Distributed Security Algorithms by Mobile Agents

Paola Flocchini[1] and Nicola Santoro[2]

[1] University of Ottawa
flocchin@site.uottawa.ca
[2] Carleton University
santoro@scs.carleton.ca

Abstract. Mobile Agents have been extensively studied for several years by researchers in Artificial Intelligence and in Software Engineering. They offer a simple and natural way to describe distributed settings where mobility is inherent, and an explicit and direct way to describe the entities of those settings, such as mobile code, software agents, viruses, robots, web crawlers, etc. Further, they allow to express immediately notions such as selfish behaviour, negotiation, cooperation, etc arising in the new computing environments. As a programming paradigm, they allow a new philosophy of protocol and software design, bound to have an impact as strong as that caused by that of object-oriented programming. As a computational paradigm, mobile agents systems are an immediate and natural extension of the traditional message-passing settings studied in distributed computing.

In spite of all this, mobile agents systems have been largely ignored by the mainstream distributed computing community. It is only in the last few years that several researchers, some motivated by long investigated and well established problems in automata theory, computational complexity, and graph theory, have started to systematically explore this new and exciting distributed computational universe.

In this paper we describe some interesting problems and solution techniques developed in this investigations.

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.

In networked systems that support autonomous mobile agents, a main concern is how to develop efficient agent-based *system protocols*; that is, to design protocols that will allow a team of identical simple agents to cooperatively perform (possibly complex) system tasks. Example of basic tasks are *wakeup*, *traversal*, *rendez-vous*, *election*. The coordination of the agents necessary to perform these tasks is not necessarily simple or easy to achieve. In fact, the computational problems related to these operations are definitely non trivial, and a great deal of theoretical research is devoted to the study of conditions for the solvability of these problems and to the discovery of efficient algorithmic solutions; e.g., see [1,2,4,5,6,7,17,18,20,45].

At an abstract level, these environments can be described as a collection of autonomous mobile *agents* (or *robots*) located in a graph G. The agents have computing capabilities and bounded storage, execute the same protocol, and can move from node to neighboring node. They are *asynchronous*, in the sense that every action they perform (computing, moving, etc.) takes a finite but otherwise unpredictable amount of time. Each node of the network, also called *host*, provide a storage area called *whiteboard* for incoming agents to communicate and compute, and its access is held in fair mutual exclusion. The research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost.

At a practical level, in these environments, *security* is the most pressing concern, and possibly the most difficult to address. Actually, even the most basic security issues, in spite of their practical urgency and of the amount of effort, must still be effectively addressed (for a survey, see [50]).

Among the severe security threats faced in distributed mobile computing environments, two are particularly troublesome: *harmful agent* (that is, the presence of malicious mobile processes), and *harmful host* (that is, the presence at a network site of harmful stationary processes).

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. In fact, a local (hardware or software) failure might render a host harmful. In this paper we concentrate on two security problems, one for each type: *locating a black hole*, and *capturing an intruder*.

2 Black Hole Search

2.1 The Problem and the Model

The problem posed by the presence of a harmful host has been intensively studied from a programming point of view (e.g., see [41,54,56]). 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 which *disposes* of visiting agents upon their arrival, leaving *no observable trace* of such a destruction. Note that this type of highly harmful host is not rare; for example, the undetectable crash failure of a site in a asynchronous network turns such a site into a black hole. The task is to unambiguously determine and report the location of the black hole by a team of mobile agents. One can easily see that the problem can also be formulated as an exploration problem. In fact, the black hole can be located only after the whole network has been visited, and all nodes but one are found to be safe. Clearly, in this process some agents have disappeared in the black hole). The searching agents start from the same safe site (the homebase); the task is successfully completed if, within finite time, at least one agent survives, and all surviving agents know 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, called *Black-Hole Search*. The main complexity measures for this problem are: the *size* of the solution (i.e., the number of agents employed), the *cost* (i.e., the number of moves performed by the agents executing a size-optimal solution protocol). Sometimes also bounded *time* complexity is considered.

In general no assumptions are made on the time for an agent to move on a link, except that it is finite; i.e., the system is asynchronous. Moreover, agents communicate by writing and reading on whiteboards located at the nodes.

2.2 A Background Problem: Safe Exploration

The problem of exploring and mapping an unknown environment has been extensively studied in a *safe* environment, due to its various applications in different areas (navigating a robot through a terrain containing obstacles, finding a path through a maze, or searching a network).

Most of the previous work on exploration of unknown graphs has been limited to single agent exploration. Studies on exploration of *labelled* graphs typically emphasize minimizing the number of moves or the amount of memory used by the agent (e.g., see [1,17,19,51,52]). Exploration of *anonymous* graphs is possible only if the agents are allowed to mark the nodes in some way; except when the graph has no cycles (i.e. the graph is a tree [20,37]). For exploring arbitrary anonymous graphs, various methods of marking nodes have been used by different authors. Pebbles that can be dropped on nodes have been proposed first in [9] where it is shown that any strongly connected directed graph can be explored using just one pebble (if the size of the graph is known) and using $O(\log \log n)$ pebbles, otherwise. Distinct markers have been used, for example, in [29] to explore unlabeled undirected graphs. Yet another approach, used by Bender and Slonim [10] was to employ two cooperating agents, one of which would stand on a node, while the other explores new edges. Whiteboards have been used by Fraigniaud and Ilcinkas [38] for exploring directed graphs and by Fraigniaud et al. [37] for exploring trees. In [20,38,39] the authors focus on minimizing the amount of memory used by the agents for exploration (they however do not require the agents to construct a map of the graph).

There have been few results on exploration by more than one agent. A two agent exploration algorithm for directed graphs was given in [10], whereas Fraigniaud et al. [37] showed how k agents can explore a tree. In both these cases, the agents start from same node and they have distinct identities. In [7] a team of dispersed agents explores a graph and constructs a map. The graph is anonymous but the links are labeled with sense of direction; moreover the protocol works if the size n of the network or the number of agents k are co-prime and it achieves a move complexity of $O(km)$ (where m is the number of edges). Another algorithm with the same complexity has been described in [15], where the requirement of sense of direction is dropped. In this case the agents need to know either n or k, which must be coprime. The solution has been made "effective" in [16], where effective means that it will always terminate, regardless of the relationship between n and k reporting a solution whenever the solution can be computed, and reporting a failure message when the solution cannot be computed.

The map construction problem is actually equivalent to some others basic problems, like *Agent Election*, *Labelling* and *Rendezvous.* Among them rendezvous is probably the most investigated; for a recent account see [2,46].

2.3 Basic Properties for Black Hole Search

When considering the black hole search problem, some constraints follow from the asynchrony of the agents. For example [21]:

- For asynchronous agents to locate the black hole, G must be 2-node-connected.
- For asynchronous agents to locate the black hole, the number of nodes of G must be known.
- For asynchronous agents it is impossible to verify if there is a back hole.

Moreover, since one agent may immediately wander into the black hole, we have:

- At least two agents are needed to locate the black hole.

How realistic is this bound? How many agents suffice? The answers vary depending on the a priori knowledge the agents have about the network, and on the consistency of the local labelings.

2.4 Impact of Knowledge

Topological Ignorance. Consider first the situation of *topological ignorance*; that is when the agents have no a priori knowledge of the topological structure of G. Then any generic solution needs at least $\Delta + 1$ agents, where Δ is the maximal degree of G, even if the agents know Δ and the number n of nodes of G. Interestingly, in any *minimal* generic solution (i.e., using the minimum number of agents), the agents must perform $\Omega(n^2)$ moves in the worst case [23]. Both these bounds are *tight.* In fact there is a protocol that correctly locates the black hole in $O(n^2)$ moves using $\Delta + 1$ agents that know Δ and n [23]. The algorithm essentially performs a collective "cautious" exploration of the graph until all nodes but one are considered to be safe. The whiteboard on the homebase is used to store information about the nodes that have been already explored and the agents move back and forth from the homebase to continue their job. If the black hole is a node with maximum degree, there is nothing to prevent Δ agents disappearing in it.

Sense of Direction. Consider next the case of topological ignorance in systems where there is *sense of direction* (SD); informally, sense of direction is a labeling of the ports that allows the nodes to determine whether two paths starting from a node lead to the same node, using only the labels of the ports along these paths (for a survey on Sense of Direction see [34]). In this case, two agents suffice to locate the black hole, regardless of the (unknown) topological structure of G. The proof of [23] is constructive, and the algorithm has a $O(n^2)$ cost. This cost is optimal; in fact, it is shown that there are types of sense of direction that, if present, impose an $\Omega(n^2)$ worst-case cost on any generic two-agent algorithm for locating a black hole using SD. As for the topological ignorance case, the agents perform an exploration. The algorithm is similar to the one with topological ignorance (in fact it leads to the same cost); sense of direction is however very useful to decrease the number of casualties. The exploring agents can be only two: a node that is being explored by an agent is considered "dangerous" and by the properties of sense of direction, the other agent will be able to avoid it in its exploration, thus insuring that one of the two will eventually succeed.

Complete Topological Knowledge. Consider the case of *complete topological knowledge* of the network; that is, the agents have a complete knowledge of the edge-labeled graph G, the correspondence between port labels and the link labels of G, *and* the location of the source node (from where the agents start the search). This information is stronger then the more common *topological awareness* (i.e., knowledge of the class of the network, but not of its size nor of the source location – e.g. being in a mesh, starting from an unknown position).

Also in this case, two agents suffice [23]; furthermore the cost of a minimal protocol can be reduced in this case to $O(n \log n)$, and this cost is worst-case optimal. The technique here is quite different and it is based on a partitioning of the graph in two portions, which are given to the two agents to perform the exploration. One will succeed in finishing its portion and will carefully move to help the other agent finishing its own.

Topology-Sensitive Universal Protocols. Interestingly, it is possible to considerably improve the bound on the number of moves without increasing the team size. In fact, there is a recent *universal* protocol, *Explore and Bypass*, that allows a team of *two* agents with a map of the network to locate a black hole with cost $O(n + d \log d)$, where d denotes the diameter of the network [25].

This means that, without losing its universality and without violating the worst-case $\Omega(n \log n)$ lower bound, this algorithm allows two agents to locate a black hole with $\Theta(n)$ cost in a very large class of (possibly unstructured) networks: those where $d = O(n/ \log n)$.

Importantly, there are many networks with $O(n/logn)$ diameter in which the previous protocols [23,24] fail to achieve the $O(n)$ bound. A simple example of such a network is the *wheel*, a ring with a central node connected to all ring nodes, where the central node is very slow: those protocols will require $O(n \log n)$ moves.

Variations with Complete Topological Knowledge. A very simple algorithm that works on any topology (a-priori known by the agents) is shown in [27]. The algorithm, based on the pre-computation of an open vertex cover by cycles of the network, uses the optimal number of agents (two); its cost (number of moves) depends on the choice of the cover and it is optimal for several classes of networks. These classes include all Abelian Cayley graphs of degree three and more (e.g., hypercubes, multi-dimensional tori, etc,), as well as many non-Abelian cube graphs (e.g., CCC, butterfly, wrapped-butterfly networks, etc.). For some of these networks, this is the only algorithm achieving such a bound.

Using Tokens. Recently the problem has been investigated also in a different, weaker model where there are no whiteboards at the nodes but each agent has an identical token that the agent can place on (or remove from) a node [26,28]. Surprisingly, the black hole search problem can be solved also in this model. Furthermore, this can be done using a minimal team size and performing a polynomial number of moves; not surprisingly, the protocol is quite complex. Also the case of the ring has been studied in details in [28].

2.5 Special Topologies

A natural question to ask is whether the bounds for arbitrary networks with full topological knowledge can be improved for networks with special topologies by topology-dependent proptocols.

Rings. The problem has been investigated and its solutions characterized for *ring* networks [21]. A $Omega(n \log n)$ lower bound holds since $\Omega(n \log n)$ moves are needed by any two-agents solution [21]. An agent and move optimal solution exists, based on a partitioning of the ring and on a non-overlapping exploration by the agent. There exists an optimal trade-off between time complexity and number of agents. In fact, increasing the number of agents the number of moves cannot decrease, but the time to finish the exploration does [21]. Notice that the lower bound for rings implies an $\Omega(n \log n)$ lower bound on the worst case cost complexity of any *universal* protocol.

The ring has been investigated also to perform another task: *rendezvous* of k anonymous agents, in spite of the presence of a black hole. The problem is studied in [22] and a complete characterization of the conditions under which the problem can be solved is established. The characterization depends on whether k or n is unknown (at least one must be known for any non-trivial rendezvous). Interestingly, it is shown that, if k is unknown, the rendezvous algorithm also solves the black hole location problem, and it does so with a bounded time complexity of $\Theta(n)$; this is a significant improvement over the $O(n \log n)$ bounded time complexity of [21].

Interconnection Networks. The negative result for rings does not generalizes. Sometimes the network has special properties that can be exploited to obtain a lower cost network-specific protocol. For example, two agents can locate a black hole with only $O(n)$ moves in a variety of highly structured interconnection networks such as *hypercubes*, square *tori* and *meshes*, *wrapped butterflies*, *star graphs* [24]. These strategies are based on the construction of a special walk in the graph and by using this walk to explore the network.

2.6 Synchronous Networks

The Black Hole search problem has been studied also in synchronous settings, where the time for an agent to traverse a link is assumed to be unitary.

When the system is synchronous the goals and strategies are quite different from the ones reviewed in the previous sections. In fact, one of the major problem when designing an algorithm for the asynchronous case is that an agent cannot wait at a node for another agent to come back; as a consequence, agents must always move, and have to do it carefully. When the system is synchronous, on the other hand, the strategies are mostly based on waiting the right amount of time before performing a move. The algorithm becomes the determination of the shortest traversal schedule for the agents, where a traversal schedule is a sequence of actions (move to a neighbouring node or stay at the current node). Furthermore, for the black hole search to be solvable, it is no longer necessary that the network is 2-node connected; thus, the black hole search can be performed by synchronous agents also in trees.

In synchronous networks tight bounds have been established for some classes of trees [13]. In the case of general networks the problem of finding the optimal strategy is shown to be NP-hard [14,44] and approximation algorithms are given in [13] and subsequently improved in [43,44]. The case of multiple black holes have been very recently investigated in [12] where a lower bound on the cost and close upper bounds are given.

3 Intruder Capture and Network Decontamination

A particularly important security concern is to protect a network from unwanted, and possibly dangerous intrusions. At an abstract level, an intruder is an alien process that moves on the network to sites unoccupied by the system's agents "contaminating" the nodes it passes by. The concern for the severe damage intruders can cause has motivated a large amount of research, especially on detection (e.g., see [3,36,55]).

3.1 Decontamination and Related Problems

Assume the nodes of the network are initially *contaminated* and we want to deploy a team of agents to *clean* (or decontaminate) the whole network. The cleaning of a node occurs when an agent transits on the node; however, when a node is left without protection (no agents on it) it might become *re-contaminated* according to a recontamination rule. The most common recontamination rule is that as soon as a node without an agent on it has a contaminated neighbour, it will become contaminated again.

A variation of the decontamination problem described above has been extensively studied in the literature under the name of *graph search* (e.g., see [30,42,47,49,53]). The graph search problem has been studied for many classes of graphs, and determining the optimal number of searchers (called *search number*) has been proved to be NP-complete in general.

In the classical graph search problem the agents can be arbitrarily moved from a node "jumping" to any other node in the graph. The main difference in the setting described in this survey is that the agents, which are pieces of software, *cannot be removed from the network*; they can only move from a node to a *neighboring* one. This additional constraint has been introduced and first studied in [5] resulting in a *contiguous, monotone, node search* or *intruder capture* problem. With the contiguous assumption the nature of the problem changes considerably and the classical results on node and edge search do not generally apply. The problem of finding the optimal number of agents is still NP-complete for arbitrary graphs. As we will survey below, the problem has been studied mostly in specific topologies. Also the arbitrary topology has been considered; in this case, some heuristics have been proposed [35] and a move-exponential optimal solution has been given in [11]. Investigations on the relationship between the contiguous model and the classical one for graph search (where the agents can "jump") has been studied, for example, in [8,40].

In this survey we use the term *decontamination* to refer to contiguous monotone node search as defined in [8].

3.2 The Models for Decontamination

Initially, all agents are located at the same node, the *homebase*, and all the other nodes are contaminated; a decontamination strategy consists of a sequence of movements of the agents along the edges of the network. The agents can communicate when they reside on the same node.

Starting from the classical model employed in [5] (called *Local Model*), additional assumptions have sometimes been added to study the impact that more powerful agents' or system's capabilities have on the solutions of our problem.

1) In the *Local Model* an agent located at a node can "see" only local information, like the state of the node, the labels of the incident links, the other agents present at the node.
2) *Visibility* is the capability of the agent to "see" the state of its neighbors; i.e., an agent can see whether a neighboring node is guarded, whether it is clean, or contaminated. Notice that, in some mobile agent systems, the visibility power could be easily achieved by "probing" the state of neighboring nodes before making a decision.
3) *Cloning* is the capability, for an agent, to clone copies of itself.
4) *Synchronicity* implies that local computations are instantaneous, and it takes one unit of time (one step) for an agent to move from a node to a neighboring one.

The efficiency of a strategy is usually measured in terms of number of agents, number of moves performed by the agents, and ideal time.

We say that a cleaning strategy is *monotone* if once a node is clean, it will never be contaminated again. All the results reported here apply for monotone strategies.

3.3 Results in Specific Topologies

Trees. The tree has been the first topology to be investigated in the Local Model [5]. In the paper, the authors show a linear distributed algorithm to determine the minimum number of agents necessary to decontaminate an arbitrary given tree and describe a decontamination strategy. The determination of the optimal number of agents is done through a saturation where appropriate information about the structure of the tree are collected from the leaves and propagated along the tree, until the optimal is known for each possible starting point. In the worst case (complete binary tree) the number of agent is $O(\log n)$, where n is the number of nodes in the tree.

Hypercubes. It has been shown in [32] that to decontaminate a hypercube of size n, $\Theta(\frac{n}{\sqrt{\log n}})$ agents are necessary and sufficient. The employ of an optimal number of agents in the Local Model has an interesting consequence; in fact, it implies that $\Theta(\frac{n}{\sqrt{\log n}})$ is the search number for the hypercube in the classical model, i.e., where agents may "jump".

In the algorithm for the Local Model one of the agents acts as a *coordinator* for the entire cleaning process. The cleaning strategy is carried out on the broadcast tree of the hypercube. The main idea is to place enough agents on the homebase and to have them move, level by level, on the edges of the broadcast tree, leaded by the coordinator in such a way that no recontamination may occur. The number of moves and the ideal time complexity of this strategy are indicated in Table 1.

The visibility assumption allows the agents to make their own decision regarding the action to take solely on the basis of their local knowledge. In fact, the agents are still moving on the broadcast tree, but they do not have to follow the order imposed by the coordinator. The agents on node x can proceed to clean the children of x in the broadcast tree when they "see" that the other neighbors of x are either clean or guarded. With this strategy the time complexity is drastically reduced (since agents move concurrently and independently), but the number of agents increases. Other variations of those two models have been studied and summarized in Table 1.

A characterization of the impact that these additional assumptions have on the problem is still open. For example: an optimal move complexity in the Local Model with Cloning has not been found, and it is not clear whether it exists; when the agents have Visibility, synchronicity has not been of any help although it has not been proved that it is indeed useless; the use of an optimal number of

Table 1. Decontamination of the Hypercube. The star indicates an optimal bound.

		Agents	**Time**	**Moves**
Local	**Local**	$(\star)\ O(\frac{n}{\sqrt{\log n}})$	$O(n \log n)$	$O(n \log n)$
	Local, Cloning, Synchronicity	$n/2$	$(\star)\ \log n$	$(\star)\ n-1$
Visibility	**Visibility**	$n/2$	$(\star)\ \log n$	$O(n \log n)$
	Visibility and Cloning	$n/2$	$(\star)\ \log n$	$(\star)\ n-1$

agents in the weaker Local Model is obtained at the expenses of employing more agents and it is not clear whether this increment is necessary.

Chordal Rings. The Local and the Visibility Models have been subject of investigation also in the Chordal Ring topology in [33].

Let $C(\langle d_1 = 1, d_2, ..., d_k\rangle)$ be a chordal ring network with n nodes and link structure $\langle d_1 = 1, d_2, ..., d_k\rangle$, where $d_i < d_{i+1}$ and $d_k \leq \lfloor \frac{n}{2} \rfloor$. In [33] it is first shown that the smallest number of agents needed for the decontamination does not depend on the size of the chordal ring, but solely on the *length* of the longest chord. In fact, any solution of the contiguous decontamination problem in a chordal ring $C(\langle d_1 = 1, d_2, ..., d_k\rangle)$ with $4 \leq d_k \leq \sqrt{n}$, requires at least $2 \cdot d_k$ searchers ($2 \cdot d_k + 1$ in the Visibility Model).

In both models, the cleaning is preceded by a deployment stage after which the agents have to occupy $2d_k$ consecutive nodes. After the deployment, the decontamination stage can start. Also in the case of the chordal ring, the visibility assumption allows the agents to make their own decision solely on the basis of their local knowledge: an agent move to clean a neighbour only when this is the only contaminated neighbour. The complexity results in the two Models are summarized in Table 2.

Table 2. Results for the Chordal Ring. The $(\star)$ indicates an optimal bound.

Chordal Ring	**Agents**	**Time**	**Moves**
Local	$2d_k + 1\ (\star)$	$3n - 4d_k - 1$	$4n - 6d_k - 1$
Visibility	$2d_k\ (\star)$	$\left\lceil \frac{n-2d_k}{2(d_k-d_{k-1})} \right\rceil$	$n - 2d_k\ (\star)$

Consistently to the observations for the Hypercube, also in the case of the chordal ring the visibility assumption allows to drastically decrease the time complexity (and in this case also the move complexity). In particular, the strategies for the visibility model are optimal both in terms of number of agents and in terms of number of moves; as for the time complexity, visibility allows some concurrency (although it does not bring this measure to optimal as was the case for the hypercube).

Tori. A lower bound for the torus has beed derived in [33]. Any solution of the decontamination problem in a torus $T(h, k)$ with $h, k \geq 4$ requires at least

$2 \cdot min\{h, k\}$ agents; in the *Local model* it requires at least $2 \cdot min\{h, k\} + 1$ agents. The strategy that matches the lower bound is very simple. The idea is to deploy the agents to cover two consecutive columns and then keep one column of agents to guard from decontamination and have the other column move along the torus. The complexity results are summarized in Table 3. As for the other topologies, Visibility decreases time and slightly increases the number of agents. In the case of the torus it is interesting to notice that in the Visibility model all three complexity measures are optimal.

Table 3. Results for the 2-dimensional Torus with dimensions h, k, $h \leq k$

Torus	**Agents**	**Time**	**Moves**
Local	$2h + 1$ $(\star)$	$hk - 2h$	$2hk - 4h - 1$
Visibility	$2h$ $(\star)$	$\lceil \frac{k-2}{2} \rceil$ $(\star)$	$hk - 2h$ $(\star)$

Finally, these simple decontamination strategies can be generalized to d-dimensional tori (although the lower bounds have not been generalized). Let $T(h_1, \ldots, h_d)$ be a d-dimensional torus and let $h_1 \leq h_2 \leq \ldots \leq h_d$. Let N be the number of nodes in the torus and let $H = \frac{N}{h_d}$. The resulting complexities are reported below.

Table 4. Results for a d-dimensional Torus $T(h_1, h_2, \ldots, h_d)$

d-dim Torus	**Agents**	**Time**	**Moves**
Local	$2\frac{N}{h_d} + 1$	$N - 2\frac{N}{h_d}$	$2N - 4\frac{N}{h_d} - 1$
Visibility	$2\frac{N}{h_d}$	$(\lceil h_d - 2 \rceil)/2$	$N - 2\frac{N}{h_d}$

3.4 Different Contamination Rules

In [48] the network decontamination problem has been considered under a new model of *immunity* to recontamination: a clean node, after the cleaning agent has gone, becomes recontaminated only if a weak majority of its neighbours are infected. This recontamination rule is called *local immunization.* The paper studies the effects of this level of immunity on the nature of the problem in tori and trees. More precisely, it establishes lower-bounds on the number of agents necessary for decontamination, and on the number of moves performed by an optimal-size team of cleaners, and it proposes cleaning strategies. The bounds are tight for trees and for synchronous tori; they are within a constant factor of each other in the case of asynchronous tori. It is shown that with local immunization only $O(1)$ agents are needed to decontaminate meshes and tori, regardless of their size; this must be contrasted with e.g. the $2 \min\{n, m\}$ agents required to decontaminate a $n \times m$ torus without local immunization [33]. Interestingly, among tree networks, binary trees were the worst to decontaminate without local immunization, requiring $\Omega(\log n)$ agents in the worst case [5]. Instead, with local immunization, they can be decontaminated by a *single* agent.

References

1. S. Albers, M. Henzinger. “Exploring unknown environments”. *Proc. 29th Annu. ACM Sympos. Theory Comput.*, 416–425, 1997.
2. S. Alpern, S. Gal. *The Theory of Search Games and Rendezvous.* Kluwer, 2003.
3. M. Asaka, S. Okazawa, A. Taguchi, S. Goto. “A method of tracing intruders by use of mobile agent”. *INET*, www.isoc.org, 1999.
4. B. Awerbuch, M. Betke, M. Singh. Piecemeal graph learning by a mobile robot. *Information and Computation* 152, 155–172, 1999.
5. L. Barrière, P. Flocchini, P. Fraigniaud, N. Santoro. “Capture of an intruder by mobile agents”. *Proc. 14th ACM-SIAM Symp. on Parallel Algorithms and Architectures (SPAA)*, 200-209, 2002.
6. L. Barrière, P. Flocchini, P. Fraigniaud, N. Santoro. “Can we elect if we cannot compare?” In *Proc. 15th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, 200–209, 2003.
7. L. Barriere, P. Flocchini, P. Fraigniaud, N. Santoro. “Election and rendezvous in fully anonymous systems with sense of direction”. In *Theory of Computer System*, to appear.
8. L. Barrière, P. Fraigniaud, N. Santoro, D.M. Thilikos. “Searching is not jumping”. *Proc. 29th Int. Workshop on Graph Theoretic Concepts in Computer Science (WG)*, LNCS 2880, 34-45, 2003.
9. M. Bender, A. Fernandez, D. Ron, A. Sahai, S. Vadhan. “The power of a pebble: Exploring and mapping directed graphs”. In *Proc. 30th ACM Symp. on Theory of Computing (STOC)*, 269–287, 1998.
10. M. Bender, D. K. Slonim. “The power of team exploration: two robots can learn unlabeled directed graphs”. In *Proc. 35th Symp. on Foundations of Computer Science (FOCS)*, 75–85, 1994.
11. L. Blin, P. Fraigniaud, N. Nisse, S. Vial. “ Distributed chasing of network intruders by mobile agents”. *Proc. of the 13th Int. Coll. on Structural Information and Communication Complexity (SIROCCO)*, 70–84, 2006.
12. C. Cooper, R. Klasing, T. Radzik “Searching for black-hole faults in a network using multiple agents”. *Proc. 10th Int. Conf. on Principle of Distributed Systems* (OPODIS), 2006.
13. J. Czyzowicz, D. Kowalski, E. Markou, A. Pelc. “Searching for a black hole in tree networks”. *Proc. 8th Int. Conf. on Principle of Distributed Systems* (OPODIS), 35-45, 2004.
14. J. Czyzowicz, D. Kowalski, E. Markou, A. Pelc. “Complexity of searching for a black hole”. *Fundamenta Informaticae*, 71(2-3), 229-242, 2006. 35-45, 2004.
15. S. Das, P. Flocchini, A. Nayak, N. Santoro. “Exploration and labelling of an unknown graph by multiple agents” *Proc. 12th Int. Coll. on Structural Information and Communication Complexity, (SIROCCO)*, 99-114, 2005.
16. S. Das, P. Flocchini, A. Nayak, N. Santoro. “Effective elections for anonymous mobile agents”. *Proc. 17th Int. Symp. on Algorithms and Computation (ISAAC)*, 2006.
17. X. Deng, C. H. Papadimitriou, “Exploring an unknown graph”. *J. of Graph Theory* 32(3), 265–297, 1999.
18. A. Dessmark, P. Fraigniaud, A. Pelc. “Deterministic rendezvous in graphs”. In *Proc. 11th European Symp. on Algorithms (ESA)*, 184–195, 2003.
19. A. Dessmark, A. Pelc. “Optimal graph exploration without good maps”. In *Proc. 10th European Symp. on Algorithms (ESA)*, 374–386, 2002.

20. K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc. "Tree exploration with little memory". *Journal of Algorithms*, 51:38–63, 2004.
21. S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro". "Mobile search for a black hole in an anonymous ring". *Algorithmica*, to appear.
22. S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro. "Multiple agents rendezvous in a ring in spite of a black hole". *Proc. 6th Int. Symp. on Principles of Distributed Systems* (OPODIS) 34-46, 2003.
23. S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro. "Searching for a black hole in arbitrary networks: optimal mobile agents protocols". *Distributed Computing*, to appear.
24. S. Dobrev, P. Flocchini, R. Kralovic, G. Prencipe, P. Ruzicka, N. Santoro. "Optimal search for a black hole in common interconnection networks". *Networks*, 47 (2), p. 61-71, 2006.
25. S. Dobrev, P. Flocchini, N. Santoro. "Improved bounds for optimal black hole search in a network with a map. *Proc. 10th Int. Coll. on Structural Information and Communication Complexity* (SIROCCO), 111-122, 2004.
26. S. Dobrev, P. Flocchini, R. Kralovic, N. Santoro. "Exploring a dangerous unknown graph using tokens". *Proc. 5th IFIP Int. Conf. on Theoretical Computer Science* (TCS), 131-150, 2006.
27. S. Dobrev, P. Flocchini, N. Santoro. "Cycling Through a Dangerous Network: A Simple Efficient Strategy for Black Hole Search". *Int. Conf. on Distributed computing Systems* (ICDCS), 2006.
28. S. Dobrev, R. Kralovic, N. Santoro, W. Shi. "Black Hole Search in Asynchronous Rings Using Tokens". *Proc. 6th Conf. on Algorithms and Complexity* (CIAC), 139-150, 2006.
29. G. Dudek, M. Jenkin, E. Milios, D. Wilkes. "Robotic exploration as graph construction". *Transactions on Robotics and Automation*, 7(6):859–865, 1991.
30. J. Ellis, H. Sudborough, J. Turner. "The vertex separation and search number of a graph". *Information and Computation*, 113(1):50–79, 1994.
31. P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk. "Multiple mobile agent rendezvous in a ring". Proc. *6th Latin American Theoretical Informatics Symp.* (LATIN), 599–608, 2004.
32. P. Flocchini, M.J. Huang, F.L. Luccio. "Contiguous search in the hypercube for capturing an intruder" *Proc. 18th IEEE Int. Parallel and Distributed Processing Symp.* (IPDPS), 2005.
33. P. Flocchini, M.J. Huang, F.L. Luccio. "Decontamination of chordal rings and tori". *Proc. 8th Workshop on Advances in Parallel and Distributed Computational Models (APDCM)*, 2006.
34. P. Flocchini, B. Mans, N. Santoro. "Sense of direction in distributed computing". *Theoretical Computer Science*, vol. 291, 29-53, 2003.
35. P. Flocchini, A. Nayak, A. Shulz. " Cleaning an arbitrary regular network with mobile agents" *Proc. of the 2nd Int. Conf. on Distributed Computing & Internet Technology (ICDCIT)*, 132-142, 2005.
36. N. Foukia,J. G. Hulaas, J. Harms. "Intrusion Detection with Mobile Agents". *INET*, www.isoc.org, 2001.
37. P. Fraigniaud, L. Gasieniec, D. Kowalski, A. Pelc. "Collective tree exploration". *Networks*, to appear.
38. P. Fraigniaud, D. Ilcinkas, "Digraph exploration with little memory". *Proc. 21st Symp. on Theoretical Aspects of Computer Science* (STACS), 246–257, 2004.
39. P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, D. Peleg. "Graph exploration by a finite automaton". *Theoretical Computer Science*, to appear.

40. P. Fraigniaud, N. Nisse. "Monotony Properties of Connected Visible Graph Searching". *Proc. 32nd Int. Workshop on Graph-Theoretic Concepts in Computer Science* (WG)22-24, 2006.
41. 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.
42. L. Kirousis, C. Papadimitriou. "Searching and pebbling". *Theoretical Computer Science*, 47(2):205–218, 1986.
43. R. Klasing, E. Markou, T. Radzik, F. Sarracco. "Approximation bounds for black hole search problems". *Proc. 9th Int. Conf. on Principle of Distributed Systems* (OPODIS), 2005.
44. R. Klasing, E. Markou, T. Radzik, F. Sarracco. "Hardness and approximation results for black hole search in arbitrary graphs". *Proc. 12th Int. Coll. on Structural Information and Communication Complexity* (SIROCCO), 200-215, 2005.
45. E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk. "Mobile agent rendezvous in a ring". In *Int. Conf. on Distibuted Computing Systems (ICDCS)*, 592–599, 2003.
46. E. Kranakis, D. Krizanc, S. Rajsbaum. "Mobile agent rendezvous". *Proc. 13th Int. Coll. on Structural Information and Communication Complexity* (SIROCCO), 1–9, 2006.
47. A. Lapaugh. "Recontamination does not help to search a graph". *Journal of the ACM* 40(2), 224–245, 1993.
48. F. Luccio, L. Pagli, N. Santoro. "Network decontamination with local immunization". *Proc. 8th Workshop on Advances in Parallel and Distributed Computational Models (APDCM)*, 2006.
49. N. Megiddo, S. Hakimi, M. Garey, D. Johnson, C. Papadimitriou. "The complexity of searching a graph". *Journal of the ACM* 35(1), 18–44, 1988.
50. R. Oppliger. "Security issues related to mobile code and agent-based systems". *Computer Communications*, 22(12):1165 – 1170, 1999.
51. P. Panaite, A. Pelc, "Exploring unknown undirected graphs". *Journal of Algorithms*, 33 281-295, 1999.
52. P. Panaite, A. Pelc. "Impact of topographic information on graph exploration efficiency". *Networks*, 36, 96–103, 2000.
53. T. Parson. "The search number of a connected graph". In the 9th *Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica, 549–554, 1978.
54. T. Sander, C. F. Tschudin. "Protecting mobile agents against malicious hosts". In *Proc. of Conf on Mobile Agent Security*, LNCS 1419, pages 44–60, 1998.
55. E. H. Spafford, D. Zamboni. "Intrusion detection using autonomous agents". *Computer Networks*, 34(4):547–570, 2000.
56. J. Vitek, G. Castagna. "Mobile computations and hostile hosts". In D. Tsichritzis, editor, *Mobile Objects*, pages 241–261. University of Geneva, 1999.