Black Hole Search in Common Interconnection Networks[*]

S. Dobrev[†] P. Flocchini[†] R. Královič[‡][§] G. Prencipe[¶] P. Ružička[||]
N. Santoro[**]

Abstract

Mobile agents operating in networked environments face threats from other agents as well as from the hosts (i.e., network sites) they visit. A *black hole* is a harmful host that destroys incoming agents without leaving any trace. To determine the location of such a harmful host is a dangerous but crucial task, called *black hole search.* The most important parameter for a solution strategy is the number of agents it requires (the *size*); the other parameter of interest is the total number of moves performed by the agents (the *cost*). It is known that at least *two* agents are needed; furthermore, with full topological knowledge, $\Omega(n \log n)$ moves are required in *arbitrary* networks. The natural question is whether, in *specific* networks, it is possible to obtain (topology-dependent but) more cost efficient solutions. It is known that this is not the case for rings. In this paper, we show that this negative result does not generalizes. In fact, we present a general strategy that allows *two* agents to locate the black hole with $O(n)$ moves in common interconnection networks: *hypercubes*, *cube-connected cycles*, *star graphs*, *wrapped butterflies*, *chordal rings*, as well as in multidimensional *meshes* and *tori* of restricted diameter. These results hold even if the networks are anonymous.

Keywords: Mobile Agents, Malicious Host, Distributed Search, Traversal Pair, Biconnected Graphs, Interconnection Networks, Anonymous Networks, Distributed Mobile Computing.

[*]**This paper is dedicated to the memory of our co-author and friend Peter Ružička, who recently and unexpectedly departed. His presence and inspiration will always be with us.**

[†]School of Information Technology and Engineering, *University of Ottawa*, {sdobrev,flocchin}@site.uottawa.ca.

[‡]Department of Computer Science, *Comenius University, Bratislava*, kralovic@dcs.fmph.uniba.sk

[§]Supported in part by grant APVT-20-018902

[¶]Dipartimento di Informatica, *University of Pisa*, prencipe@di.unipi.it.

[||]Department of Computer Science, *Comenius University, Bratislava*, ruzicka@dcs.fmph.uniba.sk

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

1 Introduction

The use of mobile agents is becoming increasingly popular when computing in networked environments, ranging from Internet to the Data Grid, both as a theoretical computational paradigm and as a system-supported programming platform.

Computing in such environments is termed *distributed mobile computing*, and has recently been the focus of extensive theoretical research (e.g., see [1, 3, 5, 6, 9, 10, 12, 15, 17, 22]). In its terminology, a network site is a *host*; local processes are *stationary agents*; mobile agents *navigate* moving from host to neighboring host, and perform computations at each host, according to a predefined set of behavioral rules called *protocol*, the same for all agents. In the setting we consider, the agents are *asynchronous* in their actions (e.g., computation, movement, etc) (i.e., the amount of time required by an action is finite but otherwise unpredictable). The hosts provide a storage area called *whiteboard* for incoming agents to communicate and compute, and its access is held in fair mutual exclusion.

The major *practical* concern in these systems is definitely *security* [4, 11, 16, 19]. Among the severe security threats faced in distributed mobile computing environments, two are particularly troublesome: *harmful agent* (that is, the presence of a malicious mobile process), and *harmful host* (that is, the presence at a network site of a harmful stationary process). The former problem is particularly acute in unregulated non-cooperative settings such as Internet (e.g., e-mail transmitted viruses). The latter not only exists in those settings, but also in environments with regulated access and where agents cooperate towards common goals (e.g., sharing of resources or distribution of a computation on the Grid [2]). In fact, a single local (hardware or software) failure might render a host harmful.

The problem posed by the presence of a harmful host has been intensively studied from a programming point of view (e.g., see [13, 14, 20, 18, 21]), and recently also from an algorithmic prospective [7, 8]. Obviously, the first step in any solution to such a problem must be to *identify*, if possible, the harmful host; i.e., to determine and report its location; following this phase, a "rescue" activity would conceivably be initiated to deal with the destructive process resident there. Depending on the nature of the danger, the task to identify the harmful host might be difficult, if not impossible, to perform.

Consider the presence in the network of a *black hole*: a host that *disposes* of visiting agents upon their arrival, leaving *no observable trace* of such a destruction [7, 8]. The task is to unambiguously determine and report the location of the black hole, and will be called *black hole search.*

Note that this type of highly harmful host is not rare; for example, the undetectable crash failure of a site in an asynchronous network turns such a site into a black hole. Hence the problem is relatively common.

Consider how a team of searching agents can solve this problem. The searching agents start from the same safe site, the *home base*; the task is successfully completed if, within finite time, at least one agent survives and knows the location of the black hole. The research concern is to determine under what conditions and at what cost mobile agents can successfully accomplish this task.

Some answers follow from simple facts. For example, if the network is not biconnected, the problem is unsolvable[1]; hence, we will only consider biconnected networks. Similarly, at

[1]i.e., no deterministic protocol exists which always correctly terminates.

least two agents are needed to solve the problem.

The problem has been investigated and its solutions characterized for *ring* networks [7]. Subsequently, the problem has been studied also for arbitrary networks and different solutions and matching lower bounds were presented, depending on the amount of topological information available to the agents [8]. In particular, if the agents have full knowledge of the network topology, *two agents* are sufficient, and can locate the black hole using $\Theta(n \log n)$ moves.

A natural question to ask is whether the $O(n \log n)$ bound for two agents with full topological knowledge of a general network can be improved for networks with special topologies. A negative result holds for rings where $\Omega(n \log n)$ moves are needed by any two-agents solution [7].

In this paper we show that the negative result for rings does not generalize. On the contrary, we present a general technique for efficient black hole location and prove that its application leads to $\Theta(n)$ protocols for most of the frequently used interconnection networks: *hypercubes*, *cube-related networks*, *chordal rings*, and multidimensional *tori* and *meshes* of restricted diameter. These results hold even if the networks are anonymous (i.e., the nodes are undistinguishable).

These results are obtained by exploiting the properties of the *traversal pair* of a biconnected graphs, a novel concept we introduce and analyze in this paper. In particular, we show how to construct a traversal pair of an arbitrary biconnected graph; and analyze the properties of traversal pairs in several common interconnection networks. We then present a general solution protocol for two agents, $\mathcal{TP}$, based on the constructed traversal pair, that allows two searching agents to efficiently locate the black hole. The properties of traversal pairs lead to the $\Theta(n)$ bound in common interconnection networks.

We also show that, for the class of networks considered here, full topological knowledge is not necessary and *topological awareness* suffices: in fact, both the network size and the position of the *home base* can be efficiently determined from topological awareness.

The paper is organized as follows. In the next section we present the model, definitions and basic properties. In Section 3, we introduce the notion of traversal pair, present the algorithm for locating the black hole using this notion and derive its complexity in terms of attributes of the traversal pair it uses. In Section 4 we show how to construct traversal pairs, analyze their properties in specific networks and apply the results to obtain $\Theta(n)$ black hole location algorithm for most commonly used interconnection networks. Finally, in Section 5 we show how to relax the somewhat strong requirements on the structural information available to the agents.

2 Definitions and Basic Properties

Let $G = (V, E)$ be a simple biconnected graph; let $n = |V|$ be the size of G, $E(x)$ be the links incident on $x \in V$, $d(x) = |E(x)|$ denote the degree of x, and Δ denote the maximum degree in G. If $(x, y) \in E$ then x and y are said to be neighbors. The nodes of G can be *anonymous* (i.e., without unique names).

At each node x, there is a distinct label (called port number) associated to each of its incident links (or ports); let $\lambda_x(x, z)$ denote the label associated at x to the link $(x, z) \in E(x)$, and λ_x denote the overall injective mapping at x. The set $\lambda = \{\lambda_x | x \in V\}$ of those mappings is called a *labelling* and we shall denote by (G, λ) the resulting edge-labelled graph.

Operating in (G, λ) is a team of two autonomous mobile agents. The agents can move from a node to a neighboring node in G, have computing capabilities and bounded computational storage ($O(\log n)$ bits suffice for all our algorithms), obey the same set of behavioral rules (the *protocol*). The agents are *asynchronous* in the sense that every action they perform (computing, moving, etc.) takes a finite but otherwise unpredictable amount of time. Initially, all agents are in the same node h, called *home base*.

Each node has a bounded amount of storage, called *whiteboard*; $O(\log n)$ bits suffice for all our algorithms. Agents communicate by reading from and writing on the whiteboards; access to a whiteboard is gained fairly in mutual exclusion.

We can assume that the agents have unique names without loss of generality. In fact, should the agents be initially anonymous, distinct names can be easily assigned; e.g., by having a counter on the whiteboard of *home base*, and having each agent increasing the counter and acquiring the current value as its name.

A *black hole* (shortly Bh) is a node where a stationary process resides that destroys any agent arriving at that node; no observable trace of such a destruction will be evident outside the node. The location of the black hole is unknown to the agents. The Black Hole Search (shortly Bhs) problem is to find the location of the black hole. More precisely, Bhs is solved if at least one agent survives, and the surviving agents know the location of the black hole.

The main measure of complexity of a solution protocol $\mathcal{P}$ is the number of agents used to locate the black hole, called the *size* of $\mathcal{P}$. Clearly

Lemma 2.1 ([7]). *At least two agents are needed to locate the black hole.*

The actual number of agents depends also on the amount of a priori network information the agents have. We assume that the agents have *complete topological knowledge* of (G, λ); that is, they have available: (1) knowledge of the labelled graph (G, λ); (2) correspondence between port labels and the link labels of (G, λ); and (3) location of the home base in (G, λ).

Example. Consider a 5×9 mesh with the source node at position $(2, 3)$ from the lower left corner. (1) means that the agents know they are in 5×9 mesh; note that (1) implies the knowledge of n. (2) means that the agents know for each node which links lead to *north*, *east*, *south* and *west*; this knowledge implies the ability to optimally route between any two nodes, even if there are given "forbidden" nodes that have to be avoided. (3) means that the agents know that they start at position $(2, 3)$ from the lower left corner.

Lemma 2.2 ([8]). *With complete topological knowledge, two agents suffice to locate the black hole.*

The other measure of complexity is the total number of moves performed by the agents, called the *cost* of $\mathcal{P}$. We are interested in size-optimal cost-efficient protocols.

At any moment of the execution of a protocol, the ports will be classified as *unexplored* – no agent has been sent/received via this port, *explored* – an agent has been received via this port or *dangerous* – an agent has been sent via this port, but no agent has been received via it. Obviously, an explored port does not lead to a black hole (we will call such ports also *safe*); on the other hand, both unexplored and dangerous ports might lead to it. To minimize the number of casualties (i.e., agents entering the black hole), we will not allow any agent to leave through a dangerous port. To prevent the execution from stalling, we will require any dangerous port not leading to the black hole, to be made explored as soon as possible.

This is accomplished as follows: Whenever an agent a leaves a node u through an unexplored port (transforming it into dangerous), upon its arrival to the node v, and before proceeding somewhere else, a returns to u (transforming that port into explored). This technique is called *Cautious Walk* and has been employed in [7, 8]. A node is considered *safe* if at least one of its incident edges is explored.

3 The BHS Protocol

3.1 Overview

The approach our agents will use consists in cooperatively and dynamically dividing the work between them. Specifically, the unexplored area is partitioned into two parts of (almost) equal size. Each agent explores one part without entering the other one. Since the parts are disjoint, one of them does not contain the black hole and the corresponding agent will complete its exploration. When this happens, the agent reaches the last safe node visited by the other agent and partitions whatever is still left to be explored, leaving a note for the other agent (should it be still alive). This process is repeated until the unexplored area consists of a single node: the black hole. Since the unexplored area is almost halved each time, the number of times (i.e., "rounds") the process must be repeated is $O(\log n)$.

In this approach, there are two costs, the one due to the *exploration* (i.e., the moves needed to explore the nodes), and the one due to *communication* (i.e., the moves needed by an agent to notify the other of a new partition). Using this type of approach in a ring network (like in [7]), the agents explore by moving in *opposite directions*, and thus the total exploration cost is $O(n)$ moves; however, the communication cost between the agents consists of $O(n)$ moves in each round, with a total of $O(\log n)$ rounds, yielding an overall cost of $O(n \log n)$ moves.

If G has an Hamiltonian circuit, we could use this circuit for the exploration (like in a ring) and use the other links as shortcuts to reduce the communication cost. The research question then becomes:

(1) How to find good shortcuts and how to estimate the resulting communication costs?

If G is not Hamiltonian, then we have the additional more *important* research question

(2) What structure, other than a circuit, would allow the agents to explore the network moving in "opposite directions"?

The answer, as we will see, is the novel notion of *Traversal Pair* ($\mathcal{TP}$) of a biconnected graph. The BHS algorithm we construct will use a traversal pair $\mathcal{TP}$, not only to indicate the order in which the network must be explored by the agents, but also to indicate to the agents how to avoid "dangerous" parts of the network.

The properties of a traversal pair $\mathcal{TP}$ will allow us to answer the first question for all biconnected G even if they are not Hamiltonian.

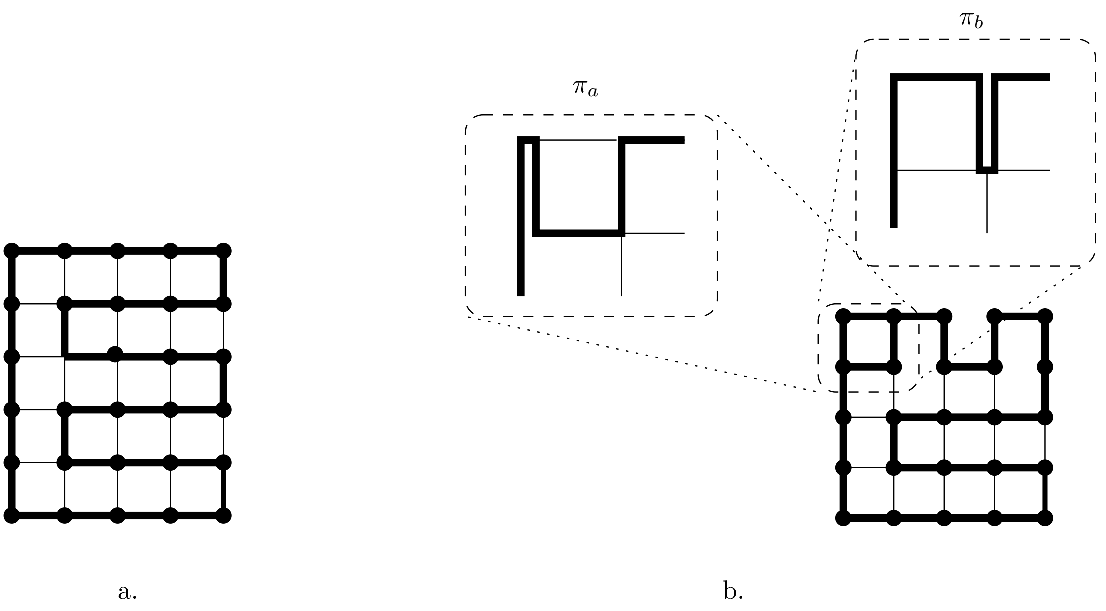

Figure 1: (a) Hamiltonian circuit defining a $\mathcal{TP}$ pair for a mesh with at least one side of even length. In this case π_b is a reverse of π_a. (b) $\mathcal{TP}$ of a mesh with both sides of odd length (only the two top rows differ from the even case). In this case π_a differs from π_b.

3.2 Traversal Pair

In the rest of the paper, we denote by $<_G$ an arbitrary fixed total ordering $v_1 <_G v_2 \ldots <_G v_n$ of the nodes of G. We say that a walk $v_1, v_2, \ldots, v_n$ *explores* a node $v_i = w$ at (logical) round i, if v_i is the first occurrence of w in the walk.

Definition 3.1 (Traversal Pair). *Let $G = (V, E)$ be an n-node graph with a total ordering $<_G$ of its nodes. Let π_a and π_b be two walks in G starting from v_1 and v_n, respectively, and exploring the nodes of G in the order $v_1, v_2, \ldots, v_n$ and $v_n, v_{n-1}, \ldots, v_1$, respectively. Then $\pi = (\pi_a, \pi_b)$ is called v_1-v_n* traversal pair *of G with respect to $<_G$.*

We will call π_a (resp. π_b) the *left* (resp. *right*) traversal (see Figure 1), and by $|\pi_a|$ (resp. $|\pi_b|$) the length of π_a (resp. π_b). Note that, in general $|\pi_a|$ need not be equal to $|\pi_b|$.

The above definition binds a v_1-v_n $\mathcal{TP}$ to the ordering $<_G$. Throughout the paper we will need the following more general notions:

Definition 3.2.

1. *G has an u-v $\mathcal{TP}$, with $u, v \in V$, if $\exists$ an ordering $<_G$ and an u-v $\mathcal{TP}$ with respect to $<_G$.*
2. *G is* traversable *if it has u-v $\mathcal{TP}$ for any $u, v \in V$.*
3. *G has $\mathcal{TP}$ from $u \in V$, if there exists a neighbor v of u such that G has u-v $\mathcal{TP}$.*

As we will see later, a black hole location algorithm with home base h is based on a $\mathcal{TP}$ from h. To achieve good complexity, we need the $\mathcal{TP}$ to have nice properties. Let $\pi = (\pi_a, \pi_b)$ be a u-v $\mathcal{TP}$ of G, and let G_i^a (resp. G_i^b) denote the subgraph of G induced

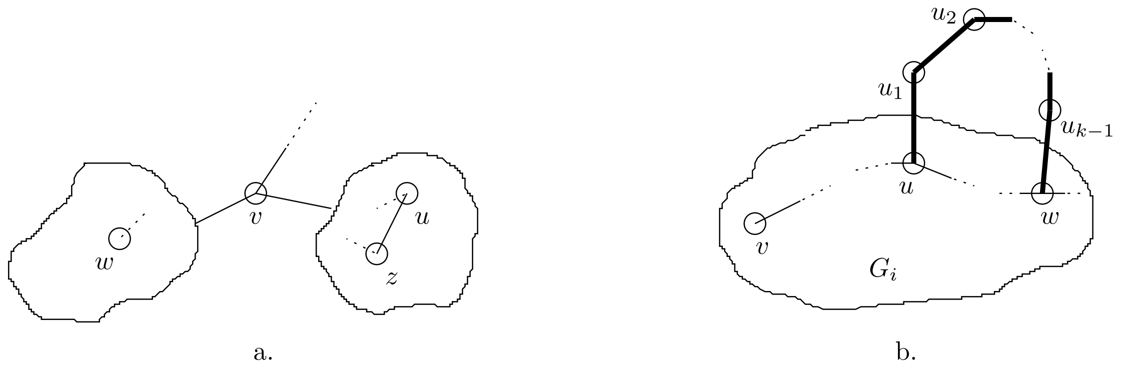

Figure 2: Proof of Lemma 3.1. (a) v is an articulation point of G. (b) The thick edges represent an "ear" in G_i.

by vertices $v_1, v_2, \ldots, v_i$ (resp. $v_n, v_{n-1}, \ldots, v_i$). Moreover, let $r(G_i^a)$ (resp. $r(G_i^b)$) be the depth of the breadth first search tree of G_i^a (resp. G_i^b) rooted at v_1 (resp. v_n).

Definition 3.3 (Size and Radius). *The* size *of* π *is* $s_\pi(G) = \max\{|\pi_a|, |\pi_b|\}$, *i.e., the maximum of the lengths of walks* π_a *and* π_b. *The* radius *of* π *is* $r_\pi(G) = \max_i(\max\{r(G_i^a), r(G_i^b)\})$.

Note that, if a graph G has an Hamiltonian circuit, then G is traversable, with $s_\pi(G) \leq n$ and $r_\pi(G) \leq n$.

The following lemma shows that our notion of a traversable graph actually coincides with biconnectivity. Since a black hole can always be located if and only if the network is biconnected [7], we do not lose anything by focusing only on graphs with a $\mathcal{TP}$.

Lemma 3.1. *A graph G is traversable if and only if it is biconnected.*

Proof. To show the "only if" direction consider a traversable graph G. By contradiction, let v be an articulation point of G (i.e., a node whose removal disconnects the graph). If every component of $G - \{v\}$ contains only one vertex then G is a star and there is no $\mathcal{TP}$. So let us consider a vertex $u \neq v$ such that there are at least two vertices in the component of $G - \{v\}$ containing u and choose z be a neighbor of u in the component of $G - \{v\}$. There exists a $\mathcal{TP}$ in G between any pair of vertices, so let us choose an u-z $\mathcal{TP}$ π. Since v is an articulation point, there is some vertex w which is in a different component of $G - \{v\}$ than u (refer to the example depicted in Figure 2.a). Clearly, $\pi_a \equiv u = v_1, v_2, \ldots, v, \ldots, w, \ldots, v_n = z$, and $\pi_b \equiv z = v_n, v_{n-1}, \ldots, v, \ldots, w, \ldots, v_1 = u$; thus, in both π_a and π_b, v is explored before w: a contradiction.

The "if" direction is shown by an inductive construction. Consider a biconnected graph G and an edge $(v, z) \in G$. Let G_i denote the induced subgraph of G for which a $\mathcal{TP}$ has been already constructed, where $G_1 = \{v, z\}$. If $G_i = G$ we are done; otherwise, take a $\mathcal{TP}$ π^i for G_i. Consider π_a^i. Let u be the first appearance of a vertex with a neighbor not in G_i. Since G is biconnected, there exists an "ear" – a path $u = u_0, u_1, \ldots, u_k = w$ such that $u, w \in G_i$ and $u_j \notin G_i$ for $1 \leq j < k$ (see Figure 2.b). The existence of an ear follows readily from the biconnectivity: u has one neighbor in G_i and another outside G_i and there is a circle containing both of them. Part of this circle outside G_i forms an ear. Moreover, as u was chosen to be the first appearance of a vertex with a neighbor outside G_i in π_a^i, it must be the case that w is explored after u in π_a^i. Given an ear, we extend the $\mathcal{TP}$ π^i as follows. After the first occurrence of u in π_a^i we insert the sequence $u_1, u_2, \ldots, u_{k-1}, u_{k-2}, \ldots, u_1, u$ and the rest of π_a^i is left unchanged. With π_b^i, the situation is somewhat different: just before the first occurrence of u the walk π_b^i returns to w using only vertices already visited by π_b^i. Then the sequence $u_{k-1}, \ldots, u_1, u$ is inserted and the rest stays unchanged. □

The following notation will be used in the rest of the paper. We will denote by $V[i,j]$ for $i < j$ the set of nodes $\{v_i, v_{i+1}, \ldots, v_j\}$. $G_{\widehat{ij}}$ is the graph induced by the nodes in $V \setminus V[i,j]$. The segment of π_a (resp. π_b) between the first occurrences of v_i and v_j will be denoted by $\pi_a[i,j]$ (resp. $\pi_b[i,j]$).

3.3 Algorithm Presto

We are now ready to present and analyze a size-optimal Bhs protocol Presto. The algorithm uses a traversal pair $\mathcal{TP}$, which has two main functions: it will indicate the order in which the network must be explored by the agents, and will be used by the agents to avoid "dangerous" parts of the network.

The two agents, a and b, start from the same node $v_0 = h$; a $\mathcal{TP}$ (π_a, π_b) of G from v_0 is available to both. The algorithm proceeds in logical rounds. In each round, the agents follow the cooperative approach of dynamically dividing the work between them: the unexplored area is partitioned into two parts of (almost) equal size. Each agent explores one part without entering the other one; exploration and avoidance are directed by the traversal pair. Since the parts are disjoint, one of them does not contain the black hole and the corresponding agent will complete its exploration. When this happens, the agent (reaches the last safe node visited by the other agent and there) partitions whatever is still left to be explored, leaving a note for the other agent (should it be still alive). This process is repeated until the unexplored area consists of a single node: the black hole.

At any time, an agent will be either exploring its part of the network, or searching for the other agent to perform another partition, or destroyed by the black hole.

We remind that a node is safe if there is a safe link incident to it, or if it is the *home base* of an agent. The safe nodes represent the explored part of the network. Let U be the set of unexplored nodes, and p be the node where the partition occurs. Initially, $U = V[1, n-1]$, and $p = v_0$.

Presto

Start -

1. Initially, one of the two agents, say a, partitions $V[1, n-1]$ into two sets $V_a[1,k]$ and $V_b[k+1, n-1]$, where $k = \lfloor n/2 \rfloor$.
2. Agents a and b leave v_0 to explore the corresponding sets, using cautious walk on $\pi_a[1,k]$ and $\pi_b[k+1, n-1]$, respectively. Note that, since V_a and V_b do not overlap, one of them does not contain Bh, and the corresponding agent will finish its exploration.
3. When the agent completes the exploration, it *searches* for the other agent to compute the new partition. In general, let $U = V[i,j]$ be the unexplored area when the exploration began (initially, $U = V[1, n-1]$). All operations on indices are modulo n.

Searching for the other agent -

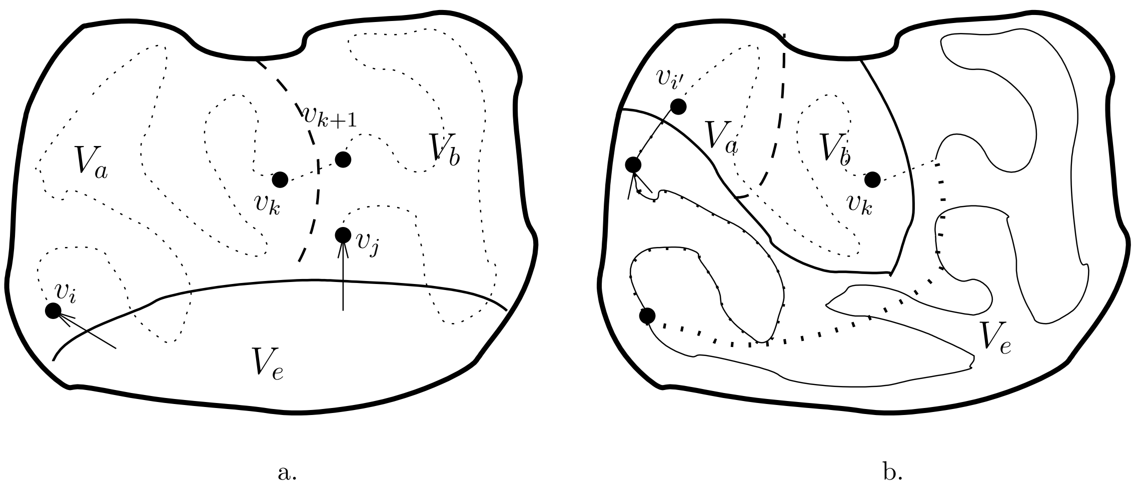

Figure 3: Algorithm Presto. (a) Beginning of a round. (b) After finishing its part, the agent b finds the node from which agent a departed to $v_{i'}$ and the remaining unexplored part is divided into new V_a and V_b.

1. If a is searching for b: a goes to v_{j+1} (the node from which b departed towards its unexplored part) using the shortest possible route avoiding V_b. It then follows the safe links of the path $\pi_b[j, k+1]$ until it reaches the last safe vertex p reached by b. Let $v_{j'}$ be the vertex to which b has departed. Then now $U = V[k+1, j']$. Agent a *computes* the new partitions of U.
2. If b is searching for a: b goes to v_{i-1} (the node from which a departed towards its unexplored part) using the shortest possible route avoiding V_a. It follows the safe links of the path $\pi_a[i, k]$ until it reaches the last safe vertex p reached by a. Let $v_{i'}$ be the vertex to which a has departed. Then, now $U = V[i', j]$. Agent b *computes* the new partitions of U.

Partitioning $U = V[f, l]$ at p -

1. The agent performing the partition sets $V_a = V[f, k']$ and $V_b = V[k'+1, l]$, where $k' = \lfloor (f+l)/2 \rfloor$ (see Figure 3);
2. it then writes at p a note informing the other agent of the partition, and leaves to *explore* its assigned set.
3. If the other agent finds the note informing it of the new partitions V_a and V_b, it will *reach* and *explore* the new assigned part.

Reaching and Exploring the Partition -

1. If a is the agent moving towards its partition, it returns to v_{f-1} using the shortest possible route avoiding the new V_b; it then departs towards v_f and starts exploring V_a using cautious walk on $\pi_a[f, k']$.
2. If b is the agent moving towards its new partition, it returns to v_{l+1} using the shortest possible route avoiding the new V_a; it then departs towards v_l and starts exploring V_b using cautious walk on $\pi_b[l, k'+1]$.

3. When an agent completes the exploration of its part, it will search for the other agent to compute the new partition.

Termination - When computing the new partition, if U contains a single node, that node is the black hole.

Theorem 3.1. *Let a graph G have a $\mathcal{TP}$ π from h of size $s_\pi(G)$ and radius $r_\pi(G)$. Then two agents placed at h can locate the black hole in G using $O(s_\pi(G) + r_\pi(G)\log n)$ moves.*

Proof. Correctness. Note that from the definition of $\mathcal{TP}$ and the way the algorithm works, it never happens that two agents depart to the same non-safe node. In fact, the algorithm uses the $\mathcal{TP}$ to be able to safely explore V_a without wandering into V_b, and vice versa. $\mathcal{TP}$ allows us to specify in a unified format the way V_a and V_b are explored, regardless of the actual values of V_a and V_b, which depend of the specifics of the particular execution. This means that one agent will always survive. The fact that the agents never wait ensures progress of the algorithm. Since in each round the number of the unexplored nodes is halved, after $\log n$ rounds there is a single unexplored node and the algorithm terminates.

Complexity. We now focus on the number of moves. The time complexity cannot be higher, and since there are only 2 agents, neither it could be asymptotically lower.

In each round, the only steps of the algorithm when agents move are to

1. Explore the assigned area (without loss of generality, we assume that b explored whole V_b, while a explored only part of V_a).
2. Move to the "beginning" of V_a.
3. Chase the other agent through the newly explored area.
4. Move to the "starting" node for the next round.

Let v_{bh} be the node containing the black hole. Note that the total exploration path performed by agent a during Step (1) over all rounds is at most $|\pi_a[1, bh]|$ (the bound is $|\pi_b[bh, n] + 1|$ for b). Clearly, the total cost of Step (1) over all rounds is less then $2s_\pi(G)$. Using similar arguments, the same bound holds also for the total cost of Step (3).

The cost of Steps (2) and (4) for one agent in one round is clearly bound by $2r_\pi(G)$.

Combining with the fact that there are at most $\lceil \log n \rceil$ rounds results in $O(s_\pi(G) + r_\pi(G)\log n$ bound on the number of moves. □

4 Traversal Pair Construction and Properties

4.1 $\mathcal{TP}$ Construction

In this subsection we present a technique for construction of $\mathcal{TP}$ based on hierarchical decomposition of the graph, making use of the $\mathcal{TP}$s of the graph's components.

Let $H = (V_H, E_H)$ be a biconnected graph with $|V_H| = k$; let $\pi_H = (\pi_a^H, \pi_b^H)$ be a traversal pair of H.

Let $F_1 = (V_1, E_1)$, $F_2 = (V_2, E_2)$, ..., $F_k = (V_k, E_k)$ be a set of (traversable) biconnected graphs. Let us denote by $s(F_i)$ the maximal size among all $\mathcal{TP}$s (between any pair of nodes) of F_i, and by $r(F_i)$ the maximal radius among all $\mathcal{TP}$s of F_i. Moreover, define $r(F) = \max_{i=1}^k(r(F_i))$. Let $d(G)$ denote the diameter of a graph G and let $d(F) = \max_{i=1}^k(d(F_i))$.

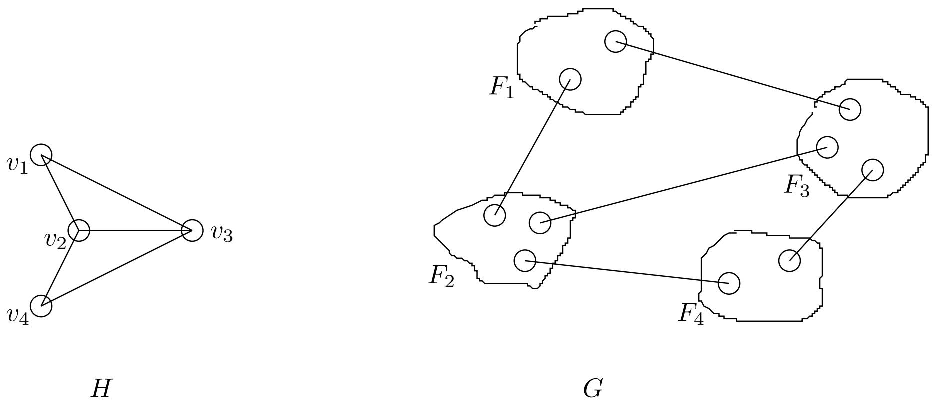

Figure 4: G is a $\mathcal{TP}$-composition of H and F_1, F_2, F_3 and F_4.

Definition 4.1. *We say that $G = (V, E)$ is a $\mathcal{TP}$-composition of H and $F_1, F_2, \ldots, F_k$ if and only if the following holds:*

1. *$V = \cup_{i=1}^{k} V_i$ and $\cup_{i=1}^{k} E_i \subset E$.*
2. *If $(v_i, v_j) \in E_H$ then there exists an edge $(u_i, u_j) \in E$ such that $u_i \in V_i$ and $u_j \in V_j$.*
3. *Let, for all $2 \leq i < k$, v_{a_i} and v_{b_i} be the nodes from which v_i is for the first time visited in π_a^H and π_b^H, respectively. Then, there are two different nodes $w, z \in V_i$ such that w has a neighbor in V_{a_i} and z has a neighbor in V_{b_i}.*

Moreover, if $\forall\ (v_i, v_j) \in E_H$, and $\forall\ u \in V_i$, $\exists w \in V_j$ such that the distance from u to w is less than or equal to c, we say that G has dilation *c.*

Informally, the $\mathcal{TP}$-composition of H and $F_1, F_2, \ldots, F_k$ is obtained by replacing a vertex v_i of H by graph F_i; the connectivity requirements are designed to allow the $\mathcal{TP}$ of H to be extended to the $\mathcal{TP}$ of G (refer to the example depicted in Figure 4).

Lemma 4.1. *Let $G = (V, E)$ be a $\mathcal{TP}$-composition of H and $F_1, F_2, \ldots, F_k$. Then G has a $\mathcal{TP}$ π_G from any vertex $u \in V_1$ with a neighbor in V_k, such that*

$$s_{\pi_G}(G) \leq (d(F) + 1) s_{\pi_H}(H) + \sum_{i=1}^{k} s(F_i),$$

and radius

$$r_{\pi_G}(G) \leq r(F) + (d(F) + 1) r_{\pi_H}(H).$$

Moreover, if G has dilation c, then

$$s_{\pi_G}(G) \leq c \cdot s_{\pi_H}(H) + \sum_{i=1}^{k} s(F_i), \qquad r_{\pi_G}(G) \leq r(F) + c \cdot r_{\pi_H}(H).$$

Finally, if G has dilation 1, then

$$r_{\pi_G}(G) \leq \max\{r_{\pi_H}(H) + d(F_1), r(F_1)\}.$$

Proof. The proof is constructive. In fact we now show how to build π_a^G (the left traversal of G); π_b^G is constructed analogously.

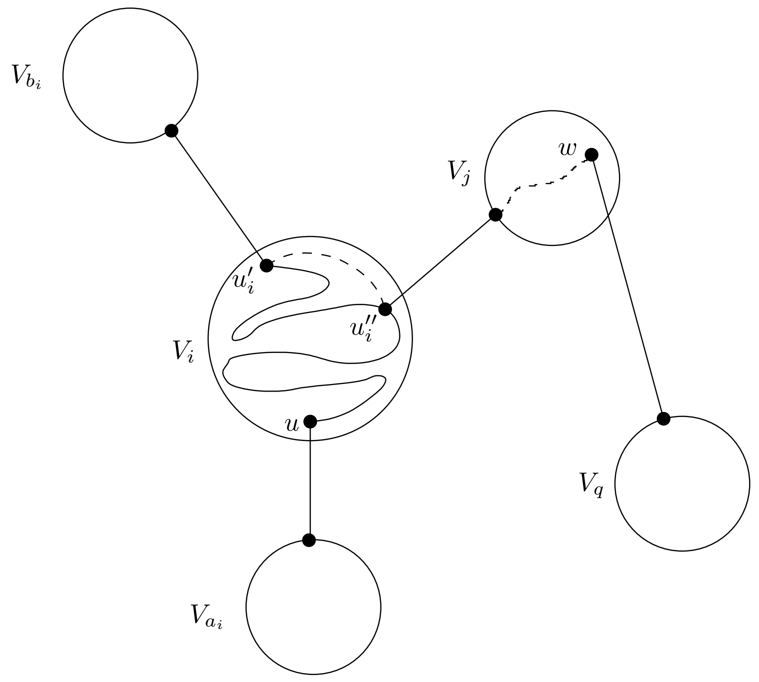

Figure 5: Extending the traversal in F_i.

Assume the vertices of H are ordered $v_1, \dots, v_k$ according to when they were explored by π_a^H. Consider the moment when π_a^G has arrived for the first time to a node u of V_i (the following works also for $i = 1$). That corresponds to the first occurrence of v_i in π_a^H. Let v_{a_i} (resp. v_{b_i}) be the node from which v_i was reached the first time in π_a^H (resp. π_b^H), and let v_j be the node following the first occurrence of v_i in π_a^H. Clearly $j \leq i + 1$, and $j = i + 1$ exactly if v_j has not yet been explored by π_a^H. Let u'_i be any node of V_i different from u which has a neighbor in V_{b_i} (from the third point of Definition 4.1 we know that such node must exist) and u''_i be any node of V_i which has a neighbor in V_j.

We extend π_a^G from u first with a u-u'_i traversal of F_i (in $\pi_a^{F_i}$), then with a path from u'_i to u''_i (if $u'_i \neq u''_i$), and finally with the edge that leads from u''_i to V_j (see Figure 5). This way π_a^G explores the vertices of F_i in the order of $\pi_a^{F_i}$, thus allowing a symmetrical construction of π_b^G by using a u'_i-u traversal of F_i (in $\pi_b^{F_i}$).

If $j = i+1$, the last added edge from u''_i entered a V_j containing only vertices unexplored by π_a^G so far, and the process of extending π_a^G continues. If, on the other hand, $j \neq i+1$, the node v_j has already been visited in π_a^H, which also means that all nodes in V_j have already been visited in π_a^G. In this case, let v_q be the node in π_a^H after v_j, and let w be a node from V_j which has a neighbor in V_q. We extend π_a^G by first adding the shortest path (in F_j) leading to w, and then by adding the link that leads to V_q.

Size: Exploring a component F_i for the first time costs $s(F_i)$; the path from u'_i to u''_i, if needed, costs at most $d(F_i) < d(F)$; finally, the edge added to reach the next component costs 1. Each next occurrence of v_i in π_a^H (resp. $\pi_b(H)$) corresponds to an additional traversing of F_i of length at most $d(F)$. Summing up over the whole length of π_a^H (resp. $\pi_b(H)$) produces the result. If G has dilation c, then all traversing can be done with at most c links (including the link for reaching to the next component).

Radius: Consider a node $w \in V_i$, and let $u_i \in V_i$ be the first node of V_i in π_a^G (the case for π_b^G is analogous). The distance between w and u_i in F_i is at most $r(F_i) \leq r(F)$. Consider now the nodes in π_H between v_1 and v_i. There are at most $r_{\pi_H}(H)$ edges in the path from v_i to v_1 using only those nodes. Since in π_G each edge is replaced by a path of length at most $1 + d(F)$, the total distance between w and u (the first node in π_G^a) results in $r(F) + (d(F) + 1)r_{\pi_H}(H)$.

If G has dilation c then the term $d(F)$ can be replaced by $c - 1$. If G has dilation 1 then each node of F_j is connected to all neighboring components. Hence, we can reach F_1 from w with a path of length at most $r_{\pi_H}(H)$. Once F_1 is reached, we still need to reach u_1; hence the total length is $r_{\pi_H}(H) + d(F_1)$.

Note that, if $i = 1$, F_1 is not yet fully explored; hence the distance between w and the first node in π_G^a is simply $r(F_1)$. □

Quite often a more limited composition will be sufficient:

Definition 4.2. *We say that G is* uniform *$\mathcal{TP}$-composition of H and F, if G is $\mathcal{TP}$ composition of H and $F_1, F_2, \ldots, F_r$, with $F = F_i$ for all $1 \leq i \leq r$.*

Directly applying Lemma 4.1 yields:

Corollary 4.1. *Let G be a uniform $\mathcal{TP}$-composition of H and F such that $s_\pi(F) \leq c|F|$ for some constant c. If there is a $\mathcal{TP}$ for H of size $s_\pi(H) \leq q|H|$ where $q \geq 2c$ then there is a $\mathcal{TP}$ for G of size $s_\pi(G) \leq q(|G| + |H|)$.*

Proof. Lemma 4.1 bounds the size of $\mathcal{TP}$ for G to be at most $(d(F) + 1)s_\pi(H) + ks_\pi(F)$. Since F is biconnected it holds $d(F) \leq |F|/2$ yielding $s_\pi(G) \leq q|H|(|F|/2 + 1) + kc|F|$. As $|H||F| = k|F| = |G|$ we get $s_\pi(G) \leq q(|G| + |H|)$. □

4.2 Traversal Pairs for Specific Topologies

Lemma 4.1 and Corollary 4.1 can be used to find good traversal pairs in a number of graphs.

Lemma 4.2. *Let G be a d-dimensional torus with n vertices and diameter $diam(G)$. Then G is traversable with a $\mathcal{TP}$ of size at most $4n$ and radius $diam(G)$.*

Proof. We denote $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ and ε_i the vector with a single 1 at position i. Torus is a Cayley graph over a group $\mathbb{Z}_{dim_1} \times \cdots \times \mathbb{Z}_{dim_d}$ where all $dim_i > 2$, with generators $\pm\varepsilon_i$. As Cayley graphs are vertex transitive, it is sufficient to show the existence of a $\mathcal{TP}$ from one vertex.

If $d = 1$ then G is a cycle and there is a traversal of size $2n$ and radius $n/2 = diam(G)$. Now consider a d-dimensional torus for $d > 1$. W.l.o.g we may assume $dim_1 \leq dim_2 \leq \cdots \leq dim_d$. The diameter of G is $\frac{1}{2}\sum_{i=1}^{d} dim_i$. Let F be a cycle in the first dimension, i.e. of length dim_1. F is biconnected and traversable with $s_\pi(F) = 2dim_1$ and $r_\pi(F) \leq dim_1$.

We show that G has a uniform $\mathcal{TP}$-composition with dilation 1 of H and F where H is a $(d-1)$-dimensional torus with dimensions $dim_2, \ldots, dim_d$. The first two conditions in Definition 4.1 are trivial. The fact that G has dilation 1 follows directly from the commutativity of the group. Consider a vertex u in a set V_i. The set V_i consists of vertices $u + c\varepsilon_1$ for all c. If u has a neighbor in some other component V_k, say, $v = u + \varepsilon_j$ then every $u + c\varepsilon_1$ has

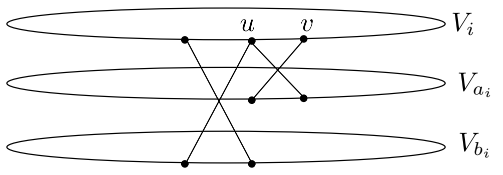

Figure 6: *WBF* is a uniform $\mathcal{TP}$-composition of a hypercube and a cycle.

neighbor $u + c\varepsilon_1 + \varepsilon_j = v + c\varepsilon_1$ in V_k. The third condition of Definition 4.1 is a consequence of G having a dilation 1.

We prove the bound on the size and radius of the $\mathcal{TP}$ by induction on the number of dimensions. The first step (a ring) is trivial. Following the induction hypothesis, H has a $\mathcal{TP}$ with size $s_\pi(H) = 4|H|$ and radius $r_\pi(H) = diam(H)$. Using Corollary 4.1 we conclude that G has a $\mathcal{TP}$ of size at most $4n$. To bound the diameter, we combine the fact that G has dilation 1 with Lemma 4.1 yielding $r_\pi(G) \leq \max\{r_\pi(H) + diam(F), r_\pi(F)\}$. Since $diam(F) = dim_1/2$, the term $r_\pi(H) + diam(F) = \frac{1}{2}\sum_{i=1}^{d} dim_i = diam(G)$. The result follows from the fact that $r_\pi(F) \leq dim_1 \leq dim_1/2 + dim_2/2$. □

Lemma 4.3. *Let G be a d-dimensional hypercube. Then G is traversable with a $\mathcal{TP}$ of size at most 2^{d+2} and radius d.*

Proof. It is the same as the proof of Lemma 4.2 with all $dim_i = 2$. This time, however, we set F to be a cycle of length four induced by the first two dimensions and then H is a $(d-2)$-dimensional hypercube. The $diam(F) = 2$, $r_\pi(F) \leq 4$ and $diam(H) = d - 2$. The basis of the induction are cases $d = 2$ and $d = 3$. □

Lemma 4.4. *Cube-connected cycles $CCC(d)$ and wrapped butterfly $WBF(d)$ are traversable with a $\mathcal{TP}$ of size $O(d2^d)$ and radius $O(d^2)$.*

Proof. It is sufficient to show that both topologies are uniform $\mathcal{TP}$-compositions of a d-dimensional hypercube with a cycle of length d. The size then comes from Corollary 4.1 and Lemma 4.3 and radius from Lemma 4.1 as $r_\pi(G) \leq r_\pi(F) + (diam(F) + 1)r_\pi(H) \leq d + (d/2 + 1)d$.

To prove the $\mathcal{TP}$-composition property consider the cycles corresponding to a particular hypercube vertex (i.e. induced by "shift" operations) in both topologies. The only nontrivial part to show is the condition 3 in Definition 4.1.

CCC: Consider a circle V_i in $CCC(d)$. Every vertex $v \in V_i$ has exactly one neighbor outside V_i, and any two distinct vertices in the circle V_i have their outside neighbors in different circles (corresponding to neighbors in the hypercube along appropriate dimensions). The condition 3 follows from the fact that $v_{l_i} \neq v_{r_i}$.

WBF(d): Condition 3 directly follows from the fact that in each circle V_i and V_j in $WBF(d)$ there are two different nodes $u, v \in V_i$ which have a neighbor in V_j (see Figure 6). □

Lemma 4.5. *The star graph $S(d)$ has a $\mathcal{TP}$ of size at most $3d!$ and radius at most $2^{d-1} + 1$.*

Proof. The star graph $S(d)$ is a Cayley graph over the symmetric group $\mathbb{S}_d$ generated by the involutions $(1,q)$ for $1 < q \leq d$. Let $S(k,d)$ be a Cayley graph over the coset group $\mathbb{S}_d|\mathbb{S}_k$ with generators $g \circ (1,q)[\mathbb{S}_k]$ where $g \in \mathbb{S}_k$ and $k < q \leq d$. Clearly, $S(1,d) = S(d)$ and $S(d-1,d) = K_d$ is a complete graph with d vertices. We can visualize the vertices of $S(k,d)$ as strings of length $d-k$ consisting of different symbols from the alphabet $\{1,2,\ldots,d\}$. The edges of $S(k,d)$ connect vertices which differ in exactly one place.

Now we show that $S(k,d)$, $2 \leq k < d-1$ is a uniform $\mathcal{TP}$-composition of $H = S(k+1,d)$ and $F = K_{k+1}$. We have to prove that the three conditions from Definition 4.1 are fulfilled. The first one is trivial. For the remaining two we show that if there is an edge $(v_i, v_j) \in E_H$, there are two pairs of vertices $(u_i, u_j) \in E$, $(u'_i, u'_j) \in E$ such that $u_i, u'_i \in V_i$ and $u_j, u'_j \in V_j$. Consider an edge $(v_i, v_j) \in E_H$ where $v_i = \alpha a \beta$, $v_j = \alpha c \beta$; here α, β stand for strings. As $k \geq 2$, there are two distinct symbols g, g' not present in α, β and different from q, c. Let $u_i = b\alpha a\beta$, $u'_i = g'\alpha q\beta$, $u_j = g\alpha c\beta$ and $u'_j = g'\alpha c\beta$. It is easy to see that $(u_i, u_j) \in E$ and $(u'_i, u'_j) \in E$.

As a next step we shall prove that $S(k,d)$, $2 \leq k < d-1$ has a $\mathcal{TP}$ of size at most $3d!/k!$ and radius $2^{d-k} - 1$. For $k = d-1$ the statement clearly holds. As $S(k,d)$ is a uniform $\mathcal{TP}$-composition of $S(k+1,d)$ and K_{k+1} we get the size and the radius from Lemma 4.1, as $s_{\pi_G}(G) \leq (d(F)+1)s_{\pi_H}(H) + \sum_{i=1}^{k} s(F_i) = 2s_{\pi_H}(H) + \frac{n}{k+1}(k+1) \leq \frac{d!}{k!}\sum_i \left(\frac{2}{k+1}\right)^i$ and $r_\pi(G) \leq r_\pi(F) + (d(F)+1)r_\pi(H) = 1 + 2 \cdot (2^{d-k-1} - 1) = 2^{d-k} - 1$.

In order to finish the proof we have to bridge the gap between $S(d) = S(1,d)$ and $S(3,d)$. Similar arguments as above lead to conclusion that $S(d)$ is a uniform $\mathcal{TP}$-composition of $S(3,d)$ and a circle of length 6 and the result follows. □

Lemma 4.6. *Let G be a d-dimensional mesh with n vertices and diameter $diam(G)$. Then G is traversable with a $\mathcal{TP}$ of size at most $4n$ and radius $diam(G)$.*

Proof. By induction on the number of dimensions. The basis of induction are cases $d = 2$ and $d = 3$. The $\mathcal{TP}$ for a $2D$ mesh is depicted in Figure 1. Its size is n for a mesh with at least one side even, and $n+2$ for all sides odd. It is not difficult to see that the radius of this $\mathcal{TP}$ is no more that $diam(G) + 1$.

The $\mathcal{TP}$ for a $3D$ mesh is depicted in Figure 7. The left traversal starts by going to the topmost $2D$ sub-mesh using the $(0,0)$ column. The $2D$ meshes are then traversed from the top to the bottom using a left traversal for $2D$ mesh from $(0,1)$, ending at $(1,0)$, returning to $(0,1)$ and going down. The right traversal traverses $2D$ meshes from the bottom to the top, and returns by the $(0,0)$ column. Each $2D$ mesh is traversed using right traversal for $2D$ mesh starting at $1,0)$ and ending at $(0,1)$, then returning to $(1,0)$ and moving one level up. Again, it is easy to see that the size of this $\mathcal{TP}$ is $O(n)$ and its radius is $O(diam(G))$.

A d dimensional mesh for $d \geq 4$ is a uniform $\mathcal{TP}$ composition of a $2D$ mesh F with a $d-2$ dimensional mesh H with dilation 1. The proof (as well as of the result bounds on its size and diameter) is analogous to the tori and hypercube case. □

4.3 Main Theorem

Combining Theorem 3.1 with the results of Section 4.2, yields the main theorem of this paper:

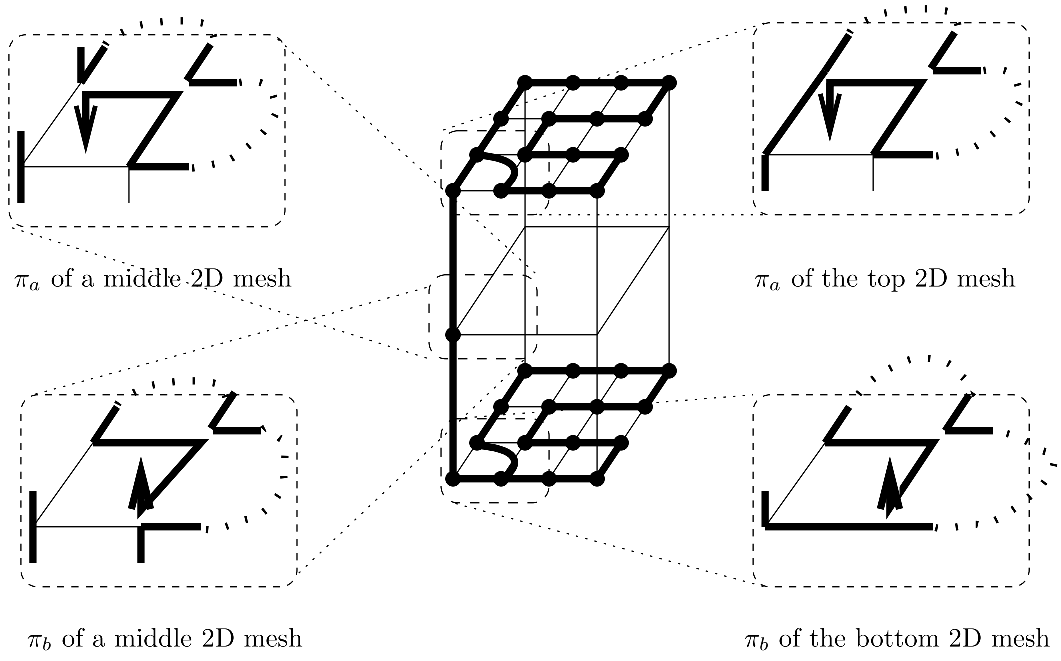
π_a of a middle 2D mesh
π_a of the top 2D mesh
π_b of a middle 2D mesh
π_b of the bottom 2D mesh

Figure 7: $\mathcal{TP}$ for a 3D mesh.

Theorem 4.1. *With complete topological knowledge,* two *agents can locate the black hole in* $O(n)$ *moves in the following topologies:*

1. *hypercubes,*
2. *CCC,*
3. *wrapped butterflies,*
4. *star graphs, and*
5. *tori and meshes of diameter* $O(n/\log n)$.

5 Relaxing the Knowledge Requirements

In deriving our results, we have assumed that the agents have complete topological knowledge; that is, the agents know not only the network topology type and labelling (e.g., torus with "N-S-E-W" labelling), but also the actual size n of the network and the location of the home base.

This requirement is somewhat stronger than the assumptions typically used in related literature, i.e. only the network topology type, not its size, is known to the agents.

In this section we show that our assumptions can be relaxed to match the standard model, by showing how to compute the network size and the location of the home base for the class of the networks considered. This is achieved by adding a precomputation phase, in which agents compute the size of the network and the location of the home base (knowledge of the topology class e.g. CCC, or mesh is still assumed, as well as a knowledge of globally consistent labelling, e.g. being able to distinguish between cycle and hypercube edges in CCC, or identify north, east, south, west in a 2-dimensional mesh).

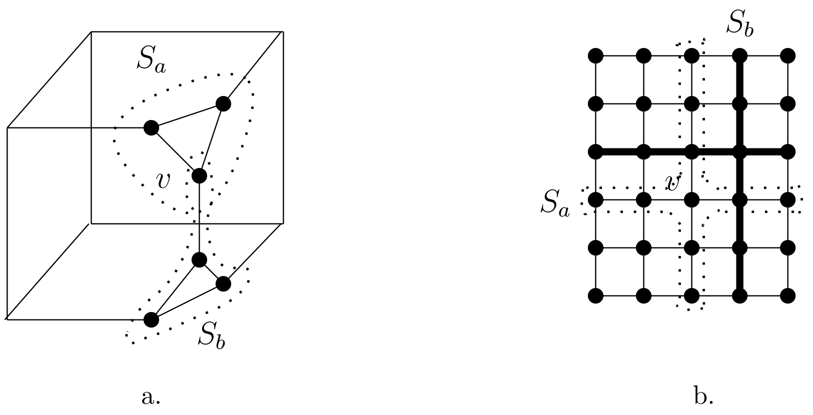

Figure 8: Left: S_l and S_r in a CCC. Right: S_l and S_r in a two dimensional mesh.

For vertex symmetric topologies (tori, hypercubes, CCC, wrapped butterflies and star graphs), the problem of identifying the location of the home base is irrelevant, as all nodes are alike. For hypercubes and star graphs the size immediately follows from the degree of nodes; in tori and meshes the number of dimensions can be determined in that way.

We present a general scheme for determining n and location of the home base (if relevant) for CCC and 2-dimensional meshes, the extensions to wrapped butterflies and multidimensional meshes and tori are quite straightforward.

The general scheme

1. Choose two disjoint sets of vertices in G: S_a and S_b such that $S_a \cap S_b = \{v\}$ (v is the home base) and it is possible to determine the size of the network (and the location of the home base, if needed) from each of them independently. See Figure 8, left.
2. If no such sets can be found, explore some neighborhood S' of v in a way that at least one agent survives. S' is chosen such that for every $|S'| - 1$ node subset S'' of S' there exist S_a and S_a such that $S_a \cap S_a \subset S''$ and n and location of v can be determined from each of them. See Figure 8, right, for an example for a 2-dimensional mesh: S' consists of the four direct neighbors of v. The cross S_b is chosen to intersect S_a in two neighbors of v which are known to be safe.
3. The agents a and b explore S_a and S_b, respectively, and return to the home base. The way S_a and S_b were chosen ensures that at least one of them (w.l.o.g. assume that b) succeeds.
4. b goes to the last safe node visited by a and leaves a mark with the meaning "Stop exploring S_a, a already know n and location of v. Join me in Algorithm PRESTO." and starts executing Algorithm PRESTO.
5. Let v_i be the node to which a was travelling when b left the message for it. The first assignment of V_a and V_b will not be $V[1..\lfloor n/2 \rfloor]$ and $V[\lfloor n/2 \rfloor + 1, n-1]$, but $V[1..i-1]$ and $V[i+1..n-1]$. Furthermore, if a is still blocked at i when b finishes its part, b will "switch" with a (i.e. b will start exploring from the left, while a will be asked to explore from the right). This prevents both agents disappearing in i if the black hole is there.

Note that the cost of such precomputation is $O(|S'|+|S_1|+|S_2|)$, which is for all relevant topologies $O(n)$.

6 Conclusions

We have presented a novel concept, traversal pairs of a biconnected graph, and shown how to use it to obtain a *size-optimal* black hole searching technique. We have shown that this technique leads to solutions which are also *cost-optimal* for all the common interconnection networks.

The outstanding open question is to determine for what other types of networks $\Theta(n)$ cost can be achieved by two searching agents.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, and by "Progetto ALINWEB: Algoritmica per Internet e per il Web", MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale.

References

[1] S. Alberts and M. R. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29:1164–1188, 2000.

[2] W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific dataset. *J. of Network and Computer Applications*, 2002.

[3] E. Arkin, M. Bender, S. Fekete, and J. Mitchell. The freeze-tag problem: how to wake up a swarm of robots. In 13th *ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 568–577, 2002.

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

[5] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In 11th *Annual European Symposium on Algorithms (ESA '03)*, pages 184–195, 2003.

[6] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. In 13th *ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 588–597, 2002.

[7] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile agents searching for a black hole in an anonymous ring. In *Proc. of 15th Int. Symposium on Distributed Computing (DISC 2001)*, pages 166–179, 2001.

[8] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Finding a black hole in an arbitrary network: optimal mobile agents protocols. In *Proc. of 21st ACM Symposium on Principles of Distributed Computing (PODC 2002)*, pages 153–162, 2002.

[9] P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. In 6th *Latin American Theoretical Informatics Symposium 2004*, 2004. to appear.

[10] P. Fraigniaud and D. Ilcinkas. Directed graph exploration with little memory. In 21st *Symposium on Theoretical Aspects of Computer Science (STACS 2004)*, 2004. to appear.

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

[12] N. Hanusse, D. Kavvadias, E. Kranakis, and D. Krizanc. Memoryless search algorithms in a network with faulty advice. In 2nd *IFIP International Conference on Theoretical Computer Science (TCS '02)*, pages 206–216, 2002.

[13] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In *Proc. of Conf on Mobile Agent Security*, LNCS 1419, pages 92–113, 1998.

[14] F. Hohl. A framework to protect mobile agents by using reference states. In *Proc. of the 20th Int. Conf. on Distributed Computing Systems (ICDCS 2000)*, 2000.

[15] L. M. Kirousis, E. Kranakis, D. Krizanc, and Y. Stamatiou. Locating information with uncertainty in fully interconnected networks. In 14th *International Symposium on Distributed Computing (DISC 2000)*, LNCS 1914, pages 283–296, 2000.

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

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

[18] T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. In *Proc. of Conf on Mobile Agent Security*, LNCS 1419, pages 44–60, 1998.

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

[20] S.K.Ng and K.W. Cheung. Protecting mobile agents against malicious hosts by intention spreading. In *Proc. 1999 Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 725–729, 1999.

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

[22] X. Yu and M. Yung. Agent rendezvous: A dynamic symmetry-breaking problem. In 23rd *International Colloquium on Automata, Languages, and Programming (ICALP '96)*, LNCS 1099, pages 610–621, 1996.