Black Hole Search and Exploration in Unoriented Tori with Synchronous Scattered Finite Automata

Euripides Markou[1,⋆] and Michel Paquette[2,⋆⋆]

[1] Department of Computer Science and Biomedical Informatics, University of Central Greece, Lamia, Greece.
emarkou@ucg.gr
[2] Department of Computer Science, Vanier College, Montréal, Canada.
michel.paquette@vaniercollege.qc.ca

Abstract. We consider the problem of locating a black hole in a synchronous, anonymous, and unoriented torus network using mobile agents. A black hole is a harmful network node that destroys any agent visiting it without leaving any trace. The objective is to locate the black hole using as few agents as possible. We present here an almost optimal deterministic algorithm for synchronous (partially) unoriented tori using five scattered agents with constant memory and three identical tokens. We also study the exploration problem of a safe (i.e., without black holes) unoriented torus. While it has been previously shown that there is no universal algorithm for one agent with constant memory and any constant number of tokens which can explore all cubic planar graphs, we give here the first algorithm which enables a finite automaton with two tokens to explore (without termination detection) any totally unoriented torus and we prove optimality on the number of tokens.

Keywords: Distributed Algorithms, Fault Tolerance, Black Hole Search, Anonymous Networks, Mobile Agents, Finite State Automata

1 Introduction

The exploration of an unknown graph by one or more mobile agents is a classical problem initially formulated in 1951 by Shannon [23] and it has been extensively studied since then (e.g., see [2, 9, 16]). In 1967, during a talk at Berkeley, Rabin [21] conjectured that no finite automaton with a constant number of pebbles (or tokens) can explore all graphs (a pebble is a marker that can be dropped at

⋆ E. Markou has been co-financed by the European Union (European Social Fund-ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF)-Research Funding Program: THALIS-UOA-GeomComp.

⋆⋆ Part of this work was done during a visit from this author at the University of Central Greece, financed by grants from the University of Central Greece and Cégep International (Programme de soutien à la mobilité enseignante.)

and removed from nodes). The first step towards a formal proof of Rabin's conjecture is generally attributed to Budach [4], for an agent without pebbles. Blum and Kozen [3] improved Budach's result by proving that *three* agents cannot cooperatively perform exploration of all graphs. In 1979, Kozen [19] proved that *four* cooperative agents cannot explore all graphs. Finally, in 1980, Rollik [22] gave a complete proof of Rabin's conjecture. More precisely, Rollik proved that no finite set of finite automata can cooperatively perform exploration of all cubic planar graphs. Since a finite automaton is more powerful than a pebble (a pebble does not have states, or a transition function), Rabin's conjecture is a corollary of Rollik's theorem.

Recently, the exploration problem has also been studied in unsafe networks which contain malicious hosts of a highly harmful nature, called *black holes*. A black hole is a node which contains a stationary process destroying all mobile agents visiting this node, without leaving any trace. In the *Black Hole Search (BHS)* problem the goal for the agents is to locate the black hole within finite time. In particular, at least one agent has to survive knowing all edges leading to the black hole. Without the knowledge of the size of the network, the only way of locating a black hole is to have at least one agent visiting it. However, since any agent visiting a black hole vanishes without leaving any trace, the location of the black hole must be deducted by some communication mechanism employed by the agents. Four such mechanisms have been proposed in the literature: a) the *whiteboard* model in which there is a whiteboard at each node of the network on which the agents can leave messages, b) the *pure token* model where the agents carry tokens which they can leave at nodes, c) the *enhanced token* model in which the agents can leave tokens at nodes or edges, and d) the *time-out mechanism* (only for synchronous networks) in which at least two agents gather at a node u, and then one agent explores a new node and returns to u to inform the other agents who wait.

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

In asynchronous networks and given that all agents initially start at the same safe node, the Black Hole Search problem has been studied under the whiteboard model (e.g., [11, 12]), the *enhanced token* model (e.g., [10]) and the *pure token* model (e.g., [1, 14]). In these models it has been proven that the problem can be solved with a minimal number of agents performing a polynomial number of moves. It has been also shown that in an asynchronous network the number of the nodes in the network must be known to the agents otherwise the problem is unsolvable ([12]). If the graph topology is unknown, at least $\Delta + 1$ agents are needed, where Δ is the maximum node degree in the graph ([11]). Furthermore the network should be 2-connected. It is also not possible to answer the question

of *whether* one black hole exists in the asynchronous network. With scattered agents (not initially located at the same node) in asynchronous networks, the problem has been investigated for the ring topology ([12]) and for arbitrary topologies ([15]) in the whiteboard model while in the *enhanced token* model it has been studied for rings ([13]) and for some interconnected networks ([24]).

The situation in synchronous networks is dramatically different. Under this assumption, two co-located distinct agents can discover one black hole in any graph (provided that the graph can be explored) by using the time-out mechanism, without the need of whiteboards or tokens. Moreover the network does not have to be 2-connected anymore, as in asynchronous networks, and furthermore it is now possible to answer the question of whether a black hole actually exists or not in the network. No knowledge about the number of nodes is needed. Hence, with co-located distinct agents, the issue is not the feasibility but the time efficiency of black hole search. The issue of efficient black hole search has been studied in synchronous networks without whiteboards or tokens, only using the time-out mechanism (e.g., [7, 8, 17, 18]) under the condition that all distinct agents start at the same node. However when the agents are scattered in the network, the time-out mechanism is not sufficient anymore: the agents have to gather at the same node in order to use the time-out mechanism.

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

In all previous papers studying the Black Hole Search problem under a token model apart from [1, 6, 5, 14], the authors have used the *enhanced* token model with agents having non-constant memory. The weakest *pure* token model has been used in [1, 14] for co-located agents with non-constant memory in asynchronous networks. The first results for scattered agents with constant memory and *pure* tokens appeared in [6] for synchronous unoriented rings and [5] for synchronous oriented tori. In [5] it has been proven that three scattered agents with constant memory and with two tokens each can locate the black hole in any synchronous oriented torus.

2 Our Results

We study the Black Hole Search problem (BHS) for scattered identical anonymous agents with constant memory in synchronous anonymous *unoriented* torus networks under the *pure token* model. We use the same model as in [6, 5] but

we focus on *unoriented* torus topologies. Throughout the paper we discuss four cases of unoriented tori:

- *type 0*: the agents do not agree on anything regarding the orientation;
- *type 1*: the agents perceive orthogonal axes but they do not agree on which axis is horizontal and which is vertical;
- *type 2*: the agents agree on which axis is horizontal and which is vertical, but there is no consensus on the orientation of each axis;
- *type 3*: the agents agree on which axis is horizontal and which is vertical and they also agree on the orientation in one of the axes.

For the BHS problem we show the results presented in Table 1:

Torus orientation	**# agents**	**# tokens**	**Black Hole Search**
Type 0	any constant	1	Impossible
	4	any constant	
Type 1 or 2	4	any constant	Impossible
	5	1	
	5	3	Algorithm $UBHS_{5,3}$
Type 3	3	any constant	Impossible
	4	1	
	5	3	Algorithm $UBHS_{5,3}$

Table 1. Summary of results for BHS in synchronous unoriented tori

We also show the following results for the exploration problem of a safe unoriented torus:

- There is no universal algorithm for any constant number of agents with one movable token solving the exploration problem (even without termination detection) in all unoriented tori.
- There is a universal algorithm for the exploration (without termination detection) of any unoriented torus (type 0) by one agent with constant memory and two movable tokens.

This last result is optimal on the number of tokens and it is somewhat surprising since it had been proven, in [22], that an agent with any constant number of tokens cannot explore all cubic planar graphs. Although it has been proved in [3] that exploration can be done with a constant number of tokens in partial grids (grids with missing nodes and edges) with *sense of direction*, our result shows that the impossibility result of [22] is not robust enough to resist in highly structured graphs, like tori, without *sense of direction*.

Due to space limitations some proofs have been omitted and will appear in the full version of the paper.

3 Our Model

Our model consists of $k \geq 2$ anonymous and identical mobile agents that are initially placed at distinct nodes of an anonymous, synchronous torus network. Each agent consistently evaluates direction across the torus (except in type 0 tori); agents do not necessarily evaluate the same directions for West, East, North or South. In *type* 0 tori, we make no assumption on the way each direction is interpreted. For *type* 1*,* 2*, and* 3 tori, however, we assume that each such function evaluates West and East to be opposites on the same axis and orthogonal to North and South (also opposites on the same axis). Each mobile agent has a constant number t of identical pure tokens which can be placed at any node visited by the agent. We call a token *movable* if it can be moved by any mobile agent to any node of the network, otherwise we call the token *unmovable* in the sense that, once released, it can occupy only the node in which it has been released. A token at a given node is visible to all agents on the same node, but is not visible to agents at other locations. While our negative results hold even when the agents have the capability to communicate when they are at the same node (which is the usual assumption in previous works), our protocol works even when the agents cannot directly communicate at all, regardless of their location. The agents follow the same deterministic algorithm and begin execution at the same time, at the same initial state. At any single time unit, a mobile agent occupies a node of the network and may 1) stay there or move to an adjacent node, 2) detect the presence of one or more tokens at the node it is occupying and 3) release/take one or more tokens to/from the node it is occupying. When two or more agents located at the same node attempt to see and/or change the node configuration (release or take tokens) at the same time, they do it by mutual exclusion. We give more details on the mutual exclusion mechanism, later in this section.

Formally we consider a mobile agent as a finite Moore automaton $\mathcal{A} = (\mathcal{S}, S_0, \Sigma, \Lambda, \delta, \phi)$, where $\mathcal{S}$ is a set of $\sigma \geq 2$ states; S_0 is the *initial* state; Σ is the set of possible configurations an agent can see when it enters a node; $\delta : \mathcal{S} \times \Sigma \to \mathcal{S}$ is the transition function; and $\phi : \mathcal{S} \to \Lambda$ is the output function. Elements of Σ are triplets (D, x, y) where $D \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}\}$ is the direction through which the agent has arrived at the node, x is the number of tokens at that node, and y is the number of tokens carried by the agent. Elements of Λ are triplets (A, s, M) where $A \in \{\texttt{Put}, \texttt{Take}\}$ is the action performed by the agent on the tokens, s is the number of tokens concerned by the action A, and $M \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{wait}\}$ is the move performed by the agent.

When two or more agents are on the same node and wish to operate at the same time then the mutual exclusion mechanism guarantees that the agents one by one evaluate functions δ, ϕ. The sequence δ, ϕ is atomic. The order by which the agents evaluate their functions is handled by an adversary.

We assume that the memory of an agent is proportional to the number of bits required to encode its states which we take to be $\Theta(\log(|\mathcal{S}|))$ bits. Note that in our algorithms all computations by the agents are independent of the size $n \times m$ of the torus network. The agents have no knowledge of n, m. There is exactly

one black hole in the network. An agent can start from any node other than the black hole and no two agents are initially co-located (we say that they are *scattered*). The agents' initial locations and the black hole location are decided by an adversary. Once an agent detects a link to the black hole, it marks the link permanently as dangerous (i.e., disables this link). Since the agents do not have enough memory to remember the location of the black hole, we require that at the end of a black hole search scheme, all links incident to the black hole (and only those links) are marked dangerous and that there is at least one surviving agent. Each agent has the following primitives:

1. Walk(x): move to an adjacent node in direction x or through port x and return the incoming port label.
2. Opposite(dir): return the direction opposite to dir.
3. Put(t): leave t tokens at the current node.
4. Read(): return the number of tokens at the current node.
5. Take(t): remove t tokens from the current node.

The mutual exclusion mechanism guarantees that two or more co-located agents execute one by one a sequence containing *Read, Put, Take* actions.

4 Negative Results

In any correct algorithm for solving the Black Hole Search problem each node apart from at most one of the network must be visited by at least one agent in the worst case, since if there are two or more unvisited nodes, the agents cannot decide which one is the black hole.

In [5] it has been proven that no finite team of agents can solve the BHS problem in all oriented torus networks using any constant number of *unmovable* tokens:

Theorem 1. *[5] For any constant numbers k, t, there exists no algorithm that solves BHS in all oriented tori containing one black hole and k scattered agents, where each agent has a constant memory and t unmovable tokens.*

Hence, as in the case of an oriented torus, a correct algorithm for the BHS problem in an unoriented torus should use movable tokens.

4.1 Black Hole Search in a Torus of Type 3

Lemma 1. *There is no universal algorithm solving the BHS problem in all synchronous semi-oriented tori of type* 3*, using less than* 4 *scattered agents carrying any constant number of tokens even when the agents have unlimited memory.*

Proof. To locate a black hole, any BHS algorithm functioning with scattered agents must move all agents by at least one node and must also force agents to traverse multiple rings of the torus. Consider such a BHS algorithm. Suppose (without loss of generality) that the agents agree on the horizontal orientation

of the torus. Then an adversary can choose agents' orientation on the vertical direction and initial locations so that one agent will vanish into the black hole while traveling horizontally and two more agents will fall into the black hole while traveling vertically (without having met any tokens other than their own tokens). Hence, a fourth agent is needed to compute where the black hole is located. ■

In [5] it has been proven that three scattered agents carrying one token each cannot solve the BHS problem in a synchronous *oriented* torus. The basic argument in that proof is the following: either the agents stay 'close' to their tokens and in that case they fail to explore the whole torus, or they go far from their tokens (more than a constant number of steps), they manage to explore the torus and meet the black hole but fail to leave a clear indication at nearby nodes for the remaining agents. We argue similarly here and prove the following lemma.

Lemma 2. *There is no universal algorithm which solves the BHS problem in all synchronous semi-oriented tori of type* 3*, using less than* 5 *scattered agents with one movable token each if the agents have constant memory.*

Proof.(Sketch) Since in view of Theorem 1, solving the BHS problem with unmovable tokens is impossible, a correct BHS algorithm should eventually instruct the agents to leave their tokens down. This decision should be taken after a constant number of steps (independent of the size of the torus) due to the constant number of agents' states. Moreover this decision has to be taken at the same time for all agents since the agents are anonymous and start at the same state.

If agents always move a constant number of steps away from their tokens, then an adversary can always select the size of the torus, the initial locations of the agents and the black hole location, so that any agent will never meet a token of another agent, another agent, or the black hole. Moreover the agents will be eventually trapped visiting the same nodes and there will be nodes in the torus which remain unvisited by any agent. Therefore in a correct algorithm the agents should move more than a constant number of steps away from their tokens. Suppose without loss of generality that the agents disagree on the horizontal orientation. First consider the case in which the agents move more than a constant number of nodes away from their own tokens in the horizontal axis. Due to disagreement in the horizontal orientation, the adversary can force two agents to vanish at the black hole at the same time leaving their tokens more than a constant number of steps away from the black hole. The adversary may also arrange that a third agent enters the black hole without having met the others' tokens and leaving its token somewhere more than a constant distance away from the black hole. The remaining case in which the agents move more than a constant number of nodes away from their own tokens in the vertical axis can be argued analogously. Hence in both cases a fourth agent may now find a token other than its own token but cannot decide the black hole location. ■

The above negative results also hold for tori of type 2, 1 or 0.

4.2 Black Hole Search in a Torus of Type 2

When the agents agree only on which axis of the torus is horizontal and which is vertical but they do not agree on their orientations (torus of type 2), then analogously as in Lemma 1 we can show that:

Lemma 3. *There is no universal algorithm solving the BHS problem in all synchronous semi-oriented tori of type* 2*, using less than* 5 *scattered agents carrying any constant number of tokens even when the agents have unlimited memory.*

Proof. An adversary can choose agents' orientation and initial locations such that two agents will enter the black hole while traveling horizontally and another two agents will vanish into the black hole while traveling vertically (without having met tokens other than their own tokens). Then, a fifth agent is needed to compute where the black hole is located. ■

With a similar reasoning as in the proof of Lemma 2 the following lemma holds:

Lemma 4. *There is no universal algorithm which solves the BHS problem in all synchronous semi-oriented tori of type* 2*, using less than* 6 *scattered agents with one movable token each if the agents have constant memory.*

The above negative results also hold for tori of type 1 or 0.

4.3 Black Hole Search and Exploration in a Torus of Type 0

In a type 0 torus the agents do not agree on anything related to the orientation of the torus. We however assume a local port labeling (i.e., port labels of incident edges of a node are different), otherwise an agent is not capable even to visit all neighbors of its current node. This port labeling is fixed by an adversary and is not globally consistent (i.e., an agent which always exit nodes by port *East* does not necessarily traverses a complete ring of the torus). We note that once a port label is fixed by the adversary, it cannot be changed (i.e., the adversary cannot change previously fixed port labels).

We show below (Theorem 2) that any constant number of scattered agents with constant memory and one token cannot solve the BHS problem in all tori of type 0. The idea of the proof is that the agents with only one token are not able even to explore a safe torus (Lemma 6), leaving many nodes unvisited (in contrast with semi-oriented safe tori, where only one agent with constant memory and one token can visit all nodes of the torus). Hence an adversary can place the black hole in one of the unvisited nodes and the agents are not able to decide its location. We first prove the following lemma:

Lemma 5. *There is no universal algorithm which solves the exploration problem in all synchronous unoriented (safe) tori of type* 0 *using one agent with constant memory and one movable token.*

Proof.(Sketch) Consider such an exploration algorithm for the sake of contradiction and let $\sigma \geq 2$ be the number of states of the agent. An adversary assigns port labels so that the agent can complete its first $2\sigma - 1$ moves (i.e., in which encounters 2σ states). The labels are assigned to ports so that for each edge the pair of the two port labels is either (*East, West*) or (*North, South*).

Suppose the algorithm instructs the agent not to release its token during this sequence p of 2σ states. Due to the σ number of different states, there is at least one state which has been repeated in p. Consider the first state S^* which is repeated and also has the following property: assuming u_1, u_2 to be the nodes where the agent is located when it encounters state S^* at times t_1, t_2 respectively (where $t_1 < t_2$), the entry port label to u_2 is different than the exit port label from u_1. If there is no such state in p, this means that there is a subsequence of p starting and ending at a state S' with the agent locating at the same node (basically in this subsequence the agent moves back and forth traversing the same edge). Suppose that such a state S^* exists. If $u_1 \equiv u_2$, then, since the whole sequence of states and moves will be repeated, the agent visits again and again the same nodes. Suppose that $u_1 \neq u_2$. Then the adversary can arrange the port labels so that nodes u_1, u_2 are on the same horizontal ring. The sequence of moves and states is repeated and if we call u_i the node at which the agent encounters state S^* for the $i-$th time, then the adversary can arrange the port labels so that nodes u_2, u_3 are on the same vertical ring (notice that the distance $d(u_2, u_3)$ will be equal to $d(u_1, u_2)$), nodes u_3, u_4 are on the same horizontal ring, u_4 is at the same vertical ring with u_1 (since $d(u_3, u_4)$ will be equal to $d(u_1, u_2)$), and finally nodes u_4, u_5 are on the same vertical ring and $u_5 \equiv u_1$ (since $d(u_4, u_5)$ will be equal to $d(u_2, u_3)$). The agent has visited a total number of at most $8\sigma - 1$ different nodes and then it keeps visiting the same nodes.

Suppose that the algorithm instructs the agent to release its token within the first $2\sigma - 1$ first moves. In fact this has to be done within the first σ moves otherwise it will never be done. The agent continues moving without a token. If the adversary can assign the port labels so that after another at most 8σ moves the agent does not meet its token, then similarly as above, the agent passes twice from the same node being at the same state and then keeps visiting the same nodes. If the agent meets its token within 8σ moves then after at most $\sigma+1$ times meeting its token will again meet its token being at the same state S_t. Between those two repetitions of state S_t the agent has visited at most $c\sigma^2$ nodes, where c is a constant. After that the agent repeats this orbit but maybe on a different area of the torus. However in an e.g., $n \times n$ torus, after at most n repetitions of this orbit the agent will be at the same state and at the same node (meeting its token) and therefore after that the agent will repeat everything visiting exactly the same nodes. The agent has been visited at most $O(n)$ nodes out of the n^2 nodes of a $n \times n$ torus. ■

Generalizing the previous lemma, it can be shown that:

Lemma 6. *There is no universal algorithm which solves the exploration problem in all synchronous (safe) unoriented tori of type* 0 *using any constant number* k *of scattered agents with constant memory and one movable token each.*

Hence, when there is a black hole in the torus network the adversary can place it in one of the unvisited nodes and therefore the following theorem holds:

Theorem 2. *There is no universal algorithm which solves the BHS problem in all synchronous unoriented tori of type* 0 *using any constant number of scattered agents with one movable token each if the agents have constant memory.*

As we will discuss in the next section, an agent with constant memory and only two tokens can explore (without stop) any unoriented torus of type 0.

In view of Lemma 3 and Theorem 2, any BHS algorithm for any type 0 unoriented torus would need at least five agents with two tokens each.

5 Positive Results

5.1 Exploration of an unoriented torus by a finite automaton

In the previous section we proved that any constant number of finite automata with one token each cannot solve the black hole search problem in all unoriented tori. The proof relies on Lemma 6 stating that there is no universal algorithm for any team of a constant number of finite automata with one token each that can *explore* all (safe) unoriented tori (type 0). This result is in agreement with the well known result of Rollik which says that an agent with any constant number of tokens cannot explore all cubic planar graphs ([22]). Hence a natural question is whether an agent with two tokens would be able to explore all unoriented tori. The answer is surprisingly positive. As we present in this section (Algorithm `Explore2Tokens`), one agent with constant memory having only two movable tokens can explore (perpetual exploration without stop) any type 0 unoriented torus. This result shows that although a torus is already a non-planar graph with degree 4, its special properties can be exploited by a not very complicated algorithm which solves the exploration problem in such a weak model. The basic idea of the algorithm is hidden in `Function ExploreRing` which enables the agent to explore a whole ring of the torus, when the torus consists of rings of at least 4 nodes[3]. The idea[4] can be described as follows: The agent located at a node u leaves a token down and selects a port leaving u and entering an adjacent node v. Now in order to discover which node adjacent to v and different than u lies in the same (horizontal or vertical) ring of the torus with v and u the agent tests all adjacent nodes of v (apart from u) to find a node w for which all paths

[3] Small tori containing rings of 2 or 3 nodes can be easily explored by traversing all paths of length 2 or 3 as shown in Procedure `SmallRing`.

[4] As it has been brought to our attention by the anonymous referees, the techniques for local orientation of tori we use in the algorithm have some similarities with the techniques presented in [20] for solving the *Leader Election* problem in unlabeled tori using messages.

Procedure SmallRing

```
1: //Take care the case of a torus with a ring of less than 4 nodes
2: Put(2)
3: for i ← 2 to 3 do
4:    Explore all paths of length i and return
5:    if a token is found in more than 2 different paths then
6:       Stop //the torus is i × i and has been already explored
7:    if a token is found in exactly 2 paths starting at ports p1, p2 then
8:       //the torus is i × n
9:       loop
10:         Move the tokens taking a port p ≠ p1, p2
11:         Explore all paths of length i and return
12:         Let p1, p2 be the starting ports of paths with tokens
13: Take(2)
```

of length 2 starting at w and not passing from v do not end up at a node with a token (i.e., node u). Then node u, v, w belong in the same ring of the torus. By repeating this procedure, the agent can explore a ring. The exploration of a ring finishes when the agent finds its (homebase) token which had been left at the starting node. Then it can move tokens in the next ring and continue. In that way the agent will eventually explore (without stop) the whole torus.

Theorem 3. *Algorithm* `Explore2Tokens` *enables one agent with constant memory and two movable tokens to explore (without stop) any unoriented torus.*

In view of Lemma 6, Algorithm `Explore2Tokens` is optimal with respect to the number of agents and tokens it uses. The algorithm can be extended for solving exploration *with* stop using three movable tokens.

5.2 BHS in semi-oriented tori of type 1

In this section, we state Algorithm $UBHS_{5,3}$ which enables 5 scattered agents with constant memory and 3 tokens each to locate the black hole in any torus of type 1 (i.e., a torus in which the agents agree only on the orthogonal axes). Intuitively the algorithm works as follows: Each agent leaves 2 of its tokens at its starting node and explores its starting horizontal (perceived) ring using the Procedure `CautiousTest` with its last remaining token (leaves a token, walks to a neighbor at a given direction and returns to pick up its token). Then moves the 2 (homebase) tokens onto the next horizontal ring and repeats the procedure. Eventually, there will be at least one and at most four agents entering the black hole and leaving at an adjacent node 1 or 3 tokens. At least one of the remaining agents will find such a configuration of 1 or 3 tokens (which we call a *bad token* configuration) and will locate the (near by) black hole by calling Procedure `LocalSearch`.

Algorithm Explore2Tokens

```
1: SmallRing //Take care the case of a torus with a ring of less than 4 nodes
2: //Any ring has at least 4 nodes
3: V_t ← ∅
4: C ← ∅
5: Choose a port H_s
6: loop
7:    H_f ← ExploreRing(H_s)
8:    if H_f = 0 then
9:       Insert H_s in set C
10:      if there is a new port not in C then
11:         Choose a new port H_s ∉ C
12:         Take(2)
13:      else
14:         Stop
15:   else
16:      Choose a port V_s ≠ H_s, H_f, V_t
17:      V_t ← Walk(V_s)
18:      repeat
19:         Move one step on the direction towards port m ≠ V_t
20:         Let m' be the incoming port
21:         Explore all paths of length 2 not starting with port m' and return
22:      until you found a token at a path ending at port H_s
23:      H_s ← m
24:      Traverse ports m', V_t, Take(1) and traverse port V_s
```

Procedure `LocalSearch`**:** Suppose the agent arrives from a direction *dir* at a node u where it finds a *bad token* configuration. This means that the black-hole is at a node v which is adjacent to u. The agent takes the following actions: If $dir = East$ ($dir = South$) then the agent searches (using `CautiousTest`) the adjacent nodes *South* and *North* (*West* and *East*) of u in this order; either the agent vanishes leaving behind another *bad token* configuration or otherwise decides that the black hole is *East* (*South*) of u respectively.

Theorem 4. *Algorithm* $UBHS_{5,3}$ *identifies a black hole in any type* 1 *torus using* 5 *scattered agents with constant memory and* 3 *movable tokens each.*

Algorithm $UBHS_{5,3}$ can also solve the BHS problem in tori of type 2 or 3. For tori of type 1 this algorithm is almost optimal since in view of Lemma 4, in such tori the problem cannot be solved using 5 agents with 1 token.

6 Conclusion

We showed that any constant number of scattered agents with constant memory and one movable token each cannot locate the black hole in all unoriented tori

Function ExploreRing(H_s)

```
1: RingCompleted ← FALSE
2: Put(2) //Ring start
3: p ← Walk(H_s)
4: repeat
5:    CorrectDir ← 0
6:    while (There is an untraversed port p' ≠ p) AND (CorrectDir = 0) do
7:       p'' ← Walk(p')
8:       Explore all paths of length 2 not starting with port p'' and return
9:       if no token is found during this exploration then
10:         CorrectDir ← p'
11:      Walk(p'') //Go back taking port p''
12:   RingCompleted ← (CorrectDir = 0)
13:   Walk(p)
14:   if (RingCompleted) AND (you see two tokens) then
15:      //The ring under exploration has 4 nodes
16:      Return 0
17:   else
18:      Take(1) (the second token) and return
19:   if ¬RingCompleted then
20:      Put(1) //Leave the second token
21:      p ← Walk(CorrectDir) //Advance the exploration
22: until RingCompleted
23: repeat
24:    Explore all paths of length 3 starting at ports different than p
25: until until you find a node with a token
26: Return the incoming port label
```

(type 0) mainly due to the fact that it is impossible for the agents with just one token to explore all such tori. However we also proved that one agent with constant memory and just two movable tokens *can* explore all unoriented tori. This result is optimal on the number of agents and tokens and has its own importance since it has been shown ([22]) that an agent with any constant number of tokens cannot explore all cubic planar graphs. It remains unclear whether a small team of agents (at least 5 are needed) with constant memory and two movable tokens each could solve the BHS problem in all unoriented tori. Since one agent with two tokens can explore all tori of type 0, a negative answer to the above question would need a different reasoning than in the one token scenario. We also showed that four (five) scattered agents with constant memory and one token are not able to locate a black hole in all semi-oriented tori of type 3 (2 or 1). We gave an almost optimal algorithm which enables five scattered agents with constant memory and three tokens each to locate the black hole in all semi-oriented tori of type 1, 2 or 3. This algorithm can be transformed to work with five agents and two tokens in all semi-oriented tori of type 3 (it will appear in the full version of

Algorithm $UBHS_{5,3}$ (each agent, in parallel)

```
1: repeat
2:    Put(2) //Start of "ring scan"
3:    count ← 0
4:    dir ← East
5:    repeat
6:       count ← count + 1
7:       repeat
8:          CautiousTest(1, dir)
9:          Walk(dir)
10:         t ← Read()
11:      until t > 0
12:      Danger ← BHConfig(t)
13:   until ((count ≥ 5)OR(Danger))
14:   if ¬Danger then
15:      dir ← South
16:      CautiousTest(1, dir) //try the next ring
17:      Take(2) //remove the home base
18:      Walk(dir)
19:      Danger ← BHConfig(Read())
20: until (Danger)
21: LocalSearch(dir)
```

the paper). While we conjecture that a small team of scattered agents (at least five) with constant memory equipped with a few tokens (at least two) would be able to locate a black hole in all totally unoriented tori, a tight solution remains an open question.

Aknowledgements: We wish to thank Dr. Shantanu Das for the discussion on the exploration problem of an unoriented torus by a finite automaton. We also wish to thank the anonymous referees for their helpful suggestions.

References

1. B. Balamohan, S. Dobrev, P. Flocchini, and N. Santoro. Asynchronous exploration of an unknown anonymous dangerous graph with o(1) pebbles. In *Proc. of 19th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 7355, pages 279–290, 2012.
2. M. A. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proc. of 35th Annual Symp. on Foundations of Computer Science*, pages 75–85, 1994.
3. M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proc. of 19th Symp. on Foundations of Computer Science*, pages 132–142, 1978.

4. L. Budach. Automata and labyrinths. *Mathematische Nachrichten*, 86(1):195–282, 1978.
5. J. Chalopin, S. Das, A. Labourel, and E. Markou. Black hole search with finite automata scattered in a synchronous torus. In *Proc. of 25th Int. Symp. on Distributed Computing*, LNCS 6950, pages 432–446, 2011.
6. J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for scattered black hole search in a ring. In *Proc. of 18th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 6796, pages 186–197, 2011.
7. C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *Proc. of 10th Int. Conf. on Principles of Distributed Systems*, LNCS 4305, pages 320–332, 2006.
8. J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2,3):229–242, 2006.
9. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *J. of Graph Theory*, 32(3):265–297, 1999.
10. S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring a dangerous unknown graph using tokens. In *Proc. of 5th IFIP Int. Conf. on Theoretical Computer Science*, pages 131–150, 2006.
11. 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.
12. S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90, 2007.
13. S. Dobrev, N. Santoro, and W. Shi. Using scattered mobile agents to locate a black hole in an un-oriened ring with tokens. *Int. J. of Foundations of Computer Science*, 19(6):1355–1372, 2008.
14. 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.
15. P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *Proc. of IEEE Int. Symp. on Parallel & Distributed Processing*, pages 1–10, 2009.
16. P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48:166–177, 2006.
17. R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Hardness and approximation results for black hole search in arbitrary graphs. *Theoretical Computer Science*, 384(2-3):201–221, 2007.
18. A. Kosowski, A. Navarra, and C. Pinotti. Synchronization helps robots to detect black holes in directed graphs. In *Proc. of 13th Int. Conf. on Principles of Distributed Systems*, pages 86–98, 2009.
19. D. Kozen. Automata and planar graphs. In *Proc. of Fundamentals of Computation Theory*, pages 243–254, 1979.
20. B. Mans. Optimal distributed algorithms in unlabeled tori and chordal rings. *Journal of Parallel and Distributed Computing*, 46(1):80–90, 1997.
21. M. Rabin. Maze threading automata. Seminar talk presented at the University of California at Berkeley, October 1967.
22. H. Rollik. Automaten in planaren graphen. *Acta Informatica*, 13:287–298, 1980.
23. C. E. Shannon. Presentation of a maze-solving machine. In *Proc. of 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics)*, pages 173–180, 1951.
24. W. Shi. Black hole search with tokens in interconnected networks. In *Proc. of 11th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems*, pages 670–682, 2009.