Recherche optimale de trou noir avec cailloux

Paola Flocchini, David Ilcinkas, Nicola Santoro

Recherche optimale de trou noir avec cailloux

Paola Flocchini[1,†], David Ilcinkas[2,‡] et Nicola Santoro[3,†]

[1] *SITE, University of Ottawa, Canada. E-mail:* flocchin@site.uottawa.ca
[2] *CNRS, LaBRI, Université Bordeaux I, France. E-mail:* david.ilcinkas@labri.fr
[3] *School of Computer Science, Carleton University, Canada. E-mail:* santoro@scs.carleton.ca

Un trou noir est un nœud d'un réseau qui détruit tout agent (ou robot) y entrant sans laisser de trace détectable. L'emplacement du trou noir doit être déterminé par une une équipe d'agents mobiles identiques déployée à partir d'un emplacement sain. Pratiquement tous les résultats existants pour des agents asynchrones supposent la présence à chaque nœud d'une mémoire partagée (tableau blanc), de taille logarithmique, sur laquelle les agents peuvent lire et écrire. Un méchanisme moins puissant et moins exigeant consiste en l'utilisation de cailloux identiques (qui peuvent être déposés sur les nœuds, repris, et transportés par les agents), traditionnellement employés pour l'exploration de graphes sains (*i.e.* sans trou noir). Deux résultats récents montrent qu'il est possible d'utiliser des cailloux comme moyen de communication pour la recherche de trou noir. Ces résultats autorisent cependant les cailloux à être placés non seulement sur les nœuds mais aussi sur les liens. Ils supposent également que les liens sont FIFO.

Dans ce papier, nous considérons le modèle des cailloux idéaux, c'est-à-dire le modèle où un caillou ne peut être placé que sur les nœuds, et pas plus d'un caillou ne peut être placé sur un nœud donné. Nous prouvons que pour les réseaux de topologie connue il est possible d'obtenir exactement les mêmes bornes optimales en utilisant des cailloux idéaux qu'en utilisant des tableaux blancs, et ce même si les liens ne sont pas FIFO. Plus précisément, nous prouvons qu'une équipe de *deux* agents asynchrones, chacun équipé d'un seul caillou indistinguable (qui ne peut être placé que sur les nœuds, avec au plus un caillou par nœud), peut localiser le trou noir. Ce résultat est obtenu en utilisant le nombre optimal de mouvements $\Theta(n \log n)$, où n est le nombre de nœuds. Pour résumer, nous fournissons la première preuve que, pour la recherche de trou noir, le modèle des cailloux idéaux est aussi puissant que le modèle des tableaux blancs.

Keywords: Calcul réparti, exploration de graphes, robots autonomes, agents mobiles, graphes dangereux.

1 Introduction

Black Hole Search (BHS) is a multi-agent problem set in a graph G: a team of (identical) cooperating mobile agents (or robots) must determine the location in G of a *black hole* (BH): a node where any incoming agent is destroyed without leaving any detectable trace. The problem is solved if at least one agent survives and all surviving agents know the location of the black hole.

The practical interest in this problem derives from the fact that, in the pressing security concern on *host attacks* on mobile agents (e.g., [Che98, Opp99]), a black hole describes a highly harmful host. Even more important, a black hole can model several types of faults, both hardware and software, arising in networked systems with code mobility. Clearly, in presence of such a harmful host, the first step must be to *identify* it, if possible; i.e., to determine and report its location.

From a theoretical point of view, the original interest in this problem is amplified by the fact that it is a new *exploration* problem; in fact, the black hole can be located only after all the nodes of the network but one have been visited and are found to be safe. Clearly, in this exploration process some agents may disappear in the black hole. In other words, while the existing wide body of literature on exploration (e.g., see [BFRSV02, DP99, PP99]) assumes that the graph is *safe*, BHS opens the research problems of the exploration of *dangerous graphs*.

Indeed BHS has been extensively studied in several settings, under a variety of assumptions on the power of the adversary and on the capabilities of the agents. The differences are on the assumptions made e.g.

[†] Partially supported by NSERC discovery grant.
[‡] This work was done during the stay of David Ilcinkas at the University of Ottawa, as a postdoctoral fellow.

on the level of synchronization of the agents; on whether or not the movements of the agents on links are FIFO; on the type of mechanisms available for inter agent communication and coordination. In these investigations, the research concern has been to determine under what conditions and at what cost mobile agents can successfully accomplish this task. The main complexity measure of a solution protocol is the number of agents employed. The second one is the total number of moves performed by the agents.

In this paper we are interested in the weakest settings that still make the problem solvable. Thus we will make no assumptions on timing or delays, and focus on the *asynchronous* setting. Indeed, while the research has also focused on the *synchronous* case (e.g. [CKR06]) where all agents are synchronized and delays are unitary, the main body of the investigations has concentrated on the asynchronous one (e.g., [DFKS06, DFPS06, DSS06]).

The majority of the investigations operate in the *whiteboard* model: every node provides a shared space for the arriving agents to read and write (in fair mutual exclusion). This model is very powerful: it endows the agents not only with direct and explicit communication capabilities, but also with the means to overcome severe network limitations, such as non-FIFO links. A weaker and less demanding interaction mechanism, used in the early investigations on (safe) graph exploration is that provided by the use of identical *pebbles* (that can be placed on nodes, picked up and carried by the agents) without any other form of marking or communication (e.g., [BFRSV02]).

The quest now is to determine if BHS can be solved using pebbles instead of whiteboards. The importance of this quest goes beyond the specific problem, in that it would shed some light on the relative computational power of these two inter-agent communication mechanisms. Two results have been established so far, both assuming FIFO links. In [DFKS06] it has been shown that $\Delta+1$ agents (Δ denotes the maximun node degree in G) without a map (the minimum team size under these conditions), each endowed with an identical pebble, can locate the black hole with a (very high but) polynomial number of moves. In [DSS06] it has been shown that two agents with a map (the minimum team size under these conditions), each endowed with a constant number of pebbles, can locate the black hole in a ring network with $\Theta(n\log n)$ moves.

Although suggestive, these results are not yet proofs that pebbles are computationally as powerful as whiteboards with regards to BHS. This is because they are *not* established within the *pure token* model used in the traditional exploration problem; in fact, here agents are allowed to place pebbles not only on nodes but also on links (e.g., to indicate on which link it is departing).

In this paper, we provide the first proof that indeed the pure token model is computationally as powerful as the whiteboard model for BHS. The context we examine is the one of agents with a map in an arbitrary graph. It is known that, in this case, with whiteboards, *two* agents are necessary and sufficient, and that $\Theta(n\log n)$ moves are necessary and sufficient [DFPS06]. In this context we prove that: (1) A team of *two* asynchronous agents, each endowed with a single identical pebble (that can be placed only on nodes) and a map of the graph can locate the black hole. (2) This result can be obtained with $\Theta(n\log n)$ moves. (3) Those bounds hold even if the links are not FIFO. In other words, for networks of known topology, using pure tokens it is possible to obtain exactly the same optimal bounds as using whiteboards.

Unlike the previous results with impure tokens, our bounds hold even if the links are not FIFO. Note that our result implies as a corollary an optimal solution for the whiteboard model using only a single bit of shared memory per node; the existing solution [DFPS06] requires whiteboards of $O(\log n)$ bits in size.

2 Terminology and Definitions

Let $G=(V,E)$ be a simple biconnected graph with $n=|V|$ nodes containing exactly one black hole. Edges incident to a node are identified by port numbers. Operating in G is a team of identical autonomous mobile agents (or robots). All agents enter the system from the same node, called homebase. The agents have computing capabilities, bounded computational storage, and a map of G with port numbers and the indication of the homebase; they can move from node to neighbouring node, and obey the same set of behavioral rules (the *algorithm*). Every agent has a *pebble*; all pebbles are identical. A pebble can be carried, put down at a node if no other pebble is already there, and picked up from a node by an agent without pebbles.

When an agent enters a node, it can see if there is a pebble dropped there; it might be however unable to see other agents there or to determine whether they are carrying a pebble with them.

The system is *asynchronous*. However, the action of determining the presence of a pebble at a node and dropping or picking up the pebble (depending on the algorithm) is locally atomic; no other action can occur at the same node at the same time. Links are not FIFO: two agents moving on the same link in the same direction at the same time might arrive at destination in an arbitrary order.

3 Optimal black hole search with pebbles

We first describe an algorithm for rings and then we briefly show how to generalize it to arbitrary graphs.

3.1 Preliminaries

Without loss of generality, we can assume that the clockwise (or right) direction is the same for both agents. An agent exploring to the right (resp. left) is said to be a *right* (resp. *left*) agent. For $i \geq 0$, the node at distance i to the right, resp. to the left, of the home base will be called node i, resp. node -i.

In the algorithm the agents obey the two following metarules: (1) an agent always ensures that a pebble is lying at u before traversing an unknown edge $\{u,v\}$ from u to v (i.e. an edge that it does not know to be safe); (2) an agent never traverses an unknown edge $\{u,v\}$ from u to v if a pebble lies at u but the pebble was not dropped here by this agent.

These two metarules imply that the two agents never die in the black hole coming from the same edge. Moreover, each agent keeps track of its progress by storing the number of the most-right, resp. most-left, node in a variable `Last_Right`, resp. `Last_Left`. These two variables are used to detect termination: when only one node remains unexplored, this node is the black hole and the agent can stop.

An agent is said to traverse an edge $\{u,v\}$ from u to v using *cautious walk* if it has one pebble, it drops it at u, traverses the edge, comes back to u, retrieves the pebble and goes again to v. An agent is said to traverse an edge $\{u,v\}$ from u to v using *double cautious walk* if it has one pebble and the other is at u, it goes to v carrying one pebble, drops the pebble at node v, comes back to u, retrieves the other pebble and goes again to v. Note that these two cautious explorations obey the first metarule.

3.2 The algorithm for rings

Our algorithm is divided in two phases. The second phase may not exist in all possible executions.

Phase 1. In the first phase, exploration to the right is always done using cautious walk, while exploration to the left is always done using double cautious walk. Note that, since an agent exploring to the right uses one pebble and an agent exploring to the left uses two pebbles, the agents cannot make progress simultaneously in two different directions because there are only two pebbles in total. This also implies that while an agent is exploring new nodes it knows all the nodes that have already been explored, as well as the position of the only unexplored node where the other agent possibly died. This prevents the agents from exploring the same node and thus from dying in the black hole from two different directions.

Phase 1 proceeds as follows. Initially both agents explore to the right. Since links are not FIFO, an agent may pass the other and take the lead without any of the two noticing it. Nevertheless it eventually happens that one agent L finds the pebble of the other agent R, say at node p (at the latest it happens when one agent locates or dies in the black hole). When this happens L drops its pebble at node $p-1$ (if its pebble is not already there) and steals R's pebble. Having control on the two pebbles, L starts to explore left using double cautious walk. When/if R comes back to p to retrieve its pebble, it does not find it. It then goes left until it finds a pebble. Agent R does eventually find a pebble because Agent L never removes a pebble before putting the other pebble further to the left. At this point R retrieves the pebble and goes right again to explore to the right. When/if L realizes that one of its pebble has been stolen, it changes role and explores to the right using its remaining pebble. At this point, both agents explore to the right. Again, one agent will find and steal the pebble of the other. To ensure progress in exploration, a right agent puts down its token only when it reaches the last visited node to the right it knows (using its variable `Last_Right`). Consequently a stealing on the right always occurs at least one node further to the right from the previous time. Hence the algorithm of Phase 1 is in fact correct by itself but the number of moves can be $\Theta(n^2)$ in the worst case (one explored node every $O(n)$ moves). To decrease the worst case number of moves to $O(n \log n)$, the agents switch to Phase 2 as soon as at least two nodes have been explored to the right.

Phase 2. Phase 2 uses the *halving* technique, reminiscent of the one of [DFPS06], but highly complicated by the absence of whiteboards and by the lack of FIFO. The idea is to regularly divide the workload (the unexplored part) in two. One agent has the left half to explore, while the second agent explores the right half. These explorations are performed concurrently by using (simple) cautious walk. After finite time, exactly one agent finishes its part and joins the other in exploring the other part. At some point, one agent A will see the other agent's token. A steals the token and moves it by one position to indicate a change of stage to the second agent B. It then computes the new workload, divide it into two parts, and goes and explores its newly assigned part. This can happen several times (if B remains blocked by the asynchronous adversary or if it is dead in the black hole). When/if agent B comes back to retrieve its pebble, it does not find it. It further goes back to retrieve its pebble. The number of moves it has to perform to find the pebble indicates how many halvings (pair of stages) it misses. Knowing that, it can compute what is the current unexplored part and what is its current workload. It then starts to explore its part. Since there are at most $O(\log n)$ stages of $O(n)$ moves each, this leads to a total number of moves of $O(n \log n)$.

The reason why the algorithm starts with a few stages of Phase 1 is that the algorithm of Phase 2 needs some safe nodes to put the pebble that is used as a message indicating the current partition of the workload.

Several other technical details and precautions have to be taken because of asynchrony and lack of FIFO.

3.3 The main result

The algorithm we propose for arbitrary networks is an adaptation of the algorithm that we described for rings. To be able to apply this latter algorithm in a general graph, each agent will maintain a partial mapping between the node numbers used in the algorithm and the actual nodes in the network (or its map), such that at any point in time an agent knows what means "go left" or "go right". If an agent lies at some node i, then going one step right, resp. left, from node i means going to node $i+1$, resp. $i-1$, by a shortest safe path.

Since these paths are not necessarily of constant length, this technique can increase the number of moves. To decrease it to $O(n \log n)$, we also use (safe) shortcuts. More precisely, in some specific cases, an agent is allowed to go directly (by a shortest safe path) from some node i to some other node $j > i$ without going through nodes $i+1$ to $j-1$ (and similarly if $j < i$).

Theorem 1 *Consider a n-node graph containing a homebase and a black hole, and two agents running our algorithm from the home base. After at most $O(n \log n)$ moves in total, there remains at least one surviving agent and all surviving agents have terminated and located the black hole.*

The optimality of the algorithm follows from the fact that, in an arbitrary graph, the problem cannot be solved with less agents or (asymptotically) less moves [DFPS06], and clearly not with less pebbles.

References

[BFRSV02] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176(1):1–21, 2002

[Che98] D.M. Chess. Security issues in mobile code systems. *Proc. Conference on Mobile Agent Security*, LNCS 1419, 1–14,1998.

[CKR06] C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *Proc. 10th Int. Conference on Principles of Distributed Systems* (OPODIS), 320-332, 2006.

[DP99] X. Deng and C. Papadimitriou, Exploring an unknown graph. *J. Graph Theory* 32(3), 265–297, 1999.

[DFKS06] S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring a dangerous unknown graph using tokens. Proc. *5th IFIP Int. Conf. on Theoretical Computer Science* (TCS), 131-150, 2006.

[DFPS06] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocol. *Distributed Computing* 19 (1), 1-19, 2006.

[DSS06] S. Dobrev, N. Santoro, and W. Shi. Black hole search in asynchronous rings using tokens. Proc. *6th Conf. on Algorithms and Complexity* (CIAC), 139-150, 2006.

[PP99] P. Panaite and A. Pelc, Exploring unknown undirected graphs, *Journal of Algorithms*, 33, 281-295, 1999.

[Opp99] R. Oppliger. Security issues related to mobile code and agent-based systems. *Computer Communications*, 22(12), 1165 – 1170, 1999.