62(3-4):1006–1033, 2012.

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

[40] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, 2012.

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

[42] P. Flocchini, B. Mans, and N. Santoro. Sense of direction in distributed computing. *Theoretical Computer Science*, 291:29–53, 2003.

[43] P. Flocchini and N. Santoro. *Mobile Agents in Networking and Distributed Computing (Eds. J. Sao and S. Das)*, chapter Distributed Security Algorithms for Mobile Agents. Wiley, 2012.

[44] N. Foukia, J. G. Hulaas, and J. Harms. Intrusion detection with mobile agents. In *Proc. of 11th Annual Conf. of the Internet Society*, 2001.

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

[46] P. Fraigniaud and D. Ilcinkas. Digraph exploration with little memory. In *Proc. of 21st Symp. on Theoretical Aspects of Computer Science*, pages 246–257, 2004.

[47] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

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

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

[50] 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.

[51] F. Hohl. A framework to protect mobile agents by using reference states. In *Proc. of 20th Int. Conf. on Distributed Computing Systems*, pages 410–417, 2000.

[52] W. Jansen. Countermeasures for mobile agent security. *Computer Communications*, 23(17):1667–1676, 2000.

[53] V. Kann. Minimum steiner tree. http://www.nada.kth.se/~viggo/wwwcompendium/node78.html.

[54] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and approximation results for black hole search in arbitrary graphs. In *Proc. of 12th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 3499, pages 200–215, 2005.

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

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

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

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

[59] F. Luccio, L. Pagli, and N. Santoro. Network decontamination with local immunization. In *Proc. of 8th Workshop on Advances in Parallel and Distributed Computational Models*, 2006.

[60] E. Markou and M. Paquette. Black hole search and exploration in unoriented tori with synchronous scattered finite automata. Manuscript.

[61] S. Ng and K. Cheung. Protecting mobile agents against malicious hosts by intention of spreading. In *Proc. of Int. Conf. on Parallel and Distributed Processing and Applications*, pages 725–729, 1999.

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

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

[64] G. Prencipe. Corda: Distributed coordination of a set of autonomous mobile robots. In *Proc. of 4th European Research Seminar on Advances in Distributed Systems*, pages 185–190, 2001.

[65] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *Proc. of 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[66] H. Rollik. Automaten in planaren graphen. *Acta Informatica*, 13:287–298, 1980.

[67] 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.

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

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

[70] W. Shi. Black hole search with tokens in interconnected networks. In *Proc. of 11th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems*, pages 670–682, 2009.

[71] E. H. Spafford and D. Zamboni. Intrusion detection using autonomous agents. *Computer Networks*, 34(4):547–570, 2000.

[72] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. on Computing*, 28(4):1347–1363, 1999.

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