Katedra informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Locating a Black Hole without the Knowledge of Incoming Link

(ŠVK 2009)

Peter Glaus

Advisor:
doc. RNDr. Rastislav Královič, PhD.

Bratislava, 2009

Abstract

We study a group of mobile agents operating on an arbitrary unknown distributed system. One of the nodes of the distributed system is extremely harmful and destroys any incoming agent without notification. The task of exploring the distributed system and locating the harmful node, *Black hole search*, has been studied with various modifications.

We are studying the effects of the *knowledge of incoming link* on the size of the optimal solution. When an agent enters a node, the information which port leads back can be given to it. We refer to this as to the *knowledge of incoming link*. In previous research, it was always assumed that the agent is given this information.

In this paper we study arbitrary, unknown distributed systems without the knowledge of incoming link. Agents are asynchronous and they communicate via whiteboards. We present a lower bound on the size of the optimal solution, proving that at least $\frac{\Delta^2+\Delta}{2}+1$ agents are necessary to locate the black hole. We provide an algorithm for black hole search without the knowledge of incoming link as well. We prove that this algorithm is correct, and that it uses $\frac{\Delta^2+\Delta}{2}+1$ agents, thus providing optimal solution.

KEYWORDS: distributed system, mobile agents, faulty node, black hole search, knowledge of incoming link

Contents

Chapter 1

Introduction

1.1 Black Hole Search

Special part of theoretical computer science concerning multi-processor or network environments is the field of Distributed algorithms. Distributed algorithms can be studied trough very intriguing agent-based model. In this model, distributed system can be regarded as a kind of maze consisting of nodes connected with passages and in this maze is a group of autonomous mobile agents trying to fulfill a given task. In real world, the nodes would be computers in a computer network and the agents would be represented as processes executed on these computers, the task might be finding the best routing path.

Agent-based distributed algorithms have been widely studied. Problems such as graph exploration, finding shortest paths, finding a spanning tree, gathering of agents, leader election and other were already tackled. There are many modifications of these tasks based on agents' knowledge and properties of the distributed system. In most cases though, distributed system is considered non-faulty. We focus our study on distributed systems containing a *black hole*.

Black hole is a harmful node, which destroys all agents that enter it. It is indistinguishable from the outside and once agent enters, it is unable to send a message or warn other agents. *Black Hole Search* is a task in which the agents have to create a map of the distributed system with the location of the black hole pointed out.

1.2 Aim of this Work

The *Black Hole Search Problem* (BHS) is the problem of finding an algorithm for the agents to fulfill the Black Hole Search. The solution to this problem and its efficiency can vary a lot and are dependent on the distributed system and the agent model properties. In unoriented arbitrary unknown graphs

the optimal algorithm needs $\Delta + 1$ agents in the worst case[6], whereas in oriented graph scenario, the number of necessary agents can be up to 2^{Δ}[3]. In this work we want to study this gap by proposing a new model which has some restrictions similar to the oriented graph scenario.

In all previous studies of BHS, it was anticipated that when an agent enters a node, it automatically knows from which link it entered the node. We call this information the *knowledge of the incoming link*. This information is not inherently present in all distributed systems. For instance, one can think of a software agent that starts its execution on a host without any information about the incoming port.

We study the effects of the knowledge of incoming link on unoriented arbitrary unknown distributed systems. We show that without this knowledge, the problem solution needs more agents to work correctly. We show that there are $\Theta(n^2)$ agents needed for algorithm that solves BHS on arbitrary unknown graphs, more precisely $\frac{n^2+n}{2} + 1$ agents are necessary and sufficient.

1.3 Related Work

First studies of BHS where focused on asynchronous distributed systems with ring networks with use of whiteboard communication [5]. They proved that in such conditions two agents are sufficient and the black hole can be located in $O(n \log n)$ steps.

Important results about influence of prior knowledge of the network can be found in [6]. In case of no prior knowledge, the size of the optimal solution is $\Delta+1$ agents and cost is $O(n^2)$ steps. When using a sense of direction, only 2 agents are necessary and they can find the black hole in $O(n^2)$ steps. With the knowledge of map of the distributed system, 2 agents are necessary, the cost of the solution is reduced to $O(n \log n)$ steps.

In [4] authors show that for graphs such as hypercubes, cube-connected-cycles, tori and other a solution with cost of $O(n)$ steps exists. It was proved later that with the consideration of the graph diameter, the cost of the solution can be improved for arbitrary graphs with known topology to $O(n + d \log d)$[8, 9].

The Black hole search problem with agents not starting in the home base, but rather scattered in the graph was considered as well. Results can be found in [7, 1, 10].

Agents working in synchronous mode are more powerful than asynchronous one. In this work, only asynchronous agents are taken into account. The Black hole search problem on synchronous distributed systems is no longer focused on finding a universal optimal solution, rather on finding the optimal traversal for a particular graph. Such variations of BHS problem can be found in [11, 2, 12].

All previously mentioned results use a simple technique called *Cautious walk* which reduces the number of agents that enter the black hole. When entering an unknown port, an agent marks it as dangerous and if it does not die in the black hole, it immediately returns back and marks the port as safe. No other agent enters a dangerous port, because it might possibly lead into the black hole. When considering oriented graphs, this technique is not available, and up to 2^{Δ} agents may be needed [3].

Chapter 2

Definitions

2.1 Computational Model

We start by defining properties of the computational model that are similar in most of the BHS studies and will be used in this work as well. The model consists of a distributed system, which can be simply described as a graph, a group of agents acting based on an algorithm and one or more black holes, which are harmful nodes. Roughly said, our goal is to design such algorithm for the agents, that the agents together fulfill a predefined task, exact definition is stated below.

Distributed System

Distributed system is a graph $G = (V, E)$ with its port labeling. We mostly denote vertices as *nodes* and edges as *links*. The size of the distributed system is the number of the nodes of the graph $n = |V|$.

We also use the term *port*, which denotes the places where a link is connected to a node. Link (u, v) has two ports, one is in the node u and the other is in the node v.

The graphs we use are simple, without double edges or loops of type (u, u). As we consider only about undirected graphs the link (u, v) is the same as (v, u). We expect the graph to be biconnected, otherwise finding the black hole might be impossible[6].

The use of the port labeling is necessary for the purpose of clarity.

Agents

A group of agents is initially placed in a random node of the distributed system. The node where the agents are at the beginning is called *home base*. The agents are autonomous and proceed according to the same algorithm. We focus on the asynchronous agents only. In general, the agents are unaware of each other, although they can communicate by means of messages

or by changing state of a node either using whiteboards or tokens.

An agent's behavior can be divided into basic steps. In one step agent checks the state of a current node, chooses some action, modifies the state of the node or communicates and, at last, it performs the action. The action can be either entering a specific port of the node, staying in the current node and waiting for some event to happen or finishing.

Agents have to fulfill a specific task, in our case it is to find a black hole. By finding a black hole we mean that agents that decide to take the action *finish* should have full map of the graph with the position of the black hole marked down, we will call this task *black hole search*.

Black Hole

There is a harmful node in the distributed system. Every agent that enters a port of a link leading to such node can be considered dead or not functional any more. Once agent enters it, there are no means of communication with other agents. The distributed system itself carries no information whether link leads into such harmful node or not. We denote such node as a *black hole*. Number of links leading into the black hole — the degree of the black hole is denoted Δ. Throughout this work we focus on distributed systems with one black hole only, even though modes with multiple black holes can be considered as well.

For agents that did not enter the black hole we say that they are alive. On the other hand, we will use the term dead or we say that agent died when it entered the black hole.

2.2 The Problem

Agents' task is to explore the graph and locate the black hole. All agents act according to the predefined algorithm. We say that the algorithm ends when all the remaining agents take action *finish*. We use the following definition when considering the correctness of the algorithm:

Definition 1 (Correctness)**.** *The algorithm is correct if for any distributed system the algorithm ends with at least one agent alive and all surviving agents have a correct map of the distributed system with correctly marked location of the black hole.*

The problem of finding such algorithm is denoted as *Black hole search problem*.

Solution Evaluation

To allow us comparison of the solutions for BHS we have to be able to measure the efficiency of algorithms. The first measure of the efficiency

is the number of agents needed by the algorithm to work correctly and is denoted as the *size* of the solution.

The second measure, the *cost* of the solution, is the number of steps that agents make in total before finishing the task. The size is much more important than the cost and thus the focus is on using the optimal number of agents first and then improving the number of steps made by the agents.

When considering the size of an algorithm on a specific distributed system, we can simply look at the number of agents that enter the black hole in the worst case for that particular distributed system. We can expect the number of agents needed by the algorithm as the number of agents that die in the worst case plus one, otherwise it could be improved.

We express both the size and the cost in terms of n — size of distributed system and its other properties such as Δ or diameter d. Unless the port labeling of the graph has some exact property[1], it is used only for clarity, thus we will be considering how does algorithm behave on a specific graph instead of the distributed system.

Adversary

When evaluating solutions, we always consider the worst case scenario. Instead of referring to the worst case we use the concept of adversary. Adversary is a kind of powerful opponent, his goal is to either make the algorithm fail the task or to work least efficiently as possible.

Firstly, when the agents are not able to distinguish between two ports, then the adversary can decide where these ports lead to, possibly make them lead to the black hole. Secondly, adversary can slow down agents traveling trough some link so agent waiting for them is not able to decide whether they fell into the black hole or not.

For more detail we advice the reader to [6, 5].

[1] e.g. sense of direction

Chapter 3

The Knowledge of Incoming Link

All previously stated results and studies assumed that when an agent enters a new node, the agent automatically knows what port did it enter through. We refer to this as to *the knowledge of incoming link*.

Main focus of this work is on the effects of the knowledge of incoming link, more precisely on how lack of this knowledge affects Bhs problem solution.

3.1 Proposed Model

We study Bhs problem in the following scenario:

Definition 2 (Black Hole Search Problem without the Knowledge of Incoming Link)**.** *Asynchronous autonomous agents operate on a distributed system consisting of arbitrary graph and its port labeling. One node of the graph is very harmful — a black hole. Agents work according to a predefined algorithm. At the beginning, they are all stationed in one node — the* home base*. Their task is to perform the Black hole search.*

Agents' only prior knowledge of the distributed system is the number of nodes n and the maximal degree of the black hole Δ. They communicate by means of whiteboards, which are located in every node. Moreover, the agents have no knowledge of incoming link: when they enter a new node they do not have the information what port leads into the previous node.

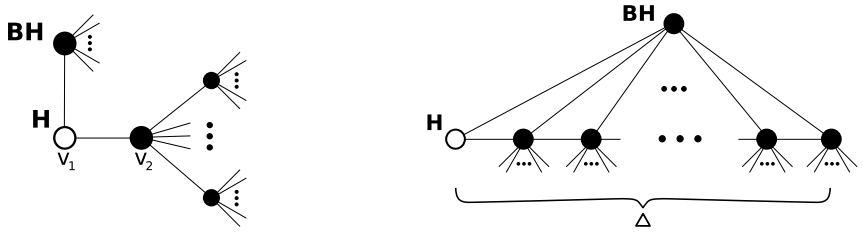
All agents and nodes have their identifiers. Asynchronous agents with the use of whiteboards can easily assign identifiers to themselves and to the nodes as well, thus this variation can be done without loss of generality.

It can be observed that our modification makes agents less powerful and that the size of the optimal solution will be at least $\Delta + 1$ and the cost will

be $O(n^2)$. We start by examining basic properties of this model and the effect of our modification on the size of the solution.

Number of Agents Needed

First observation of the newly proposed model is that the Cautious walk is no longer possible in its original form. If the agent is not able to decide which port it should use to get back, then it is not necessarily able to get back and withdraw the warning. The cautious walk is a mechanism that ensures that maximally one agent travels trough an edge leading into the black hole and thus only Δ agents die.



(a) (b)

Figure 3.1: Two example graphs.

In the Fig. 3.1(a) the node v_1 is the home base. The question is, how many agents have to leave the node v_2 so at least one gets back into the home base. We know that the node v_2 can have n outgoing ports and thus if we send one agent trough each port then at least one will get back into the home base. This leads us to a rough estimation that if we send n agents into the node v_2 then at least one will return.

The agents do not have a map and can not distinguish a port leading to v_2 from a port leading to the black hole and thus we need to send n agents trough the both ports, to make sure that n agents get to v_2 and at least one returns.

It might seem that the model is very similar to the oriented graph scenario, where edges have orientation and it is never possible to get back using the same edge. Luckily it is not quite so. If at least one agent returns from v_2, we might be able to distinguish which port leads into still unknown node and which one into the safe node v_2. It is simply accomplished by writing on the whiteboard which port did agents use when they left.

Let us consider example on the Fig. 3.1(b). We see that by using the previous reasoning, in order to decide which port is leading into the next node on the right, we need to send n agents trough each. Thus we send approximately $n\Delta$ agents into the black hole.

We proposed very rough estimates for the number of agents that can die in the studied model. We can see that designing a graph where more than Δ agents die is quite easy. In the second example, it is noticeable that $n\Delta$ agents is more than necessary. With this in mind we present the lower bound on the number of agents needed for finding a black hole.

3.2 Lower Bound on the Size of the Optimal Solution

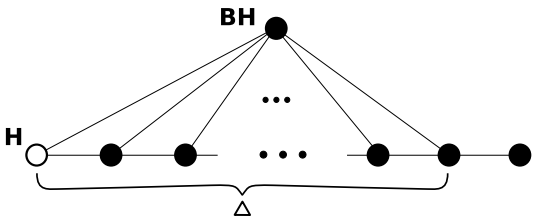


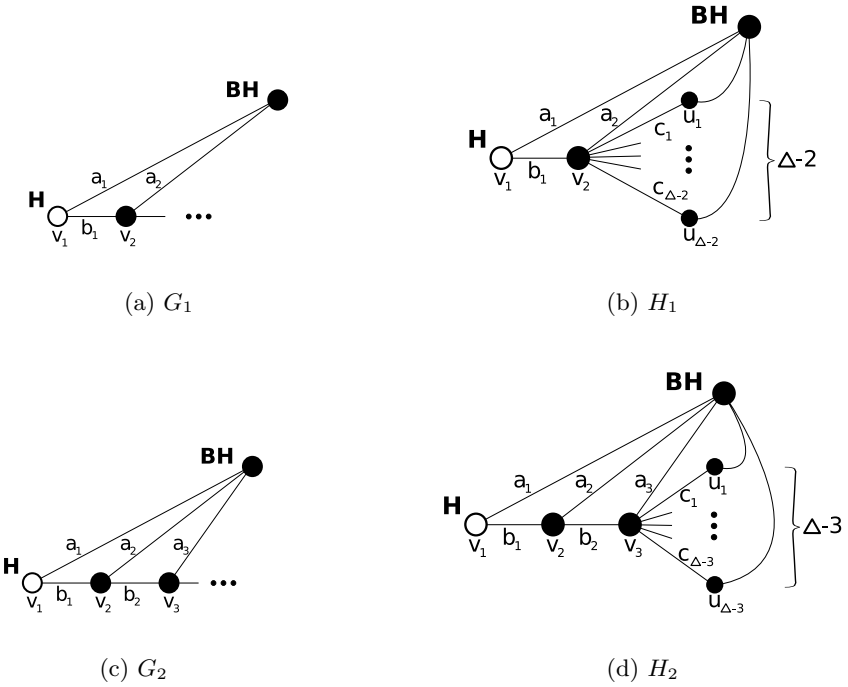
Figure 3.2: Worst case scenario: G_{lb}

We show that any algorithm solving BHS for an arbitrary unknown graph needs at least $\frac{\Delta^2+\Delta}{2}+1$ agents to solve BHS on the graph G_{lb} shown on the Fig. 3.2.

Let us consider the Fig. 3.3(b). Agents start in the home base v_1. They see two indistinguishable edges. The algorithm has to send a number of agents through each edge and then other agents wait until some agents come back. If the algorithm sends a different number of agents through each edge, the adversary can make sure that more agents walk through the edge a_1 and thus end up in the black hole. This means that an efficient algorithm has to send the same number of agents through the both edges a_1 and b_1.

It can be seen, that there have to be at least Δ agents walking through the edge b_1. If the algorithm sends only $k < \Delta$ agents, the adversary can make sure that none of the agents returns to the home base and thus the algorithm fails. There are Δ edges leading out of v_2. Without loss of generality it can be assumed that no two agents use the same edge, one agent uses the edge a_2 and the other $k-1$ agents use the edges $c_1 \dots c_{k-1}$, they get into the nodes $u_1 \dots u_{k-1}$ and from these they use edges leading into the black hole. All k agents die in the black hole and no agent returns.

For the graph H_1, there have to be at least Δ agents using the edge b_1 and thus also Δ agents using the edge a_1. When considering the Fig. 3.3(a) and the graph G_1 it can be observed that for agents waiting in the home base, the graphs G_1 and H_1 are indistinguishable. This means that the algorithm has to act in the same way in both cases, thus also in the graph G_1 there have to be at least Δ agents traveling through the edges a_1 and b_1.



(a) G_1 (b) H_1

(c) G_2 (d) H_2

Figure 3.3: Indistinguishable graphs.

A very similar situation can be seen in the Fig. 3.3(d). As stated before, the adversary can make sure that there are at least Δ agents traveling through the edge a_1. Now the edge b_1 is known to be safe and agents can freely move into the node v_2. Let us assume that the algorithm now knows that a_1 leads into the black hole so we do not have to take it into account. Once again there are two indistinguishable edges a_2, b_2 and there is no point for sending more agents into one of them.

Similarly as in the graph H_1, to make sure that at least one agent gets back into the node v_2 we need at least $\Delta-1$ agents in the node v_3. Otherwise all agents that went into v_3 can end up in the black hole. With the knowledge of v_1 and v_2, the graph G_2 from the Fig. 3.3(c) is indistinguishable from the graph H_2. This implies that the algorithm, which solves BHS for an arbitrary unknown graph, will in the worst case send Δ agents through the edge a_1 and $\Delta - 1$ agents through the edge a_2 in both, G_2 and H_2.

Extending the previous argument on graphs $G_3 \dots G_\Delta$, where G_Δ is similar to G_{lb} we can prove that for $i \in \{1, \dots, \Delta\}$ there are $\Delta - i + 1$ agents traveling through an edge a_i and thus dying in the black hole. Following theorem is thus implied.

Theorem 1 (Lower Bound on the Size of the Optimal Solution)**.** *Any algorithm solving* BHS *problem for an arbitrary unknown graph without the*

knowledge of incoming link needs at least $\frac{\Delta^2+\Delta}{2}+1$ *agents.*

Chapter 4

Size-Optimal Algorithm

Before we describe the algorithm that uses the optimal number of agents, let us look again at the example 3.1(a). There are some ideas left undiscussed and further investigation leads us to the optimal solution.

In the previous chapter, we discussed that if we send n agents into the node v_2 then one will come back, if we design the algorithm in such way that no more than one agent will enter two ports of v_2. The question is whether we really need n agents. The fact that some agents do not leave trough the returning port does not mean that they are dead and useless. Some of them might come back. If $\Delta - 1$ ports eventually lead into the black hole, we could expect that all agents leaving trough these ports would die, but the rest could come back.

The second idea that should be considered are the conditions under which we actually need to send more than one agent trough an unknown port in the first place. The thing is that we need some agents to come back, thus they can not fall into the black hole. If the home base would have $\Delta + k$ ports then it might be enough to send one agent trough each. Maybe Δ of them would fall into the black hole but the rest k of them would be perfectly safe because the black hole has only Δ links leading into it.

We clarify these conjectures while describing the algorithm and proving its correctness.

4.1 Algorithm

We propose the following algorithm for solving the BHS problem on a distributed system without the knowledge of incoming link.

General Ideas of the Algorithm

All ports are classified into four categories, similar to those used in the *cautious walk*, with one extra. Unmarked ports are called *unknown*. When

one or more agents have entered a port, it is marked as *dangerous*. If a port is dangerous and it is known that at least one of the agents that entered such port has been alive afterwards, then the port can be marked as *safe*. When a link has both ports safe, then the link is safe as well and both ports of the link are marked as *explored*. Furthermore, for both explored ports the nodes to which they lead are known and are written on the whiteboard.

One or more nodes of the graph that are connected by safe links are called *safe component*. Safe component which contains the home base is called *main safe component* (MSC).

Every agent searching the graph has a counter of steps it has made. Every time an agent is going to use a port, it leaves a message on the whiteboard stating the port's label, agent's ID and the current state of its counter and it increases its counter as well. Each agent keeps track of how many steps have other agents done. When an agent A enters a node, it looks on the whiteboard and for every agent B that left a message, it updates its information about B's number of steps. An agent always remembers the complete list of agents that visited the current node. The list of counters, known map of the distributed system and a list of agents that entered a dangerous port are kept in the home base as well.

The lists are used as follows. An agent that enters a node u first compares its list of agents in the previous node with the list of agents that left the node u. By comparing the state of their counters, the agent can immediately find out, whether any of the agents went through the same link and it knows by which port as well. If there was such agent, it marks the port explored, goes back to mark the other port as well and returns to the node u.

Second thing an agent has to do when it enters a new node u is to compare its list of all agents and maxima of their counters to the list of agents that left u. If there is a port marked as dangerous and it knows that one of the agents that entered it stayed alive afterwards then it marks this port safe and enters it.

After these two checks, the agent proceeds with its own exploration.

Now we can state rules for the agents, which they obey at all circumstances. We will prove the correctness of this algorithm in the next section.

1. When an agent is in a node that is not part of MSC, it will never enter a dangerous port.

2. When an agent is in a node, it will always make a move, unless the problem is solved or all ports are dangerous. If all ports are dangerous the agent will wait until one of the ports is made safe and then proceeds with the exploration.

3. When agent is in MSC it can enter a dangerous port under following conditions:

- There are no unknown or safe ports in Msc.
- The size of Msc is less than $n - 1$.
- There are k dangerous ports leading out of Msc and $k \leq \Delta$.
- There are less than $\Delta - k + 2$ agents, that entered the port it wants to enter.

This condition ensures that if there are $k \leq \Delta$ dangerous ports at a certain time, then there will be no more than $\Delta - k + 2$ agents entering any of the dangerous ports. Of course, if in some previous state there where k' dangerous ports where $k' < k$ then there might be some ports that were used by $\Delta - k' + 2$ agents which is more than $\Delta - k + 2$. In this case, however, no more agents will enter these ports.

4. An agent has to enter a port when it wants to mark it as safe.

5. An agent moves through an unknown port from node u to node v, upon arrival to the node v it checks its list of agents that visited the node u and compares it with the whiteboard. If it finds out that some agent previously traveled from v through port p and came to u then it has to mark the port p and the whole link as safe[1].

From these rules it follows that when an agent is not able to reach the home base by a safe path, it does not enter any dangerous ports. The agent does not stop exploring. If all ports are dangerous it waits until some port is marked safe. When the agent discovers that some port/link can be marked as safe, then it is its duty to do so.

More Detailed Description

Agents collect information about Msc in the home base. There is a map of Msc with all outgoing ports. For every dangerous port, there is a list of agents that entered this port. List of agents with last known values of their counters is kept in the home base as well. Except form this, all nodes of Msc are labeled and will have marked port that leads to the home base.

The algorithm can be divided into three main parts based on where the agent is and what has it done. First part is for agents stationed in Msc, without any mission. Second part is used when agent is outside of Msc. Last part is designed for an agent that has just returned to Msc.

In the procedure Explore, procedure Search is used to search for a next node with an unknown port. All nodes connected by safe and explored ports can be considered as a subgraph in which agents move as in a directed graph. Dangerous ports are being ignored. Ports marked as safe are directed edges, because on the other side of the link it is not clear which port

[1] By marking a link safe we always mean marking its ports as explored as well.

Procedure 1 LEAVE MSC

```
1: return to home base
   update lists and map
2: if size(MSC)= n − 1 then
3:   if #DANGEROUS ≤ Δ then
4:     FINISH
5:   else
6:     add MY(ID) to waiting list
       sleep until wake up
       goto 1
7:   end if
8: end if
9: if (∃p) AND (p is dangerous) AND
   (∃A) AND (A entered p) AND (counterAtPort(A,p)< counterMax(A))
   then
10:  mark p as safe
     EXPLORE(p)
11: end if
12: if (∃p) AND (p is unknown) then
13:  mark p as dangerous
     EXPLORE(p)
14: else if (∃p) AND (p is safe) then
15:  EXPLORE(p)
16: else
17:  if (∃p) AND (p is dangerous) AND (#DANGEROUS < Δ) AND
     (#ENTERED AGENTS(p) < Δ − #DANGEROUS + 2) then
18:    EXPLORE(p)
19:  end if
20: end if
21: add MY(ID) to waiting list
    sleep until wake up
22: goto 1
```

leads back through that link. Explored ports, with their safe links can be considered as two symmetric directed links.

For such graph a basic algorithm for exploration of directed graphs can be used, which is the main part of the procedure SEARCH. Apart from that, agent always looks for ports that could be made explored or safe, similarly to the initial checks of the procedure EXPLORE.

Procedure RETURN is automatically initiated once an agent returns to MSC.

Procedure 2 EXPLORE

1: $u \leftarrow$ current node
2: $u' \leftarrow$ previous node
3: **if** ($\exists A$) AND (A in u) AND (A in u') AND
(counterAtPort(A,lastPort(A,u))+1 = counterAtPort(A,lastPort(A,u'))) AND
(lastPort(A,u) is dangerous OR safe) **then**
4: $q \leftarrow$ lastPort(A,u)
mark q as explored
enter q
mark lastPort($MY(ID)$,u') as explored
enter lastPort($MY(ID)$,u')
5: **end if**
6: **if** ($\exists A$) AND (A in u) AND (lastPort(A,u) is dangerous
(counterAtPort(A,lastPort(A,u))<maxCounter(A)) **then**
7: $p \leftarrow$ lastPort(A,u)
mark p as safe
EXPLORE(p)
8: **end if**
9: **if** ($\exists p$) AND (p is unknown) **then**
10: mark p as dangerous
EXPLORE(p)
11: **else if** ($\exists p$) AND ((p is safe) OR (p is explored)) **then**
12: SEARCH()
13: **else**
14: add $MY(ID)$ to waiting list
sleep until `wake up`
15: **end if**

Procedure 3 RETURN

1: $u \leftarrow$ current node
2: $u' \leftarrow$ previous node
3: **if** ($\exists A$) AND (A in u) AND (A in u') AND
(counterAtPort(A,lastPort(A,u))+1 =counterAtPort(A,lastPort(A,u'))) AND
(lastPort(A,u) is dangerous OR safe) **then**
4: $q \leftarrow$ lastPort(A,u)
mark q as explored
enter q
mark lastPort($MY(ID)$,u') as explored
enter lastPort($MY(ID)$,u')
5: **end if**
6: find *Home Base*
update lists and map
write down `Wake up`
7: **if** size(MSC)= $n-1$ **then**
8: **if** $\#_{DANGEROUS} < \Delta$ **then**
9: FINISH
10: **else**
11: add $MY(ID)$ to waiting list
sleep until `wake up`
goto 7
12: **end if**
13: **end if**
14: determine outgoing port p previously used
15: **if** p is dangerous **then**
16: mark p as safe
17: **end if**
18: locate p
write down destination of p
EXPLORE(p)

4.2 Correctness

First we state some supporting lemmas and then prove that MSC will keep expanding until the whole graph except the black hole is explored.

Lemma 1. *When an agent A is not in* MSC *and A is in a node v_D with all ports marked as dangerous, one of the ports will be eventually marked as explored.*

Proof. The node v_D is not the home base, thus there is an agent A_0 that entered this node as first using the link l_0. That means that it read an empty whiteboard, wrote down its ID, port by which it left and the state of its counter on the whiteboard. All the agents that enter after the agent A_0 will read its message on the whiteboard.

Now all ports are marked as dangerous thus there exists an agent A_x that left through the link l_0. When the agent A_x arrives to the other end of the link l_0, it finds out that the agent A_0 was there just before the node v_D and thus it can identify a port of the link l_0. He marks the port as explored and goes back to v_D and marks the other port as explored as well, thus one of the dangerous ports has been marked explored and the agent A can leave. □

The use of the first agent A_0 in the previous proof has its reason. By picking the first agent it is clear that the agent A_x read the both messages from A_0. Important fact is that before A_x performs any other action and thus risks falling into the black hole it has to make l_0 safe. This lemma can be expanded for more than one node.

Corollary 1. *When an agent is not in* MSC *and it is in a safe component with all outgoing ports marked as dangerous, one of the ports will be eventually marked as explored.*

Proof. Once again, there was an agent A_0 entering one of the nodes of the safe component from outside of the safe component as first, by the link l_0. Thus there is an agent that left the safe component through the link l_0 and will mark it safe. □

Lemma 2. *A link that has both ports safe will be marked as safe, and the ports as explored. Furthermore, it will be marked safe by one of the agents that marked its ports safe.*

Proof. Link l has ports p and p', one of the ports had to be marked safe first, let us assume, that it was the port p. If the port p is safe, it means that an agent A_1 traveled already through it. Then second agent A_2 with the knowledge that A_1 survived, marked it safe and traveled through it as well.

Agent B_2 that is going to mark the port p' as safe can do it only after A_2 does. This means that A_1 has already emerged from the link l and thus B_2 reads its messages on the whiteboard. When B_2 emerges on the other side of the link l it finds previous message of A_1 and knows that it can mark the whole link safe and its ports as explored. □

In the following lemma we focus on the agents exploring outside of MSC. These agents are not allowed to enter any dangerous ports. We show that there are always enough agents outside, so that at least one does not die in the black hole. Right now we will expect that there are enough agents in the main safe component and that it is always possible to send out more agents if needed. The actual number of agents that are really needed is shown in the section 4.3.

Lemma 3. *For* MSC *of size smaller than* $n-1$ *the following statement holds. When there are no more agents that could leave* MSC *according to the third rule, then at least one of the agents outside of* MSC *does not enter the black hole.*

Proof. The fact that no more agents could leave MSC can only mean that all k outgoing ports of MSC are dangerous and through each one went at least $\Delta - k + 2$ agents if $k \leq \Delta$ and at least one agent if $k > \Delta$, thus no more agents are allowed.

Let l be the number of ports leading out of MSC and into the black hole, we know that $l < k$ unless the size of MSC is $n-1$.

When $k > \Delta$, we do not need all k ports being entered by at least one agent. We expect that we have enough agents, so that there are at least $m \geq \Delta + 1$ outgoing ports that were used by at least one agent. This means that at least $m - l$ agents survived after leaving MSC. There are $\Delta - l$ ports leading into the black hole from outside of MSC and thus maximally $\Delta - l$ agents can enter these ports and die. Thus we get that at least $(m-l)-(\Delta-l)$ agents in total will survive outside MSC. As we mentioned, $m \geq \Delta + 1$ and thus $m - \Delta \geq 1$, which means that at least one agent survives.

Let us consider $k \leq \Delta$. We can expect that all allowed agents entered a port meaning that at least $\Delta - k + 2$ agents entered each port. If l ports were dangerous then $(k - l) \times (\Delta - k + 2)$ is the number of agents that got out of MSC and not into the black hole. From these agents, only $\Delta - l$ will die in the black hole outside MSC. Thus the number of surviving agents can be expressed:

$$\begin{aligned} \#_s &= (k-l) \times (\Delta - k + 2) - (\Delta - l) \\ &= k\Delta - k^2 + 2k - l\Delta + lk - 2l - \Delta + 2 \\ &= -k^2 + k\Delta + 2k - \Delta + l \times (k - \Delta - 1) \end{aligned}$$

We know that Δ and k are fixed and that coefficient of l: $(k - \Delta - 1)$ is negative, so $\#_s$ drops with raising l. Number of ports leading from MSC

into the black hole is bounded by the number of all outgoing ports, $l < k$. Thus the smallest number of agents survive when $l = k - 1$:

$$\begin{aligned} \#_s &= (k_s - l) \times (\Delta - k + 2) - (\Delta - l) \\ &= (k - (k-1)) \times (\Delta - k + 2) - (\Delta - (k-1)) \\ &= 1 \end{aligned}$$

Which again means that at least one agent outside MSC survives. □

Lemma 4. *For* MSC *of size smaller than* $n - 1$ *the following statement holds. When there are no more agents that could leave* MSC *according to the third rule, then at least one of the agents that left* MSC *returns back into* MSC.

Proof. What we are going to show is that one of the surviving agents will eventually enter a port leading back to MSC and thus it returns.

From the Lemma 3 we know that at least one of the agents that left MSC survives. Agents work in such way that they always enter some port, unless all are dangerous in which case they wait. Lemma 1 further tells us that even though an agent does not enter a dangerous port and waits, it does not get stuck somewhere either. From these three statements it is quite clear that the agent never really stops moving, once it leaves MSC.

There are only n nodes and thus less than n^2 links between them. Every link has two ports and every port can have four states. Thus there is a finite number of ports outside MSC and they have only finite number of states together.

We will show that agents not only never stop exploration, but when they move through the graph, the states of the ports are always being changed. This means that either an agent itself changes a state of a port or it gets into a situation in which the state of a port will be inevitably changed by some other agent.

Agents can encounter the following situations:

1. When an agent is in a node with an unknown port then it enters it and changes its state to dangerous.

2. If there are only dangerous ports, then the agent waits until one of the ports is made explored, we proved in the Lemma 1 that it will happen.

3. If there are only dangerous, safe and explored ports in a node the agent searches the subgraph of such nodes using only safe and explored ports. When the agent finds an unknown port, it stops the search protocol and enters the port. If there are no reachable unknown ports, then the agent can wait, because one of the dangerous ports will be changed by other agent. To prove this, we could use the same reasoning as in the proof of Lemma 1.

If surviving agents never stop changing states of the ports outside MSC and there is only finite number of ports, eventually some agent has to change a state of a port leading back to MSC, put in different words, the agent will enter it. □

We prove that MSC will be expanding until it contains $n-1$ nodes. At the beginning, only the home base is a part of MSC and it has all ports marked as unknown. Agents start the exploration by entering these ports. We now proceed with a general case, there is the main safe component, with k outgoing ports.

Let us assume that MSC does not expand. Once again we use the fact that there is only finite number of ports leading out of MSC and they can be of only 3 different states. If a port would be marked as explored, it means that MSC is being expanded. The fact that MSC does not expand implies that there must be a state of MSC that does not change any more.

By state of MSC we will understand state of all outgoing ports, with additionally, for each safe outgoing port a list of agents that entered it. When an agent returns to MSC, it leaves it through the safe port, which it has already used before, in such case the agent will be on the list twice. Moreover, if an agent is entering outgoing port second time, it knows where does it lead and it writes the information on the whiteboard in the node of that port.

It is important to note that if at least n agents enter a safe port, then there is certainly an agent that entered the link from the other side, because unknown links are entered first. Thus when an agent comes out of such port into the node of MSC it does not know by which port it arrived but it either reads information left by an agent that entered the port twice and is able to mark the port explored. Or the agent leaves information about itself on the whiteboard so that the next agent which travels through that port is able to mark it explored afterwards.

If the number of agents that entered the safe port would be increasing, MSC would eventually expand, so it has to be stable as well.

There are only two options for the stable state. First one is that there are agents in MSC but they are all waiting in the home base. This further means that all outgoing ports are dangerous and no other agents are allowed to leave through these ports, thus they are waiting. The second option is that there is no agent in MSC otherwise it would exit and thus change the stable state.

Both these states would change by any agent returning to MSC. In the first case, an agent that enters MSC had to leave the MSC before by a port p. Now that the agent is back in MSC it can, and will, mark the port p as safe, thus changing the state of MSC, allowing more agents to continue the exploration.

In the second case, the fact of an agent returning to Msc changes the stable state. Even if the state would not be changed directly, the agent had previously left Msc and traveled through outside of Msc by some path. Part of this path might have become a part of Msc but there must be one port which now leads out. This port is either dangerous and the agent can mark it as safe, or it was marked as safe in the meanwhile. In such case it will do as mentioned above, leave information about the port's destination, add itself into the list and enter it again.

Lemma 4 proves that when the number of nodes of Msc is less than $n-1$ then every time there are no more agents able to leave Msc at least one agent returns to Msc. We formulate our statement in the following theorem.

Theorem 2 (Correctness)**.** *Agents with their exploration, will expand the main safe component until it contains $n-1$ nodes, thus exploring whole graph except the edges possibly leading into the black hole. By this they finish the Black hole search.*

4.3 Upper Bound on the Size of the Optimal Solution

In the proof of the lower bound value we used the graph G_{lb} on the Fig. 3.2. Note that our algorithm uses optimal number of agents to fulfill Bhs on such graph — $\frac{\Delta^2+\Delta}{2}+1$. We do not need to prove this statement for the purposes of the upper bound proof.

Even more, let us order all ports leading into the black hole based on the number of agents that used them in descending order, these numbers make a following sequence: $p_1 = \Delta, p_2 = \Delta - 1, \ldots p_{\Delta-1} = 2, p_\Delta = 1$. This corresponds with the claim from the lower bound proof that for each edge a_i leading into the black hole, there had to be $\Delta - i + 1$ agents that passed through this edge.

Theorem 3 (Upper Bound on the Size of the Optimal Solution)**.** *Algorithm for the Black hole search problem obeying conditions stated before will succeed with $\frac{\Delta^2+\Delta}{2}+1$ agents in the worst case.*

Proof. Consider a sequence of Δ numbers similar to the one mentioned before, for $i \in \{1, \ldots, \Delta\}; p_i = \Delta - i + 1$.

Let us consider graph G_{ub}, with the degree of the black hole Δ, where our algorithm needs more than $\frac{\Delta^2+\Delta}{2}+1$ agents to solve Bhs. In other words, there are at least $\frac{\Delta^2+\Delta}{2}+1$ agents that die in the black hole. Now again we take the numbers of agents that used a port leading into the black hole and order them in descending order. These numbers form a non-increasing

sequence of length Δ: $q_1, q_2 \ldots, q_\Delta$. We stated previously that there are more agents ending up in the black hole in the graph G_{ub} and thus:

$$\sum_{i=1}^{\Delta} q_i > \sum_{i=1}^{\Delta} p_i = \frac{\Delta^2 + \Delta}{2}$$

Both sequences have the same lengths and thus there must exist index j where $q_j > p_j$. We know that $q_j > p_j = \Delta - j + 1$, and thus $q_j \geq \Delta - j + 2$. Now we have to consider what does this mean in terms of the rules for our algorithm.

In the last rule about the number of agents entering a dangerous port at certain conditions, we state that for $k \leq \Delta$ maximally $\Delta - k + 2$ agents enter. When at least $\Delta - j + 2$ agents enter dangerous ports it means that:

$$\begin{aligned} \Delta - j + 2 &\leq \Delta - k + 2 \\ j &\geq k \end{aligned}$$

For all ports corresponding to the numbers $q_1, \ldots q_j$, there is an agent that entered these ports last. For this last agent, some conditions must have been fulfilled. There were no unknown ports in Msc. Number of the nodes in Msc was less than $n - 1$, put in other words, there were still some nodes that were not part of Msc. And at last, there were maximally k dangerous ports at that moment, where $k \leq j$ from above stated equation.

Numbers $q_1, \ldots, q_j$ correspond to j different ports leading **into** the black hole. If the graph is biconnected then we are missing at least one dangerous port, that **did not lead** into the black hole. So the last agent would obviously have to break one of the conditions by entering the dangerous port, which leads us to contradiction. □

Corollary 2 (Size Optimal Solution)**.** *Proposed algorithm provides size optimal solution for the Black hole search problem without the knowledge of incoming link.*

Conclusion

In this work, we have studied the Black hole search problem without the knowledge of incoming link. We have shown that this modification has effects on the size of the solution.

We provided lower bound on the number of agents that are necessary to locate the black hole. Any correct algorithm solving the Black hole search problem without the knowledge of incoming link needs at least $\frac{\Delta^2+\Delta}{2}+1$ agents. The algorithm is presented with the proof of correctness, it uses the optimal number of agents in the worst case. The cost of the algorithm and bounds on the optimal cost of the solution are left for further investigation.

Bibliography

[1] Jérẽmie Chalopin, Shantanu Das, and Nicola Santoro. *Distributed Computing*, chapter Rendezvous of Mobile Agents in Unknown Graphs with Faulty Links, pages 108–122. LNCS. Springer Berlin / Heidelberg, 2007.

[2] Colin Cooper, Ralf Klasing, and Tomasz Radzik. Searching for black-hole faults in a network using multiple agents. In *Proceedings of the 10th International Conference on Principles of Distributed Systems (OPODIS 2006)*, volume 4305 of *Lecture Notes in Computer Science*, pages 320–332. Springer Verlag, December 2006.

[3] J. Czyzowicz, S. Dobrev, R. Královič, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In *SIROCCO 2009*, Lecture Notes in Computer Science. Springer, 2009. to appear.

[4] Stefan Dobrev, Paola Flocchini, Rastislav Královič, Giuseppe Prencipe, Peter Ružička, and Nicola Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[5] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. *Lecture Notes in Computer Science*, 2180:166–179, 2001.

[6] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: optimal mobile agent protocols. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 153–162, New York, NY, USA, 2002. ACM.

[7] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *OPODIS*, volume 3144 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2003.

[8] Stefan Dobrev, Paola Flocchini, and Nicola Santoro. *Structural Information and Communication Complexity*, chapter Improved Bounds for Optimal Black Hole Search with a Network Map, pages 111–122. LNCS. Springer Berlin / Heidelberg, 2004.

[9] Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Cycling through a dangerous network: A simple efficient strategy for black hole search. pages 57–57, 2006.

[10] Paola Flocchini, David Ilcinkas, and Nicola Santoro. Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *DISC '08: Proceedings of the 22nd international symposium on Distributed Computing*, pages 227–241, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Hardness and approximation results for black hole search in arbitrary networks. *Theoretical Computer Science*, 384(2–3):201–221, 2007.

[12] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Approximation bounds for black hole search problems. *Networks*, 52(4):216–226, 2008.