Exploring an unknown dangerous graph with a constant number of tokens

B. Balamohan[e], S. Dobrev[f], P. Flocchini[e], N. Santoro[h]

[a]*School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada.*
[b] *Institute of Mathematics, Slovak Academy of Sciences, Bratislava, Slovak Republic. stefan.dobrev@savba.sk*
[c] *School of Computer Science, Carleton University, Ottawa, Canada. santoro@scs.carleton.ca*

Exploring an unknown dangerous graph with a constant number of tokens

B. Balamohan[e], S. Dobrev[f], P. Flocchini[e], N. Santoro[h]

[d]*School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada.*

[e] *Institute of Mathematics, Slovak Academy of Sciences, Bratislava, Slovak Republic. stefan.dobrev@savba.sk*

[f] *School of Computer Science, Carleton University, Ottawa, Canada. santoro@scs.carleton.ca*

Abstract

Consider a team of asynchronous agents that must explore an unknown graph G in presence of a black hole, a node which destroys all incoming agents without leaving any observable trace. Interagent communication and coordination is achieved using tokens that agents can pick up, carry, and drop on the nodes. It is known that, if the agents have a map of G, the problem can be solved with a constant number of tokens. The question we study is under what conditions it is possible to explore with $O(1)$ tokens if the agents have no map of G. The contribution of this paper is to provide a definite answer to this question.

We first prove the unexpected negative result that if only a constant number of tokens are available, then $\Delta + 1$ agents are *not* sufficient, where Δ is the maximum node degree. We also prove that, regardless of the team size, two tokens are *not* sufficient. In other words, any solution protocol using a constant number of tokens, must use at least $\Delta + 2$ agents and at least 3 tokens.

We then show that these bounds are *tight* by presenting a protocol that allows the exploration of an unknown anonymous dangerous graph using only 3 tokens and $\Delta + 2$ agents. At the core of our solution is a novel token-based asynchronous communication protocol, which is of independent interest.

Our algorithm assumes that the number of agents is known; in case the number of agents is sufficient but unknown, we show that the problem has a simple solution that uses only five tokens.

Keywords: Black Hole, Dangerous Networks, Unknown Graph Exploration, Tokens, Asynchronous Mobile Agents, Distributed Algorithms
2010 MSC: 68W15, 68W40, 68Q17, 68M10

1. Introduction

1.1. The Problem

The classic problem of exploring an unknown graph using a team of one or more mobile agents has been extensively studied since its initial formulation in 1951 by Shannon [31]. It requires the agents, starting from the same node, to visit within finite time all the nodes of a graph whose topology is unknown to them.

Different instances of the problem exist depending on a variety of factors, including the (a)synchrony of the agents, the presence of distinct agent identifiers, the amount of memory, the coordination and communication tools available to the agents, etc. (e.g., see [1, 3, 12, 13, 14, 23, 24, 30]).

Notice that, except for trees, the exploration of anonymous graphs is possible only if the agents are allowed to mark the nodes in some way; various methods of marking nodes have been used by different authors ranging from the weak model of *tokens* to the most powerful model of *whiteboards*. The majority of the solutions proposed in the literature succeed in their task assuming that the network is *safe* for the agents.

The exploration problem has been examined also when the network is *unsafe*; the danger considered is the presence in the network of a *black hole* (Bh), a node that disposes of any incoming agent without leaving any observable trace of this destruction (e.g., [2, 6, 7, 8, 9, 10, 11, 15, 16, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 32]). Note that such a dangerous presence is not uncommon. For example, a network site that contains a local program (virus) that destroys incoming mobile code is a black hole; similar effect can have just the malfunctioning of the site's communication system. From a computational point of view, the undetectable crash failure of a node in an asynchronous network transforms that node into a black hole.

The problem of exploring the graph in spite of this harmful presence is called *black hole search* (Bhs). It requires the team of agents to explore the network and, within finite time, discover the location of the Bh. More precisely, at least one agent must survive, and any surviving agent must have constructed a map of the network indicating the edges leading to the Bh.

It is known that, when the graph is unknown (i.e., the agents have no map of the network) some information must however be available for Bhs to be solvable; in particular, some metric information such as the number n of nodes and the number m of links, or the number of safe links (i.e., not leading to the black hole) is necessary for termination. Furthermore, if the graph is unknown, at least $\Delta + 1$ agents are needed, where Δ is the maximum node degree in the graph [17].

The problem of asynchronous agents exploring a dangerous graph has been investigated assuming powerful inter-agent communication mechanisms: *whiteboards* at all nodes. In the whiteboard model, each node has available a local storage area (the whiteboard) accessible in fair mutual exclusion to all incoming agents; upon gaining access, the agent can write messages on the whiteboard and can read all previously written messages. This mechanism can be used by the agents to communicate and mark nodes or/and edges, and has been employed e.g. in [12, 15, 17, 18, 23]. Using whiteboards, Bhs can be solved with a minimal team size and performing a polynomial number of moves (e.g., [15, 17, 18]). Indeed, the whiteboard model is very powerful, providing not only direct and explicit communication but also a mechanisms for leader election (and assigning distinct ids) when agents are co-located, and FIFO capabilities even when the network is not so.

An alternative coordination and communication mechanism is provided by the use of identical *tokens* (or pebbles) that an agent can hold, carry while moving, place on nodes, and pick up from a node. The *token* model is generally viewed as less taxing of the system resources than whiteboards, and has been employed in the exploration of safe graphs (e.g., [3, 13])); it gives rise to additional research questions, in particular: How many tokens are needed? Can tokens be placed only on nodes (pure token model) or also on edges (hybrid token model)? etc.

These questions focus on how much *space* is required from the system by an asynchronous token-based solution. These questions are also relevant because of the relationship between solutions in the two models. In fact, any protocol which uses at most t tokens in the pure token model, can be directly implemented using whiteboards of size at most $\lceil \log t \rceil$ bits; note that h tokens in the hybrid model correspond to $t = h\Delta$ tokens in the pure token model. This measure t, which we call the *token load*, is clearly important in that it determines the usability of token-protocols in the whiteboard model, and provides a simple mechanism to transform[1] complexity results from the token setting to the whiteboard one. Importantly, the transformation preserves also other costs (such as total number of moves by agents and time).

The recent token-based solution for graphs of *known* topology with minimum number of agents and $O(1)$ pure tokens opens immediately the question of whether a similar result holds also if no map is available to the agents. In other words, is it possible to locate the black hole in a graph of *unknown* topology using a constant number of tokens? and if so, under what conditions?

The current results for unknown graphs are not very useful and provide no hints. The existing whiteboard solution uses $O(\log n)$ bits whiteboards [17], implying a solution that uses $t = O(n^2)$ tokens; the existing token-based solution uses $t = O(\Delta^2)$ tokens [16]. Similar questions have been recently raised, in the

[1] Note that an automatic transformation exists also in the other direction: any solution that uses whiteboards of size s bits can be implemented with a token load at most $t = n2^s$.

case of synchronous rings and synchronous tori [5, 6].

In this paper we provide a definite answer to those questions.

1.2. Main Contributions

We first examine how many agents are needed to locate a black hole without a map of the graph. We prove the unexpected negative result that if $t = O(1)$, then $\Delta + 1$ agents are *not* sufficient. That is, the minimum team size to solve the problem with a map $(\Delta + 1)$ is not sufficient to solve the problem without a map if the number of tokens is a fixed constant. This must be contrasted with the case when the graph is known: in that case, constraining the number of tokens to be constant does not affect the minimum team size.

We next examine how many tokens are really needed, i.e. how small t can be. We prove that, regardless of the team size, $t = 2$ tokens are *not* sufficient.

Summarizing, we prove that any solution protocol using a constant number of tokens, must consist of at least $\Delta + 2$ agents and use at least $t = 3$ tokens.

We then show that these bounds are tight: it is indeed possible for a team of $k \geq \Delta + 2$ agents to locate a black hole in an unknown graph using only 3 tokens, provided k is known. The protocol achieving this result is rather complex; the complexity comes from the need to achieve control, coordination and communication among the agents, while keeping the number of tokens (and hence also the number of observable configurations) to a minimum. At the core of our solution is a simple, but novel token-based communication protocol using three tokens to communicate arbitrary information between the agents.

Although not the primary concern of this paper, the proposed protocol is rather efficient in terms of total number of agent's moves: in fact the agents perform at most $O(mn)$ moves in total. We show that $\Omega(n^2)$ moves are necessary in the worst case; hence, in sparse graphs, our protocol is also move-optimal.

We consider also the number of *token transitions*, i.e. the overall number of times a token has been picked up or dropped down. The number of transitions in our protocol is $O((m + n\Delta) \log n)$, which is bounded, and depends only on the input graph, not on the whims of the asynchronous scheduler.

Our algorithm assumes that the number of agents is known; in case the number of agents is sufficient (i.e., $k \geq \Delta + 2$) but unknown, we show that the problem has a simple solution that uses only $k = 5$ tokens.

2. Definitions and Basic Constraints

2.1. Model and Definitions

The network environment is modelled as a simple undirected edge-labelled graph $G = (V, E)$ of $|V| = n$ nodes and $|E| = m$ edges. The nodes are *anonymous*; that is, they do not have distinct identifiers that can be used in the computation.

At each node $x \in V$ there is a distinct label (called a port number) associated to each of its incident edges. Without loss of generality, we assume that the labels at $x \in V$ are the consecutive integers $1, 2 \ldots, d(x)$, where $d(x)$ denotes the degree of x. We denote by $\Delta(G)$ (or simply Δ) the maximum node degree in G. If $(x, y) \in E$ then x and y are said to be neighbours.

Operating in G is a team $\mathcal{A}$ of mobile computational entities called *agents*; the agents are anonymous (i.e., they have no distinct identifiers). We impose no a priori restriction on the agents' computational power nor on the size of their memory[2]. The agents do not know G; however, they do know the number of nodes n and the number of edges m (as discussed in Section 2.2). The agents obey the same set of behavioural rules (called the *algorithm* or protocol). Initially, they are all in the same state, and enter the network from the same node HB, called their *homebase*, but not necessarily at the same time.

In the system, there are t identical tokens, which can be held by agents or placed at nodes. Initially, all tokens are placed at the homebase. Each node provides a specific shared area for the visiting agents, called the *centre*, where tokens can be placed. Access to the centre is in fair mutual exclusion: any agent wanting to access the centre will succeed in finite time. The agents can move from node to neighbouring node in G, and carry with them the tokens they are holding. Links are not necessarily first in, first out (FIFO).

Upon arriving at a node, an agent requests to access the centre. Once the request is granted, using the number of tokens found there as an input, it executes a protocol (the same for all agents) to determine whether to drop or pick up some tokens, and how many; once that is done, it computes the next destination: either a neighbouring node or the current node (a null move); it then terminates its access to the centre and moves to the computed destination. A null move is treated like any other move: after performing it, the agent is considered as arriving to the node. Note that, when at a node, an agent has no knowledge of the other agents currently there nor of the tokens they hold.

For the reader looking for the practical implications of such a setting, consider the large applicative research field of software mobile agents from which BHS and many other problems in distributed mobile computing originate. In this field, the agent is really mobile code that is executed at the site where it has moved (i.e., it has been sent). For security and privacy reasons, the executed code has access only to its own variables (e.g., its tokens) and to the dedicated shared area (e.g., the centre) but has no access to other mobile code nor their variables (e.g., the other agents and the tokens they are holding). For a recent review of the field, the

[2] In our algorithms, the size of the agent's memory and all computational steps performed by the agent are (low) polynomial in the network size.

reader is referred to [4] and the chapters therein.

The system is *asynchronous* in the sense that every action an agent performs (moving, computing, token drop and pick) takes an unpredictable (but finite) amount of time; similarly, the interval of time until a request to access the centre is granted is finite but unpredictable.

The network contains a *black hole*, a node $BH \in V$ that destroys any incoming agent without leaving any detectable trace of that destruction. The goal of a *black hole search* algorithm $\mathcal{P}$ is to identify the location of the black hole; that is, at least one agent survives and all the surviving agents within finite time know the edges leading to the black hole. Knowledge of the location of the black hole is *exact* if all and only the links leading to the black hole have been determined; it is *partial* if all the links leading to the black hole have been determined (i.e., some links might be incorrectly suspected to lead to the black hole). Termination of a solution protocol is *explicit* if within finite time all surviving agents enter a terminal state and have the same information on the location of the black hole in the graph. We are interested in protocols that explicitly terminate with exact information.

The primary complexity measures we are interested in are team size k (i.e. the number of agents used) and token load t (the total number of tokens in the system). The secondary complexity measures are the *number of moves* (of the agents between the nodes of the network) and the number of *token transitions* (each token pick-up or drop-off is counted; an agent's action in which it observes the current token configuration at the node and decides to do nothing is not counted). These last two measures are worst-case w.r.t. all possible input networks of given size, black hole locations and scheduler actions. Consider the situation, called *deja vu*, of an agent that, upon performing a null move, accesses the centre and finds no change in the number of tokens. If the reaction to some deja vu is for the agent to move to another node and/or change the token configuration, then due to asynchrony the number of moves and/or token transitions will be unbounded; hence our focus will be on algorithms where, in cases of deja vu, the agent neither moves to another node nor changes the token configuration.

2.2. Basic Limitations

Some basic limitations are well known:

Lemma 1. *[18]*
(1) If G has a cut vertex different from the homebase, Bhs *is unsolvable.*
(2) In the absence of other topological knowledge, if n is not known, Bhs *is unsolvable.*

As a consequence, we assume that G remains connected once the black hole is removed (e.g., G is biconnected), and that n is known to the agents. Knowledge of

n implies that an algorithm could terminate as soon as $n-1$ safe nodes have been visited. However, due to asynchrony, explicit termination (even with just partial knowledge) may be impossible, as expressed by the following simple observations:

Lemma 2. *[17]*
In absence of any other topological knowledge, explicit termination is not possible
(1) if only n is known;
(2) if only m is known;
(3) if only n and an upper bound $\Delta' \geq \Delta$ are known.

On the other hand

Lemma 3. *If n and m are known, explicit termination with exact knowledge is possible.*

Proof. Since n is known, an algorithm may explore until all but one node is determined to be safe. At this point, the algorithm knows the degrees of all but one node. Since m is known, and since $2m$ is the sum of the degrees of all the nodes, the degree of the black hole can be determined in finite time. Hence the links leading to black hole can be identified accurately. After this is done, the surviving agents can terminate. □

As a consequence, to ensure explicit termination, we assume that m is also known. Our algorithm can be easily modified for the case in which m is unknown; in that case, the termination will be implicit

3. Impossibilities and Lower Bounds

Recall that $\Delta+1$ agents are sufficient to locate the black hole if the token load is a function of the size of the network, and thus a priori unbounded [16, 17]. In this section we show that if the total number of tokens is bounded by a constant, $\Delta+1$ agents no longer suffice. Furthermore, we prove that 2 tokens are insufficient to solve the Bhs problem, regardless of the number of agents. This means that any black hole location algorithm must use at least $\Delta+2$ agents and at least 3 tokens—and we provide matching upper bound in the next section. Finally, we prove that any algorithm for Bhs using $(2-\epsilon)\Delta$ agents incurs cost of $\Omega(\epsilon n^2/\Delta)$ moves, showing that our algorithm is optimal (asymptotically) also in the number of moves.

3.1. Framework and Notation

The proofs of impossibility and lower bound results are based on a game between the algorithm and an adversary. The adversary has the power to choose the graph, the port labelling, and the asynchronous timing of the edges; the algorithm specifies agent movement; w.l.o.g., we can assume the steps of the algorithm and the adversary alternate. At any moment of time, the agents can be classified as *free* or *blocked*. The free agents are those that are located at a node, or are traversing an already explored edge (an edge is unexplored if no agent has so far crossed it in either direction). The algorithm specifies in each step the movement of the free agents located at the nodes; w.l.o.g. we may assume the algorithm specifies the movement (the port taken) of at most one free agent in each step. The *blocked* agents are the agents that are in transit over the yet unexplored edges. In each step, the adversary lets the free agents crossing edges reach their destinations. Furthermore, if the algorithm specifies that an agent enters an unexplored edge with a given label, the adversary chooses which of the incident unexplored edges the agent enters and assigns that label to the edge. Finally, the adversary can unblock a blocked agent and allow it to reach its destination.

The adversary is limited by the requirement that it can indefinitely block only the agents crossing the links entering the black hole. However, due to asynchrony, the unknown topology and the unknown location of the black hole, this means that the only case when the adversary cannot block the blocked agents indefinitely is if there are at least $\Delta + 1$ blocked agents or there are two blocked agents traveling on edges leaving the same node. In such a case, the algorithm can wait until the adversary unblocks one of the blocked agents; in all other cases, the algorithm must specify an agent movement during its turn.

3.2. Impossibilities and Lower Bound Results

Theorem 1. Bhs *is unsolvable by $\Delta+1$ agents without a map, even if the number of nodes and edges is known.*

Proof. By contradiction, assume $\Delta + 1$ agents are sufficient. The unknown graph G in which the agents must locate the black hole is one of G_a, G_{b_1} or G_{b_2} shown in Figure 1, where Hb is the homebase, and Bh is the black hole; the choice of which graph G the game is played on is up to the adversary. We now show that, even if the agents have the map of these three graphs, the uncertainty about which one G really is allows the adversary to send all $\Delta + 1$ agents to the black hole.

Starting from the homebase Hb $= x_0$, whenever the algorithm sends an agent over the first unexplored link from node x_i $(0 \leq i \leq \Delta - 2)$, the adversary blocks it. Since G might be graph G_a, the algorithm must send an agent also over the second unexplored link incident on x_i; otherwise, no agent would ever reach node $x_{\Delta-1}$ and beyond. At this point, the adversary is forced to unblock one of these two links; it will unblock the link leading to node x_{i+1}. Hence, within finite time, $\Delta - 1$ agents are blocked and an agent reaches node $x_{\Delta-1}$.

Since the adversary might have chosen $G = G_a$ and send all blocked agents to Bh, the remaining two free agents must now explore the remainder of the graph

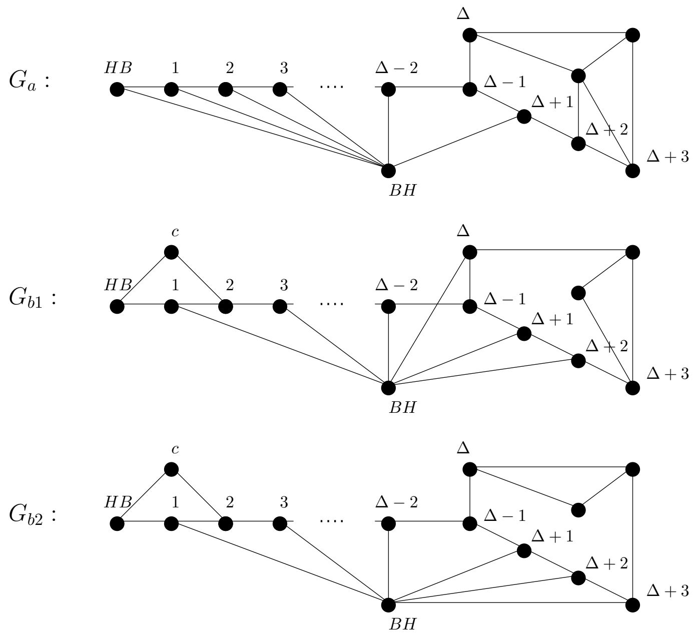

Figure 1: The graphs used in Theorem 1. The dotted part represents a path, with each node of the path connected to the black hole and to its two neighbours in the path.

without waiting for help from another agent. Hence, there must be an agent that reaches either x_Δ or $x_{\Delta+1}$ and leaves it via an incident link. As nodes x_Δ and $x_{\Delta+1}$ look the same to the algorithm, the adversary can force this agent to take the link from $x_{\Delta+1}$ to Bh. Since the input graph can still be G_a, the last available agent must continue exploration of the remainder of the graph. It has one safe choice (exploring the second link of $x_{\Delta+1}$ leading to $x_{\Delta+2}$), but after that it has only two options: either it explores a link incident to x_Δ or a link from $x_{\Delta+2}$. In the first case, the adversary chooses $G = G_{b_1}$, in the second case it chooses $G = G_{b_2}$.

Notice that at this point, all agents are blocked on links leading to the black hole, except for the two agents traversing the edges leading to node c. The adversary then unblocks those two agents; to finish the exploration of the graph, these agents must reach node $x_{\Delta-1}$ and go beyond.

Regardless of the choice, $G = G_{b_1}$ or $G = G_{b_2}$, made by the adversary, at each of the nodes $x_3, x_4, \ldots, x_{\Delta-2}$ on the path to node $x_{\Delta-1}$, there is a link leading to the black hole, which the two agents must avoid. This means there are $2^{\Delta-5}$ possible labeled paths from x_3 to $x_{\Delta-1}$, only one of which is safe. However, the algorithm can place only C tokens on the Δ nodes $c, x_0, ..., x_{\Delta-2}$, resulting in $O(\Delta^C)$ possible token configurations. Hence, for large enough Δ the adversary can force also these last two agents to enter the black hole while they are moving from node x_3 to node $x_{\Delta-1}$. □

As a consequence, at least $\Delta + 2$ agents are required to locate the black hole.

We now focus on the needed token load t, i.e., how many tokens are really necessary.

Theorem 2. *With two tokens initially stored at the homebase and no additional tokens, no team of anonymous identical agents can locate the black hole in an unknown network with homebase of degree at least two, even if the number of nodes $n \geq 4$, edges $m \geq n + 1$ and maximal degree $\Delta \geq 3$ are known beforehand.*

Proof. By case analysis over the possible algorithms, we show that in every case the adversary can cause the algorithm to fail.

Let us call an agent that is in initial state a *fresh* agent. Note that thanks to asynchrony, it is sufficient to limit ourselves to algorithms which perform actions only on wake-up, when arriving to a node, or when noticing a change in the number of tokens at a node they are currently waiting at.

If the algorithm's action for a fresh agent seeing two tokens at the homebase does not change the number of tokens there, then the adversary wakes-up the fresh agents one by one and either all start waiting forever at the homebase, or all leave the homebase towards the same vertex v_1. In the first case, the algorithm obviously fails, in the second case the adversary places the black hole at v_1 and the algorithm fails.

Consider now the case that a fresh agent seeing two tokens at the homebase picks both of them up. If the agent leaves the homebase, the adversary will direct it towards the black hole and both tokens will be lost, leaving no possibility for the remaining agents to communicate and the algorithm fails. On the other hand, if the first agent starts waiting at the homebase, it will wait forever, again effectively eliminating the tokens from the play and causing the algorithm to fail.

Therefore, the algorithm for a fresh agent seeing two tokens at the homebase has only two options: (*O1W*): pick one token and wait at the homebase until the number of tokens changes to 0, or (*O1M*): pick one token and move to a neighbouring node v_1.

Note that if a fresh agent seeing one token does not pick it up, the adversary can make all remaining fresh agents do the same thing. If all of them wait, the adversary places the black hole at v_1 and the algorithm fails; if all of them leave towards v_2 then the adversary places the black hole at v_2 and the single remaining agent cannot locate it. Therefore, there are only two options for the second agent: (*O2W*): pick up one token and wait at the homebase until a token appears there, or (*02M*): pick up one token and leave (to a neighbour v_2 of the homebase).

Consider now what a fresh agent seeing no tokens can do: either leave the homebase towards its neighbour, or wait in the homebase until a token or two appear there. However, in the first case the adversary can place the black hole in that neighbour and wake-up the fresh agents one by one until all of them enter the black hole. Since $\Delta \geq 3$, the two surviving agents are insufficient to locate the black hole. Therefore, we may assume a fresh agent seeing no tokens at the homebase will wait.

Let us analyze the possible combinations for the actions of the first two agents:

Case *01M* and *O2M*: The adversary will place the black hole at v_2 and allow the first agent to travel among homebase and its neighbours, while the fresh agents are kept asleep. Eventually, the first agent will either (a) enter v_2, (b) leave a neighbour of the homebase via unexplored link, (c) start waiting at a neighbour of the homebase or (d) drop a token at homebase (and do something). In case (a) both tokens disappear at the black hole and the algorithm fails. In case (b) the adversary will direct that unexplored link to the black hole and the same happens. In case (c) all agents are either dead or waiting and the algorithm fails. Finally, in case (d), the adversary wakes-up the third agent, which will pick up the token and enter the black hole (the algorithm according to *O2M* case), again eliminating both tokens.

Case *O1W* and *O2M*: The adversary puts the black hole at v_2. Once the first agent notices that a token disappeared from the homebase, it will perform one of the actions (a), (b), (c) or (d) from the previous case and the adversary will deal with that in the same manner as before.

Case *O1M* and *O2W*: The adversary puts the black hole at v_1 and the first agent disappears there. Therefore all other agents will remain waiting at the homebase and the algorithm fails.

Case *O1W* and *O2W*: The adversary wakes-up the first agent while keeping the fresh agents asleep. Consider now the possible actions of the first agent once it notices that there are no tokens in the homebase. If the agent leaves the homebase without dropping its token, the adversary will direct it towards the black hole and everybody else will remain waiting. Therefore the first agent has to drop its token. Using asynchrony, the adversary will make sure that the second agent waiting for appearance of this token will not notice this. Instead, a fresh agent is woken-up, which (according to *O2W*) picks up the token and starts waiting. At this moment both tokens are effectively out of play as the agents holding them are waiting and the algorithm fails.

□

Finally we look at the number of moves necessary for black hole search in an unknown graph.

Theorem 3. *In an unknown graph, black hole search by $k = (2-\epsilon)\Delta$ agents costs $\Omega(\epsilon n^2/\Delta)$ moves in the worst-case.*

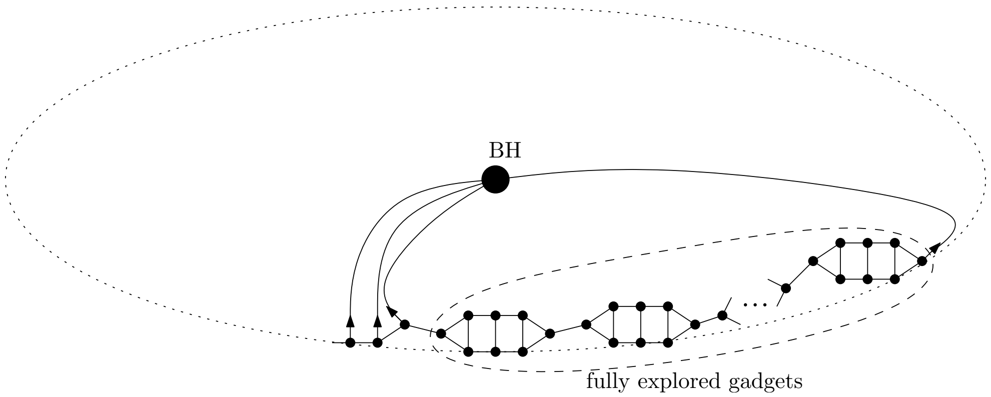

Figure 2: The graph for $\Delta = 4$ used for the lower bound on the number of moves. The arrows represent blocked agents while. A way how the black hole may be connected is shown as well.

Proof. In [17] it has been shown that with $\Delta+1$ agents, the worst-case complexity of BHS in the whiteboard model is $\Omega(n^2)$. The same proof with a bit more careful analysis yields the statement of the theorem.

The game between the adversary and the algorithm is played on a class $\mathcal{C}_\Delta$ of graphs based on a specific graph G_Δ consisting of $\Theta(n/\Delta)$ gadgets of size $\Theta(\Delta)$ connected by bridge edges into a ring – see Figure 2. The class $\mathcal{C}_\Delta$ consists of all the graphs in which the black hole is connected to Δ vertices of G_Δ so that $G_\Delta \backslash \{BH\}$ is connected. The gadgets are designed in such a way, that the adversary can prevent exploration progress by directing all the agents into the black hole, unless Δ agents cooperate in exploring a gadget. Once a gadget has been explored, the adversary blocks the subsequent bridge edge and unblocks the bridge edge at the opposite end of the explored area. Now the agents are allowed to explore the next gadget on that side. However, Δ of them are needed to make progress, which means that at least $2\Delta - k = \epsilon\Delta$ agents must cross the explored area after exploration of each gadget. Since there are $\Theta(n/\Delta)$ gadgets and the diameter of the explored area containing i gadgets is $\Theta(i\Delta)$, the $\Omega(\epsilon n^2/\Delta)$ lower bound follows. □

4. Possibility and Upper Bounds

4.1. Overview

The basic idea of Algorithm Black hole search is to have one designated agent (the leader) coordinating and directing the exploration activities by the other agents (the followers). The leader collects the map of the explored graph, identifies the black hole, and at the end instructs the surviving follower(s) to terminate. Here we assume that the number of agents is $k \geq \Delta + 2$ and it is known to them. The case where $k \geq \Delta + 2$ but is unknown is discussed at the end of this section.

The communication between the leader and the followers is performed using the communication module described in the following subsection. Algorithm Black hole search consists of three phases: *Leader Election and Initialization*, *Waiting Room Location and Synchronization*, and *Search*. The goal of the first phase is to elect a unique leader, and have all other agents become followers. In particular, this phase ensures that all agents join the computation and no agent remains sleeping. The purpose of this approach is to enable reuse of token configurations at the homebase.

As the exploration proceeds (and especially at the beginning), there might be the case that the exploration front (the number of unexplored edges leaving the already explored component) is smaller than the number of available agents. In order to avoid the unused agents cyclically asking the leader for work (causing potentially unbounded number of token transitions), the leader sends them to sleep and wakes them up once there is work for them. However, this sleeping cannot be done at the homebase, as the number of possible configurations is too low. Instead, a safe node (the *waiting room* (WR)) neighbouring the homebase is identified and used. The goal of the second phase is to identify this waiting room and ensure no interference with the third phase.

Once the waiting room is identified and necessary synchronization performed, the algorithm proceeds to the third phase in which the leader coordinates the search of the graph by the followers until the black hole is located. The protocol is designed in such a way that there will always be at most one follower communicating with the leader; all other followers are either searching, sleeping, or waiting for their turn to communicate.

4.2. Communication Using Tokens

At the core of our algorithm is a simple, novel technique that uses three tokens to communicate finite but arbitrary information (a sequence of bits) between two agents: A and B. All communication takes place at the homebase. Let (x, y, z) denote the configuration of tokens in which A has x tokens, B has z tokens, and the homebase contains y tokens. Let $\mathcal{S} = b_1 b_2 \cdots b_r$ be a sequence of r bits to be transmitted from B to A and let $\mathcal{S}' = b'_1 b'_2 \cdots b'_{r'}$ be a self-terminating sequence encoding $\mathcal{S}$ (in the protocol we use $\mathcal{S}' = b_1 1 b_2 \cdots 1 b_{r-1} 1 b_r 0$, i.e., 0 at an even position indicates the end of message).

Communication always starts from configuration $(1, 1, 1)$, with B's turn. B indicates the next communicated bit by either picking up or dropping a token. The subsequent moves transition the configuration back to the initial one, allowing communication of the next bit. The agents alternate their moves; due to asynchrony, an agent X knows that the other agent has observed X's action and performed its step only when observing a change of the configuration left by X. No other agents interfere with the communication, as all other agents present in

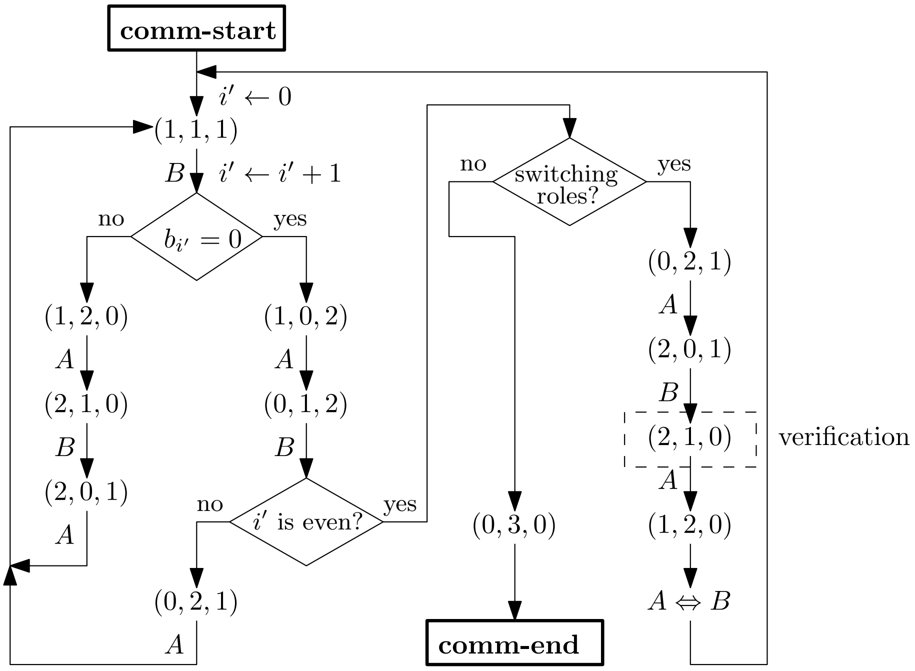

Figure 3: The communication subroutine. The agent performing the move is indicated to the left of the corresponding transition arrow. In the search phase, the leader perform the *verification* protocol before performing its action on the configuration marked *verification*.

the homebase wait until configuration $(0,3,0)$ appears – which happens only when the communication has been completed.

In the protocol, there is a provision for a two-way communication: After B has completed transmission of its message to A (indicated by 0 on an even position), it either indicates end of communication by setting a terminal configuration $(0,3,0)$, or signals (by configuration $(0,2,1)$) the need for switching roles between A and B. Consult Figure 3 for the full communication protocol.

4.3. Leader Election and Initialization

Figure 4 describes the token manipulation, the agent's state and the leader's variable transitions of the Algorithm Black hole search. As with the communication module, the state of the tokens is described as a triple (x,y,z), where x is the number of tokens held by the leader, y is the number of tokens placed in the homebase, and z is the number of tokens held by the follower communicating with the leader.

Initially, all agents are in initial state C (*candidate*). The first agent to wake-up observes three tokens in the homebase, picks up one token and becomes the *proto-leader* (PL). An agent in state C observing two tokens picks up a token to register with the leader and becomes a *proto-follower* (PF). After some token manipulation (see Figure 4, left) the configuration is set back to two tokens at the homebase and the next candidate can register. The leader counts the registered followers and, once $k-1$ of them have registered, it knows that all agents have woken-up, so it transitions to state *acknowledged leader* (AL) and drops the tokens it holds so that the homebase now contains 3 tokens. From now on, 3 tokens in the homebase means that the leader is ready for interaction with followers (for agents in state C it means that no leader has been elected yet; however, the initialization phase ensures that there are no agents in state C left).

4.4. Waiting Room Location and Synchronization

The goal of this phase is to identify a safe node adjacent to the homebase. If the homebase is of degree one, its only neighbour is the waiting room and, at the end of the first phase, the leader becomes the *ready leader* (RL) and the followers upon registration directly become *ready followers* (RF) instead of *proto-followers.*

Consider now the case that the homebase is of degree at least two. The second phase starts in configuration $(0, 3, 0)$, with the leader making the last turn. A follower in state PF observing 3 tokens takes two tokens to indicate that it wants to help. After some token manipulation (see Figure 4, the leftmost chain of the middle phase), the leader sends the first two such followers (now in states $PF1$ and $PF2$) to explore the first and second adjacent links. While these followers are exploring, other followers want to help too. The leader sends them to sleep in state *sleeping follower* (SF). As the waiting room has not been identified yet, these followers "sleep" at the homebase waiting until two tokens appear there.

Eventually, the follower sent to the first link and/or the one sent to the second link returns back to the homebase. If a follower returns from the first link, it waits until the leader is ready to communicate with it (i.e., when it sees 3 tokens in the homebase), grabs the three tokens and brings them to the node it went exploring. The leader knows that the only way all tokens can disappear is if the first follower found the first link safe, so it goes there and picks up the tokens (see Figure 4, middle top, where the fourth element in a tuple denotes the number of tokens in the node adjacent to the homebase over the first link). If the waiting room has not yet been identified, this node will be the waiting room.

If the second follower returns from the second link, it cannot communicate with the leader in the same way. Instead, it follows the protocol of a proto-follower trying to help. Eventually, (if the first follower meanwhile did not return) the leader will note that there were k proto-followers trying to help. Since there

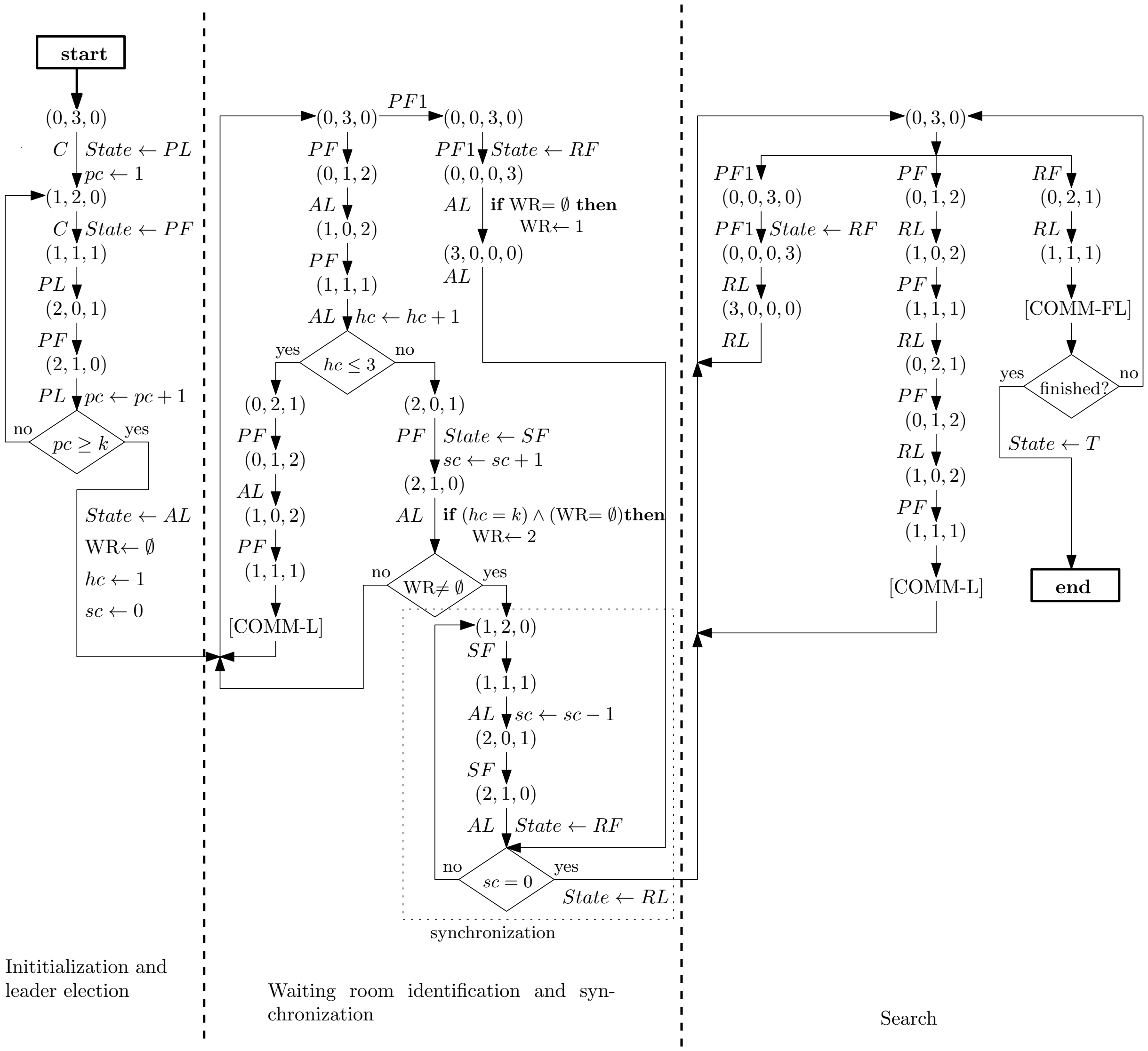

Figure 4: Token manipulation in Algorithm Black hole search. On the left of transition arrows is the state of the agent performing the transition. On the right are the actions performed by the agent. Note that the states are in fact groups of states, with sub-states associated with concrete locations in the diagram. COMM-L refers to a call to the communication subroutine in which the leader serves as agent B, without switching roles. In COMM-FL, first the follower acts as agent B, informing the leader; then the roles are switched and the leader tells the follower what to do next. The four-element configurations describe token manipulation involving the waiting room – the fourth element describes the content of the waiting room.

are altogether $k-1$ proto-followers, one must have done so twice, i.e. the second follower has returned and the second link leads to a safe node.

At the moment the waiting room is identified, the leader wants to become the *ready leader* (RL) and to transition to the main *search* phase. However, there may be sleeping followers waiting to be woken up (i.e., for two tokens in the homebase). Therefore, before becoming RL, the leader must wake up all the other agents into the *ready follower* (RF) state. This is achieved by putting two tokens in the homebase and subsequently manipulating the tokens as shown in the lower right loop of the middle phase in Figure 4. The leader maintains variables hc (counting the followers trying to help) and sc (the number of followers put to sleep) to help it perform these tasks.

Note that at the end of the second phase, the waiting room is identified, the leader is in state RL and there are no followers in state SF. However, there might be followers in state $PF1$, $PF2$, and PF. As we will see in the next subsection, this poses no problem for the algorithm.

4.5. Search

The search is coordinated by the leader, which creates, maintains, and updates a partial map of the explored component, until the location of the black hole is determined. As in the previous phase, configuration $(0, 3, 0)$ means that the leader is ready to interact with a follower.

Depending on what kind of follower starts the communication ($PF1$, PF, or RF) the leader responds accordingly (consult Figure 4, the rightmost phase). In the first two cases, the follower is told to become a *ready follower*. In the latter case, a two-way communication between the leader and follower is performed. The follower tells the leader the possible result of its exploration journey, if any. The leader responds with the next work order, with a command to go to wait to the waiting room, or by indicating termination if the black hole has already been located.

The work order is specified as a sequence of link labels identifying a path π to be explored, of which only the last edge is unexplored. A follower receiving such an instruction follows the path π and, if successful, returns back to the leader to report the degree of the last node v. The leader maintains the invariant that, at any moment, at most one agent is exploring a given unexplored link. This ensures that at most $\Delta = k-2$ agents enter the black hole, i.e., at least one follower survives.

When a follower reports that a path π leads to a safe node v, the leader must locate v in the partially constructed map. In particular, it must determine whether v is a new node, or an already visited node reached by the newly explored edge e at the end of π. To do so, the leader interrupts the communication during the switching roles phase at the moment when it holds two tokens and the homebase holds one token (in the node marked as *verification* in Figure 3) and determines the location of v as follows: It first traverses π until it comes to v, drops two tokens there[3], then returns to the already known part via e. The leader then visits all nodes explored so far, using the map it has constructed. If a node with two tokens is encountered, v has been visited before and only e is added to the map; otherwise, v is a new node and both e and v are added to the map. The leader then collects the tokens from v, brings them to the homebase, and completes the communication with the follower.

If the black hole has not yet been located, the leader either gives the follower a new work order, or sends it to wait in the waiting room if all currently unexplored edges are being explored by other followers. If new unexplored edges have appeared and there are waiting followers in the waiting room (the leader maintains a counter of them), the leader goes there and wakes one of them using the following protocol: When finishing the communication, instead of ending in state $(0, 3, 0)$, the leader brings the three tokens to the waiting room and drops them there. The first waiting follower to see three tokens picks them up and places two at the homebase[4]. The leader returns to the homebase, picks up one token and now the leader and freshly ready follower are ready to communicate.

Finally, if the black hole has been located and there are still ready or waiting followers, the leader wakes up the waiting followers (if needed), waits for the work requests of ready followers and tells them to terminate. Once there are no ready followers, the leader terminates.

4.6. Correctness and Complexity

One way to look at the Algorithm Black Hole Search is that each of the candidate/follower states has a "main-loop" sub-state in which the follower waits for the right to interact with the leader: C state waits for either 3 or 2 tokens in the homebase, SF state waits for 2 tokens in the homebase, WF state waits for 3 tokens in the waiting room, and all other follower states wait for 3 tokens in the homebase.

The following lemma captures this crucial, by-design property of the algorithm:

Lemma 4. *At any moment in time (after the leader has been selected), there is at most one follower interacting with the leader, all the other followers are either exploring, or waiting in their respective main-loop sub-states and do not manipulate the tokens. Furthermore:*

[3] It can happen that v is the homebase, this is easily identified by finding one token there; in which case, the leader does not drop any tokens.

[4] This might look like a signal for yet another switching of roles with the follower with which the leader previously communicated. However, from the context, that follower knows that no further role switching is needed and terminates its interaction with the leader.

(1) Eventually, the first phase is completed by an agent reaching state AL. At that moment, there are no agents left in state C and all agents are in state PF.

(2) Eventually, the second phase is completed by an agent reaching state RL. At that moment, a safe waiting room is correctly identified and there are no agents in state SF.

Proof. We prove the lemma statement directly for the first phase; we need (1) and (2), respectively, to show that the lemma is true for the second and third phase as well.

In the first phase, an agent A in state C starts interaction with the leader only when seeing two tokens in the homebase. At that moment, A removes a token from there, preventing another agent from entering the interaction. At the moment two tokens are restored in the homebase, A is already in state PF, waiting for appearance of three tokens in the homebase.

(1) now follows from construction: The first agent to wake-up sees three tokens, takes one of them and becomes the leader. As long as there are agents in state C, the leader's counter pc is less than k and the leader cannot progress to the second phase. Eventually, each agent in state C will be awaken and will transition to state PF; when the last one does so, the leader will notice and enter the AL state. As the only way a leader can be created is by an agent C observing three tokens, this implies there will be exactly one leader.

The followers in state $PF*$ (i.e. PF, $PF1$ or $PF2$) in the second stage enter interaction by observing three tokens and removing two of them, again preventing other followers from entering interaction. Since three tokens in the homebase do not appear during the communication subroutine, no other such follower can enter interaction before the previous interaction has run its course. Note that a follower in state SF can do so in principle, as it is waiting for only two tokens. However, the first follower in such a state is created only after the communication with the first two helping followers has completed. Since these first two communications are the only calls to the communication subroutine during the second phase, the statement holds also during the second phase.

(2) now follows from construction: Eventually, either the agent exploring the first link or the one exploring the second link returns to the homebase. If the first agent returns first, or if it returns before the last proto-follower registers trying to help, the transition sequence corresponding to state $PF1$ is followed, and the waiting room is set to 1. If the second agent returns and all proto-followers register before the first agent returns, the waiting room is set to 2. In both cases the leader proceeds to release all followers sleeping in state SF into state RF and only after doing so changes its state to RL, completing the second phase.

As there are no agents in state C or SF in the third phase, all followers wait for three tokens to start interaction with the leader. The sequentiality of this interaction then follows from previously used arguments (an agent starting interaction removes a token from the homebase, three tokens are restored only when the interaction has concluded, no three tokens at the homebase are present during the communication subroutine). Note that the same arguments imply sequentiality of interaction in the waiting room as well.

Finally, there is no unwanted interference (an agent picking up tokens meant for another agent) due to verification phase: Such an interaction can appear only in a node in which there are agents waiting for a change in token configuration, i.e., if the node at the end of the newly explored edge is either the homebase, or the waiting room. In the first case, the leader detects this by noticing a token at the homebase and does not drop the two tokens. In the second case, two dropped tokens do not disrupt the waiting agents, as those agents wait for three tokens. □

Lemma 4 means that there is no unwanted/unexpected interference between the agents, i.e., all token transitions are according to the diagrams from Figure 3 and Figure 4.

Now we are ready for the main theorem:

Theorem 4. *Algorithm* Black Hole Search *solves* Bhs *using* $\Delta + 2$ *agents and* 3 *tokens, at the cost of* $O(mn)$ *agent moves and* $O((m + \Delta n)\log n)$ *token transitions.*

Proof. **Correctness:** By (2) of Lemma 4, the algorithm eventually reaches the third, exploration phase. By construction of the search phase, no two agents are sent to explore the same unexplored edge. Therefore, as there are $k - 1 = \Delta + 1$ followers, at least one of them survives. As the communication between the leader and the followers works as intended (Lemma 4), eventually all links not leading to the black hole will be explored. As both n and m are known, at that moment the leader identifies the location of the black hole and Bhs is solved.

Cost: Each edge is explored at most twice (at most once from each direction). The cost of exploring an edge is bound by $O(n)$: $O(n)$ moves for the follower to reach it and return back, $O(n)$ moves for the leader to verify whether it is a new vertex or not (e.g. using a spanning tree of the currently explored component). The possible cost $O(1)$ of waking-up a follower waiting at the waiting room can be charged to the edge it has been woken-up to explore (into this we can hide also the cost of sending it to the waiting room before). Summing over all edges yields $O(nm)$ bound on the total number of moves.

Let us now estimate the total number of token transitions. According to Figure 4, each follower performs $O(1)$ transitions until it reaches state RL. Subsequently, additional $O(1)$ transitions per explored edge are spent outside the communication subroutine. As the transitions of followers and the leader alternate, the overall number of token transitions outside the communication subroutine can be bound by $O(\Delta + m)$.

Let us now bound the overall number of token transitions during the communication subroutine. There, $O(b)$ token transitions are used in order to communicate b bits. Note that each call to the communication subroutine can be attributed either to a new edge being explored (if the call contains the work order for this edge), or to putting an agent to sleep. As the cost of putting an agent to sleep is $O(1)$ and can be charged to its next work order, it is sufficient to bound the total bit length of all work orders.

Directly encoding the exploration path π of each work order would result in overall $O(mn \log n)$ communicated bits. However, a more efficient scheme can be used: Each follower builds its own knowledge of safe edges, based on where the leader sends it. The leader remembers this information for each follower. When the leader gives a follower A a new edge e to explore, instead of specifying the full path π, it specifies the closest (over safe edges known to the leader) vertex u which is known to be safe to A (e.g. by telling A: "take i-th vertex you learned to be safe") and from there specifies a short(-ened) path π' by which to reach e from u over links known to be safe to the leader, but not to A.

Note that π' never traverses a vertex u' known to be safe to A (otherwise u' would have been used as the starting point of π' instead of u). Hence, $\bigcup \pi''$ is a tree, where π'' is obtained from π' by omitting the last edge and the union is taken over all π' told to A by the leader. Therefore, the total number of communicated bits to A can be bound by $(n+m_A) \log n$, where m_A is the number of edges explored by A. Summing over all agents yields $O((m + \Delta n) \log n)$ for the overall number of token transitions during the communication subroutine. □

4.7. The case of k being unknown

The algorithm Black Hole Search heavily relies on the fact that the number of agents is known beforehand, in order to ensure that the number of different token configurations used in the homebase is minimized. If the number of agents is sufficient (i.e., $k \geq \Delta + 2$) but unknown, there will always be a possibility that in the middle of the execution, a new agent will wake-up and try to get involved. This situation causes an increase in the number of possible configurations at the homebase, and hence the total number of tokens used.

A much simpler approach can be employed using two more tokens: Five tokens in the home base means there is no leader elected—the first agent to wake-up takes two tokens and becomes a leader. Three tokens means the leader is ready for communication to receive reports and assign new work, while four tokens means the leader is available to receive reports, but there is no new work to be assigned. Zero to two tokens means the leader is interacting with another follower, do not disturb them. As a consequence, there is no need for distinct phases, nor for a separate waiting room: Any follower, either freshly awaken, returning from work assigning, or asleep waiting for work can immediately decide upon its action.

It remains an open question whether the Bhs can be solved using four tokens in the case that the number of agents is unknown.

5. Conclusions

In this paper we have established the first result on the system space needed for black hole search without a map. We have proven that, with a constant number of tokens, $\Delta + 1$ agents are not sufficient to solve the problem. In conjunction with the result of [20], this result means that the minimal team size achievable with an unbounded number of tokens, can also be achieved with a constant number of tokens if and only if the agents have a map.

Furthermore, we have shown that, regardless of the team size, two tokens are *not* sufficient. In other words, any solution employing a bounded number of tokens, must consist of at least $\Delta+2$ agents and use at least 3 tokens. Hence, our protocol is optimal.

An immediate open question is the following: since a constant number of tokens is not sufficient to locate the black hole with $\Delta + 1$ agents, how many tokens are needed for such a team size to solve the problem? Recall that the current solution with $\Delta + 1$ agents employs Δ^2 tokens [16].

6. Acknowledgments

The authors would like to thank the anonymous referees for their comments that helped improving the presentation. This work has been supported in part by NSERC Discovery Grants, VEGA, and by Professor Flocchini's University Research Chair.

[1] S. Albers and M.R. Henzinger. Exploring unknown environments. *SIAM J. Comp.*, 29:1164–1188, 2000.

[2] B. Balamohan, P. Flocchini, A. Miri, and N. Santoro. Improving the optimal bounds for black hole search in rings. *Discrete Mathematics, Algorithms and Applications*, 3(4):457–472, 2011.

[3] M. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble. *Information and Computation*, 176(1):1–21, 2002.

[4] J. Cao and J.K. Das (Eds). *Mobile Agents in Networking and Distributed Computing.* John Wiley & Sons, 2012.

[5] J. Chalopin, S. Das, A. Labourel, and E. Markou. Black hole search with finite automata scattered in a synchronous torus. In *25th International Conference on Distributed Computing (DISC)*, pages 432–446, 2011.

[6] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for black hole search with scattered agents in synchronous rings. *Theoretical Computer Science*, 509:70–85, 2013.

[7] C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *10th International Conference on Principles of Distributed Systems (OPODIS)*, pages 320–332, 2006.

[8] 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, 2011.

[9] J. Czyzowicz, S. Dobrev, R. Královic, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In *16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 182–194, 2009.

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

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

[12] 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.

[13] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.

[14] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. *Journal of Algorithms*, 51(1):38–63, 2004.

[15] S. Dobrev, P. Flocchini, R. Královic, P. Ruzicka, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[16] S. Dobrev, P. Flocchini, R. Královic, and N. Santoro. Exploring an unknown dangerous graph using tokens. *Theoretical Computer Science*, 472:28–45, 2013.

[17] 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.

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

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

[20] 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.

[21] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *24th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–10, 2009.

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

[23] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(23):166–177, 2006.

[24] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2–3):331–344, 2005.

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

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

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

[28] A. Kosowski, A. Navarra, and C. M. Pinotti. Synchronous black hole search in directed graphs. *Theoretical Computer Science*, 412(41):5752–5759, 2011.

[29] R. Královic and S. Miklík. Periodic data retrieval problem in rings containing a malicious host. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 157–167, 2010.

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

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

[32] W. Shi, J. Garca-Alfaro, and J.-P. Corriveau. Searching for a black hole in interconnected networks using mobile agents and tokens. *Journal of Parallel and Distributed Computing*, 74(1):1945–1958, 2014.