Black Hole Search with Finite Automata Scattered in a Synchronous Torus

Jérémie Chalopin[1], Shantanu Das[1], Arnaud Labourel[1], and Euripides Markou[2]

[1] LIF, Aix-Marseille University, Marseille, France.
jeremie.chalopin,shantanu.das,arnaud.labourel@lif.univ-mrs.fr
[2] Department of Computer Science and Biomedical Informatics, University of Central Greece, Lamia, Greece. emarkou@ucg.gr

Abstract. We consider the problem of locating a black hole in synchronous anonymous networks using finite state agents. A black hole is a harmful node in the network that destroys any agent visiting that node without leaving any trace. The objective is to locate the black hole without destroying too many agents. This is difficult to achieve when the agents are initially scattered in the network and are unaware of the location of each other. In contrast to previous results, we solve the problem using a small team of finite-state agents each carrying a constant number of identical tokens that could be placed on the nodes of the network. Thus, all resources used in our algorithms are independent of the network size.
We restrict our attention to oriented torus networks and first show that no finite team of finite state agents can solve the problem in such networks, when the tokens are not movable, i.e., they cannot be moved by the agents once they have been released on a node. In case the agents are equipped with movable tokens, we determine lower bounds on the number of agents and tokens required for solving the problem in torus networks of arbitrary size. Further, we present a deterministic solution to the black hole search problem for oriented torus networks, using the minimum number of agents and tokens, thus providing matching upper bounds for the problem.

1 Introduction

The exploration of an unknown graph by one or more mobile agents is a classical problem initially formulated in 1951 by Shannon [27] and it has been extensively studied since then (e.g., see [1, 8, 20]). 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* problem the goal for the agents is to locate the black hole within finite time. In particular, at least one agent has to survive knowing all edges leading to the black hole. The only way of locating a black hole is to have at least one agent visiting it. However, since any agent visiting a black hole is destroyed without leaving any trace, the location of the black hole must be deduced by some communication mechanism employed by the agents. Four such mechanisms have been proposed in the literature: a) the *whiteboard* model in which there is a whiteboard at each node of the network where the agents can leave messages, b) the *'pure' token* model where the agents carry tokens which they can leave at nodes, c) the *'enhanced' token* model in which the agents can leave tokens at nodes or edges, and d) the time-out mechanism (only for synchronous networks) in which one agent explores a new node while another agent waits for it at a safe node.

The most powerful inter-agent communication mechanism is having whiteboards at all nodes. Since access to a whiteboard is provided in mutual exclusion, this model could also provide the agents a 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., [9, 12–14]), the 'enhanced' token model (e.g., [10, 15, 28]) and the 'pure' token model in [18]. It has been proved that the problem can be solved with a minimal number of agents performing a polynomial number of moves. Notice that in an asynchronous network the number of the nodes of the network must be known to the agents otherwise the problem is unsolvable ([13]). If the graph topology is unknown, at least $\Delta + 1$ agents are needed, where Δ is the maximum node degree in the graph ([12]). Furthermore the network should be 2-connected. It is also not possible to answer the question of *whether* a black hole exists in the network.

In asynchronous networks, with scattered agents (not initially located at the same node), the problem has been investigated for the ring topology ([11, 13]) and for arbitrary topologies ([4, 19]) in the whiteboard model while in the 'enhanced' token model it has been studied for rings ([16, 17]) and for some interconnected networks ([28]).

The consideration of synchronous networks makes a dramatic change to the problem. Now two co-located distinct agents can discover one black hole in any graph by using the time-out mechanism, without the need of whiteboards or tokens. Moreover, 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) in [5, 7, 22–24] 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.

Indeed the problem seems to be much more difficult in the case of scattered agents and there are very few known results for this scenario. In this paper we study this version of the problem using very simple agents that can be modeled as finite state automata. Our objective is to determine the minimum resources, such as number of agents and tokens, necessary and sufficient to solve the problem in a given class of networks. For the class of ring networks, recent results [3] show that having constant-size memory is not a limitation for the agents when solving this problem. We consider here the more challenging scenario of anonymous torus networks of arbitrary size. We show that even in this case, finite state agents are capable of locating the black hole in all oriented torus networks using only a few tokens. Note that the exploration of anonymous oriented torus networks is a challenging problem in itself, in the presence of multiple identical agents [26]. Since the tokens used by the agents are identical, an agent cannot distinguish its tokens from those of another agent.

While the token model has been mostly used in the exploration of safe networks, the whiteboard model is commonly used in unsafe 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, while the 'enhanced' token model can be implemented having a $O(\log d)$-bit whiteboard on a node with degree d. In the whiteboard model, the capacity of each whiteboard is always assumed to be of at least $\Omega(\log n)$ bits, where n is the number of nodes of the network. In all previous papers studying the Black Hole Search problem under a token model apart from [18] and [3], the authors have used the 'enhanced' token model with agents having non-constant memory. The weakest 'pure' token model has been used in [18] for co-located non-constant memory agents equipped with a map in asynchronous networks.

The Black Hole Search problem has also been studied for co-located agents in asynchronous and synchronous directed graphs with whiteboards in [6, 24]. In [21] they study the problem in asynchronous networks with whiteboards and co-located agents without the knowledge of incoming link. A different dangerous behavior is studied for co-located agents in [25], where the authors consider a ring and assume black holes with Byzantine behavior, which do not always destroy a visiting agent.

Our Contributions: We consider the problem of locating the black hole in an anonymous but oriented torus containing exactly one black hole, using a team of identical agents that are initially scattered within the torus. Henceforth we will refer to this problem as the BHS problem. We focus our attention on very simple mobile agents. The agents have constant-size memory, they can communicate with other agents only when they meet at the same node and they carry a constant number of identical tokens which can be placed at nodes. The tokens may be movable (i.e. they can be released and picked up later) or unmovable (i.e. they cannot be moved by the agents once they have been released on a node). We prove the following results:

- No finite team of agents can solve the BHS problem in all oriented torus networks using a finite number of *unmovable* tokens.
- For agents carrying any finite number of *movable* tokens, at least three agents are required to solve the problem.

– Any algorithm for solving BHS using 3 agents requires more than one movable token per agent.
– The BHS problem can be solved using three agents and only two movable tokens per agent, thus matching both the lower bounds mentioned above.

In Section 2, we formally describe our model, giving the capabilities of the agents. In Section 3, we prove lower bound on the number of agents and tokens needed to solve the BHS problem in the torus. In Section 4, we present deterministic algorithms for BHS: (i) using $k \geq 3$ agents carrying 3 movable tokens per agents, and (ii) using $k \geq 4$ agents carrying 2 movable tokens per agent. We also present a more involved algorithm that uses exactly 3 agents and 2 tokens per agent thus meeting the lower bounds. All our algorithms are time-optimal and since they do not require any knowledge about the dimensions of the torus, they work in any synchronous oriented torus, using only a finite number of agents having constant-size memory. Due to space limitations, proofs of some of theorems and formal descriptions of the algorithms are omitted and can be found in [2].

2 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 of size $n \times m$, $n \geq 3$, $m \geq 3$. We assume that the torus is oriented, i.e., at each node, the four incident edges are consistently marked as North, East, South and West. Each mobile agent owns a constant number of t identical tokens which can be placed at any node visited by the agent. In all our protocols a node may contain at most three tokens at the same time and an agent carries at most three tokens at any time. A token or an agent at a given node is visible to all agents on the same node, but is not visible to any other agent. The agents follow the same deterministic algorithm and begin execution at the same time and being at the same initial state.

At any single time unit, a mobile agent occupies a node u of the network and may 1) detect the presence of one or more tokens and/or agents at node u, 2) release/take one or more tokens to/from the node u, and 3) decide to stay at the same node or move to an adjacent node. 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.

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 quadruplets (D, x, y, b) where $D \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{none}\}$ is the direction through which the agent has arrived at the node, x is the number of tokens (at most 3) at that node, y is number of tokens (at most 3) carried by the agent and $b \in \{true, false\}$ indicates whether there is at least another agent at the node or not. Elements of Λ are quadruplets (P, s, X, M) where $P \in \{\texttt{put}, \texttt{pick}\}$ is the action performed by the agent on the tokens, $s \in \{0, 1, 2, 3\}$ is the number of tokens concerned by the action A, $X \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{none}\}$ is the edge marked as dangerous by the agent, and $M \in \{\texttt{North}, \texttt{South}, \texttt{East}, \texttt{West}, \texttt{none}\}$ is the move performed by the agent. Note that the agent always performs the action before the marking and the marking before the move.

Note that all computations by the agents are independent of the size $n \times m$ of the network since the agents have no knowledge of n or 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. 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. Thus, our definition of a successful BHS scheme is slightly different from the original definition. The time complexity of a BHS scheme is the number of time units needed for completion of the scheme, assuming the worst-case location of the black hole and the worst-case initial placement of the scattered agents.

3 Impossibility results

In this section we give lower bounds on the number of agents and the number and type of tokens needed for solving the BHS problem in any anonymous, synchronous and oriented torus.

3.1 Agents with unmovable tokens

We will prove that any constant number of agents carrying a constant number of unmovable tokens each, can not solve the BHS problem in an oriented torus. The idea of the proof is the following: We show that an adversary (by looking at the transition function of an agent) can always select a big enough torus and initially place the agents so that no agent visits nodes which contain tokens left by another agent, or meets with another agent. Moreover there are nodes on the torus never visited by any agent. Hence the adversary may place the black hole at a node not visited by any of the agents to make the algorithm fail. This result is based on ideas presented earlier in [26].

Theorem 1. *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.*

3.2 Agents with movable tokens

We show the following impossibility results.

Lemma 1. *Two agents carrying any number of movable tokens cannot solve the BHS problem in an oriented torus even if the agents have unlimited memory.*

Proof. Assume w.l.o.g. that the first move of the agents is going East. Suppose that the black hole has been placed by an adversary at the East neighbor of an agent. This agent vanishes into the black hole after its first move. The adversary places the second agent such that it vanishes into the black hole after its first vertical move, or it places the agent in a horizontal ring not containing the black hole if the agent never performs vertical moves. Observe that the trajectories of the two agents intersect only at the black hole and neither can see any token left by the other agent. Neither of the agents will ever visit the East neighbor of the black hole and thus, they will not be able to correctly mark all links incident to the black hole.

Thus, at least three agents are needed to solve the problem. We now determine a lower bound on the number of tokens needed by three scattered agents to solve BHS.

Lemma 2. *There exists no algorithm that could solve the BHS problem in all oriented tori using three agents with constant memory and one movable token each.*

Proof. (Sketch) Clearly, in view of Theorem 1, an algorithm which does not instruct an agent to leave its token at a node, cannot solve the BHS problem. Hence any potentially correct algorithm should instruct an agent to leave its token down. Moreover this decision has to be taken after a finite number of steps (due to agents' constant memory). After that the agents visit new nodes until they see a token. We can show that if the agents visit only a constant number of nodes before returning to meet their tokens they cannot visit all nodes of the torus. If they move their tokens each time they see them and repeat the previous procedure (i.e., visit a constant number of nodes and return to meet their tokens), we can show that they will find themselves back at their initial locations and initial states without having met with other agents and leaving some nodes unvisited. An adversary may place the black hole at an unvisited node to make the algorithm fail. Now consider the case that at some point an algorithm instructs the agents to visit a non-constant number of nodes until they see a token (e.g., leave your token down and go east until you see a token). Again we can show that an adversary may initially place the agents and the black hole, and select the size of the torus so that two of the agents enter the black hole without leaving their tokens close to it: The agent (say A) that enters first into the black hole has been initially placed by an adversary so that it left its token more than a constant number of nodes away from the black hole. The adversary initially places another agent B so that it enters the black hole before it meets A's token. Furthermore B leaves its token more than a constant number of nodes away from the black hole. Hence the third agent, even if it meets the tokens left by A or B, it could not decide the exact location of the black hole.

Theorem 2. *At least three agents are necessary to solve the BHS problem in an oriented torus of arbitrary size. Any algorithm solving this problem using three agents requires at least two movable tokens per agent.*

4 Algorithms for BHS in a torus using movable tokens

Due to the impossibility result from the previous section, we know that unmovable tokens are not sufficient to solve BHS in a torus. In the following, we will use only movable tokens. To explore the torus an agent uses the Cautious-Walk technique [13] using movable tokens. In our algorithms, a *Cautious-Walk in direction D with x tokens* means the following (see Procedure Cautious-walk): (i) the agent releases a sufficient number of tokens such that there are exactly x tokens at the current node, (ii) the agent moves one step in direction D and if it survives, the agent immediately returns to the previous node, (iii) the agent picks up the tokens released in step (i) and again goes one step in direction D. If an agent vanishes during step (ii), any other agent arriving at the same location sees x tokens and realizes a potential danger in direction D. Depending on the algorithm an agent may use 1, 2, or 3 tokens to implement the Cautious-Walk.

Procedure Cautious-walk(Direction D, integer x)

```
  /* Procedure used by the agent to explore the nodes               */
1 Put tokens until there are $x$ tokens;
2 **Go** one step along $D$ and then go back;
  /* test if the node in direction $D$ is the black hole             */
3 Pick up the tokens released in the first step;
4 **Go** one step along $D$;
```

4.1 Solving BHS using $k \geq 3$ agents and 3 tokens

We show that three agents suffice to locate the black hole if the agents are provided with three tokens using the algorithm presented below.

Algorithm BHS-torus-33:
An agent explores one horizontal ring at a time and then moves one step South to the next horizontal ring and so on. When exploring a horizontal ring, the agent leaves one token on the starting node. This node is called the *homebase* of the agent and the token left (called homebase token) is used by the agent to decide when to proceed to the next ring. The agent then uses the two remaining tokens to repeat Cautious-Walk in the East direction until it has seen twice a node containing one token. Any node containing one token is a homebase either of this agent or of another agent. The agent moves to the next horizontal ring below after encountering two homebases. However before moving to the next ring, it does a cautious walk in the South direction with three tokens (the two tokens it carries plus the homebase token). If the agent survives and the node reached by the agent has one token, the agent repeats a cautious walk in the East

direction (with two tokens) until it reaches an empty node. The agent can now use this empty node as its new homebase. It then repeats the same exploration process for this new ring leaving one token at its new homebase.

Whenever the agent sees two or three tokens at the end of a cautious-walk, the agent has detected the location of the black hole: If there are two (resp. three) tokens at the current node, the black hole is the neighboring node w to the East (resp. South). In this case, the agent stops its normal execution and then traverses a cycle around node w, visiting all neighbors of w and marking all the links leading to w as dangerous.

Theorem 3. *Algorithm BHS-torus-33 correctly solves the BHS problem with* 3 *or more agents.*

Proof. An agent may visit an unexplored node only while going East or South. If one agent enters the black hole going East (resp. South), there will be two (resp. three) tokens on the previous node and thus, no other agent would enter the black hole through the same link. This implies that at least one agent always survives. Whenever an agent encounters two or three tokens at the end of a Cautious-Walk, the agent is certain of the location of the black hole since any alive agent would have picked up its Cautious-Walk tokens in the second step of the cautious walk (The agents move synchronously always using cautious walk and taking three steps for each move).

4.2 BHS using $k \geq 4$ agents and 2 tokens each

We now present an algorithm that uses only two tokens per agent, but requires at least 4 agents to solve BHS.

During the algorithm, the agents put two tokens on a node u to signal that either the black hole is on the South or the East of node u. Eventually, both the North neighbor and the West neighbor of the black hole are marked with two tokens. Whenever there is a node v such that there are exactly two tokens at both the West neighbor of v and the North neighbor of v, then we say that there exists a *Black-Hole-Configuration* (BHC) at v.

Algorithm BHS-torus-42:
The agent puts two tokens on its starting node (homebase). It then performs a Cautious-Walk in the East direction. If the agent survives, it returns, picks up one token and repeats the Cautious-Walk with one token in the East direction (leaving the other token on the homebase) until it reaches a node containing one or two tokens.

- If the agent reaches a node u containing two tokens, then the black hole is the next node on the East or on the South (See Property C of Proposition 1). The agent stops its exploration and checks whether the black hole is the East neighbor or the South neighbor.

– Whenever an agent reaches a node containing one token, it performs a Cautious-Walk in East direction with two tokens and then continues the Cautious-Walk in the same direction with one token. If the agent encounters three times[3] a node containing one token, it moves to the next horizontal ring below. To do that it first performs a Cautious-Walk with two tokens in the South direction. If the agent survives and reaches the ring below, it can start exploring this horizontal ring. If the current node is not empty, the agent repeats a cautious walk with two token in the East direction until it reaches an empty node. Now the agent repeats the same exploration process using the empty node as its new homebase. Whenever the agent encounter a node with two tokens, it stops its exploration and checks whether the black hole is the East or South neighbor.

In order to check if the black hole is the East neighbor v or the South neighbor w of the current node u (containing two tokens), the agent performs the following actions: The agent reaches the West neighbor x of w in exactly three time units by going West and South (and waiting one step in between). If there are two tokens on this node x then w is the black hole. Otherwise, the agent performs a cautious walk in the East direction with one token (or with two tokens if there is already one token on node x). If it safely reaches node w, then the black hole is the other node v. Otherwise the agent would have fallen into the black hole leaving a BHC at node w. An agent that discovers the black hole, traverses a cycle around the black hole visiting all its neighbors and marking all the links leading to it as dangerous.

Proposition 1. *During an execution of BHS-torus-42 with $k \geq 4$ agents, the following properties hold:*

A When an agent checks the number of tokens at a node, all surviving agents have picked up their cautious-walk token.
B At most three agents can enter the black hole:
  (a) at most one agent going South leaving two tokens at the previous node.
  (b) at most two agents going East, each leaving one of its tokens at the previous node.
C When an agent checks the number of tokens at a node, if there are two tokens then the black hole is either the East or the South neighbor of the current node.
D After an agent starts exploring a horizontal ring, one of the following eventually occurs:
  (a) If this ring is safe, the agent eventually moves to the next horizontal ring below.
  (b) Otherwise, either all agents on this ring fall into the black hole or one of these agents marks all links to the black hole.

[3] The agent may encounter homebases of two agents which have both fallen into the black hole. (In this case it must continue in the same direction until it locates the black hole)

Theorem 4. *Algorithm BHS-torus-42 correctly solves the black hole search problem with $k \geq 4$ agents, each having two tokens.*

Proof. Property B of Proposition 1 guarantees that at least one agent never enters the black hole. Property D ensures that one of the surviving agents will identify the black hole. Property C shows that if the links incident to a node w are marked as dangerous by the algorithm, then node w is the black hole.

4.3 BHS with 3 agents and 2 movable tokens

Using the techniques presented so far, we now present the algorithm that uses exactly 3 agents and two tokens per agent. The algorithm is quite involved and we present here only the main ideas of the algorithm. The complete algorithm along with a proof of correctness can be found in [2]. Notice first that we can not prevent two of the three agents from falling into the black hole (see proof of Lemma 1). To ensure that no more than two agents enter the black hole, the algorithm should allow the agent to move only in two of the possible four directions (when visiting unexplored nodes). When exploring the first horizontal ring, an agent always moves in the East direction, using a Cautious-Walk as before and keeping one token on the starting node (homebase). This is called procedure *First-Ring*. Once an agent has completely explored one horizontal ring, it explores the ring below, using procedure *Next-Ring*. During procedure *Next-Ring*, an agent traverses the already explored ring and at each node u of this ring, the agent puts one token, traverses one edge South (to check the node just below node u), and then immediately returns to node u and picks up the token. Note that an agent may fall into the black hole only when going South during procedure *Next-Ring* or when going East during procedure *First-Ring*. We ensure that at most one agent falls into the black hole from each of these two directions. The surviving agent must then identify the black hole from the tokens left by the other agents.

Algorithm 1: BHS-Torus-32

```
  /* Algorithm for BHS in Oriented Torus (3 agents, 2 tokens)       */
1 First-Ring;
2 repeat
3     Init-Next-Ring;
4     Next-Ring;
5 until until black hole is found;
```

For this algorithm, we redefine the *Black-Hole-Configuration* (BHC) as follows: If there is a node v such that there is one or two tokens at both the West neighbor of v and the North neighbor of v, then we say that a BHC exists at v. The algorithm should avoid forming a black hole configuration at any other node except the black hole. In particular, when the agents put tokens on their homebase, these tokens should not form a BHC. This requires coordination between any two agents that are operating close to each other (e.g. in adjacent rings of the torus) and it is not always possible to ensure that a BHC is never formed at a safe node.

The main idea of the algorithm is to make two agents meet whenever they are close to each other (this requires a complicated synchronization and checking procedure). If any two agents manage to gather at a node, we can easily solve BHS using the standard procedure for colocated agents[4] with the time-out mechanism (see [3, 5]) . On the other hand, if the agents are always far away from each other (i.e. more than a constant distance) then they do not interfere with the operations of each other until one of them falls into the black hole. The agents explore each ring, other than the first ring, using procedure *Next-Ring*. We have another procedure called *Init-Next-Ring* that is always executed at the beginning of *Next-Ring*, where the agents check for certain special configurations and take appropriate action. If the tokens on the potential homebases of two agents would form a BHC on a safe node, then we ensure two agents meet.

Synchronization:
During the algorithm, we ensure that two agents meet if they start the procedure *Init-Next-Ring* from nearby locations. We achieve this by keeping the agents synchronized to each other, in the sense that they start executing the procedure at the same time, in each iteration. More precisely, we ensure the following property:

Property 1. When one agent starts procedure *Init-Next-Ring*, any other surviving agent either (i) starts procedure *Init-Next-Ring* at exactly the same time, or (ii) waits in its current homebase along with both its tokens during the time the other agent is executing the procedure or, (iii) has not placed any tokens at its homebase.

Notice that if there are more than one agent initially located at distinct nodes within the same horizontal ring, an agent cannot distinguish its homebase from the homebase of another agent, and thus an agent would not know when to stop traversing this ring and go down to the next one. We get around this problem by making each agent traverse the ring a sufficient number of times to ensure that every node in this ring is explored at least once by this agent. To be more precise, each agent will traverse the ring until it has encountered a homebase node six times (recall that there can be either one, two or three agents on the same ring). This implies that in reality the agent may traverse the same ring either twice, or thrice, or six times. If either all agents start in distinct rings or if all start in the same ring then, the agents would always be synchronized with each other (i.e. each agent would start the next iteration of *Next-Ring* at the same time). The only problem occurs when two agents start on the same ring and the third agent is on a different ring. In this case, the first two agents will be twice as fast as the third agent. We introduce waiting periods appropriately to ensure that Property 1 holds.

[4] Note that the agents meeting at a node can be assigned distinct identifiers since they would arrive from different directions.

For both the procedures *First-Ring* and *Next-Ring*, we define one *big-step* to be the period between when an agent arrives at a node v from the West with its token(s) and when it has left node v to the East with its token(s). During a big-step, the agent would move to an unsafe node (East or South), come back to pick its token, wait for some time at v before moving to the next node with its token. The waiting period is adjusted so that an agent can execute the whole procedure *Init-Next-Ring* during this time. Thus, the actual number of time units for each big-step is a fixed constant D.

Algorithm *BHS-Torus-32*:

Procedure *First-Ring*:
During this procedure the agent explores the horizontal ring that contains its starting location. The agent puts one token on its homebase and uses the other token to perform cautious-walk in the direction East, until it enters a node with some tokens. If it finds a node with two tokens then the next node is the black hole. Thus, the agent has solved BHS. Otherwise, if the agent finds a node with a single token this is the homebase of some agent (maybe itself). The agent puts the second token on this node and continues the walk without any token (i.e. it imitates the cautious-walk). If it again encounters a node with a single token, then the next node is the black hole and the algorithm terminates. Otherwise, the agent keeps a counter initialized to one and increments the counter whenever it encounters a node containing two tokens. When the counter reaches a value of six, the procedure terminates. At this point the agent is on a node with two tokens (which it can use for the next procedure).

Unless an agent enters or locates the black hole, the procedure *First-Ring* requires exactly $6nD$ time units for an agent that is alone in the ring, $3nD$ for two agents that start on the same ring, and $2nD$ if all the three agents start on the same ring.

Procedure *Init-Next-Ring*:
An agent executes this procedure at the start of procedure *Next-Ring* in order to choose its new homebase for exploring the next ring. The general idea is that the agent checks the node u on the South of its current location, move its two tokens to the East, and then goes back to u. If there is another agent that has started *Next-Ring* on the West of u (i.e., without this Procedure, the homebases of the two agents would have formed a BHC), the agents can detect it, and *Init-Next-Ring* is designed in such a way that the two agents meet. More precisely, when an agent executes *Init-Next-Ring* on horizontal ring i without falling into the black hole, we ensure that either (i) it meets another agent, or (ii) it locates the black hole, or (iii) it detects that the black hole is on ring $i + 2$, or (iv) the token it leaves on its homebase does not form a BHC with a token on ring $i + 1$. In case (iii), the agent executes *Black-Hole-in-Next-Ring*; in case (iv), it continues the execution of *Next-Ring*.

Procedure *Next-Ring*:
The agent keeps one token on the homebase and with the other token performs a special cautious-walk during which it traverses the safe ring and at each node it puts a token, goes South to check the node below, returns back and moves the token to the East. The agent keeps a counter initialized to zero, which it increments whenever it sees a node with a token on the safe ring. When the agent sees a token on the safe ring, it does not go South, since this may be dangerous. Instead, the agent goes West and South, and if it does not see any token there, it puts a token and goes East. If the agent enters the black hole, it has left a BHC. When the counter reaches a value of six, the procedure terminates.

During the procedure, an agent keeps track of how many (1 or 2) tokens it sees in the safe ring and the ring below. This information is stored as a sequence (of length at most 24). At the end of the procedure, using this sequence, an agent in the horizontal ring i can detect whether (i) the Black hole lies in the horizontal ring $i+1$ or $i+2$, or, (ii) there are two other agents in the ring i and ring $i+1$ respectively, or, (iii) the ring $i+1$ does not contain the black hole. In scenario (i), the agent executes procedure *Black-Hole-in-Next-Ring*; in scenario (ii), the agent meets with the other agent in the same ring; in scenario (iii), the agent moves to the next horizontal ring (i.e. ring $i+1$) to start procedure *Init-Next-Ring* again.

Procedure *Black-Hole-in-Next-Ring*:
The agent executes this procedure only when it is certain that the black hole lies in the ring below its current position. The procedure is similar to *Next-Ring*; the main difference being that the agent does not leave a homebase token. During the procedure, either (i) the agent detects the location of the black hole and marks all links to the black hole or (ii) the agent falls into the black hole, forming a BHC at the black hole.

Theorem 5. *Algorithm* BHS-Torus-32 *correctly solves the BHS problem in any oriented torus with exactly three agents carrying two tokens each.*

5 Conclusions

We showed that at least three agents are needed to solve BHS in oriented torus networks and these three agents must carry at least two movable tokens each for marking the nodes. The algorithm BHS-Torus-32 uses the smallest possible team of agents (i.e., 3) carrying the minimum number of tokens (i.e., 2) and thus, it is optimal in terms of resource requirements. However, on the downside this algorithm works only for $k = 3$ agents. In combination with algorithm BHS-Torus-42 (which solves the problem for any $k \geq 4$ agents carrying 2 tokens each), these algorithms can solve the black hole search problem for any $k \geq 3$, if the value of k is known. Unfortunately, algorithms BHS-Torus-32 and BHS-Torus-42 cannot be combined to give an algorithm for solving the BHS problem for any $k \geq 3$ agents without the knowledge of k: Algorithm BHS-Torus-32 for 3 agents will not correctly locate the black hole if the agents are more than 3, while in the algorithm BHS-Torus-42, 3 of the agents may fall into the black hole. Hence, whether the problem can be solved for $k \geq 3$ agents equipped with 2 tokens, without any knowledge of k, remains an interesting (and we believe challenging) open question. Another interesting open problem is to determine the minimum size of a team of agents carrying one token each, that can solve the BHS problem. Note that the impossibility result for three agents carrying one token each, does not immediately generalize to the case of 4 or more agents, as in those cases, we cannot exclude the possibility that two surviving agents manage to meet.

It is interesting to compare our results with the situation in a synchronous, oriented, anonymous ring, which can be seen as a one dimensional torus ([3]): The minimum trade-offs between the number of agents and the number of tokens, in this case, are 4 agents with 2 unmovable tokens or 3 agents with 1 movable token each. Additionally, in an *unoriented* ring the minimum trade-offs are 5 agents with 2 unmovable tokens or 3 agents with 1 movable token each whereas the situation in an unoriented torus has not been studied. Hence another open problem is solving the BHS problem in a d-dimensional torus, $d > 3$, as well as in other network topologies.

References

1. M. A. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 75–85, 1994.
2. J. Chalopin, S. Das, A. Labourel, and E. Markou. Black hole search with finite automata scattered in a synchronous torus, arxiv:1106.6037, 2011.
3. J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for scattered black hole search in a ring. In *Proceedings of 18th International Colloquium on Structural Information and Communication Complexity*, lncs 6796, pages 186–197, 2011.
4. J. Chalopin, S. Das, and N. Santoro. Rendezvous of mobile agents in unknown graphs with faulty links. In *Proceedings of 21st International Conference on Distributed Computing*, pages 108–122, 2007.
5. C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *Proceedings of 10th International Conference on Principles of Distributed Systems*, LNCS 4305, pages 320–332, 2006.
6. J. Czyzowicz, S. Dobrev, R. Kralovic, S. Miklik, and D. Pardubska. Black hole search in directed graphs. In *Proceedings of 16th International Colloquium on Structural Information and Communication Complexity*, pages 182–194, 2009.
7. J. Czyzowicz, D. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability & Computing*, 16(4):595–619, 2007.
8. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.
9. S. Dobrev, P. Flocchini, R. Kralovic, G. Prencipe, P. Ruzicka, and N. Santoro. Optimal search for a black hole in common interconnection networks. *Networks*, 47(2):61–71, 2006.
10. S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring a dangerous unknown graph using tokens. In *Proceedings of 5th IFIP International Conference on Theoretical Computer Science*, pages 131–150, 2006.

11. S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *Proceedings of 6th International Conference on Principles of Distributed Systems*, pages 34–46, 2003.
12. 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.
13. S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90, 2007.
14. S. Dobrev, P. Flocchini, and N. Santoro. Improved bounds for optimal black hole search in a network with a map. In *Proceedings of 10th International Colloquium on Structural Information and Communication Complexity*, pages 111–122, 2004.
15. S. Dobrev, R. Kralovic, N. Santoro, and W. Shi. Black hole search in asynchronous rings using tokens. In *Proceedings of 6th Conference on Algorithms and Complexity*, pages 139–150, 2006.
16. S. Dobrev, N. Santoro, and W. Shi. Scattered black hole search in an oriented ring using tokens. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2007.
17. S. Dobrev, N. Santoro, and W. Shi. Using scattered mobile agents to locate a black hole in an un-oriened ring with tokens. *International Journal of Foundations of Computer Science*, 19(6):1355–1372, 2008.
18. P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *Proceedings of 22nd International Symposium on Distributed Computing*, LNCS 5218, pages 227–241, 2008.
19. P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *Proceedings of IEEE International Symposium on Parallel & Distributed Processing*, pages 1–10, 2009.
20. P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48:166–177, 2006.
21. P. Glaus. Locating a black hole without the knowledge of incoming link. In *Proceedings of 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 128–138, 2009.
22. 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.
23. R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Approximation bounds for black hole search problems. *Networks*, 52(4):216–226, 2008.
24. A. Kosowski, A. Navarra, and C. Pinotti. Synchronization helps robots to detect black holes in directed graphs. In *Proceedings of 13th International Conference on Principles of Distributed Systems*, pages 86–98, 2009.
25. R. Kràlovic and S. Miklìk. Periodic data retrieval problem in rings containing a malicious host. In *Proceedings of 17th International Colloquium on Structural Information and Communication Complexity*, pages 156–167, 2010.
26. E. Kranakis, D. Krizanc, and E. Markou. Deterministic symmetric rendezvous with tokens in a synchronous torus. *Discrete Applied Mathematics*, 159(9):896–923, 2011.
27. C. E. Shannon. Presentation of a maze-solving machine. In *Proceedings of 8th Conference of the Josiah Macy Jr. Found. (Cybernetics)*, pages 173–180, 1951.
28. W. Shi. Black hole search with tokens in interconnected networks. In *Proceedings of 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 670–682, 2009.