Searching with Mobile Agents in Networks with Liars

Nicolas Hanusse[1], Evangelos Kranakis[2], and Danny Krizanc[3]

[1] INRIA Rocquencourt, Carleton University, School of Computer Science, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada,
hanusse@scs.carleton.ca
[2] Carleton University,
kranakis@scs.carleton.ca
[3] Weysleyan University, Middletown, Connecticut 06459, US,
dkrizanc@caucus.cs.wesleyan.edu

Abstract. In this paper, we present algorithms to search for an item s contained in a node of a network, without prior knowledge of its exact location. Each node of the network has a database that will answer queries of the form "how do I get to s?" by responding with the first edge on a shortest path to s. It may happen that some nodes , called *liars*, give bad advice. If the number of liars k is bounded, we show different strategies to find the item depending on the topology of the network. In particular we consider the complete graph, ring, torus, hypercube and bounded degree trees.

1 Introduction

Mobile agents can perform very complex information gathering, like assembling and digesting "related" topics of interest. Depending on their "behavior" mobile agents can be classified as reactive (responding to changes in their environment) or pro-active (seeking to fulfill certain goals). Moreover agents may choose to remain stationary (filtering incoming information) or become mobile (searching for specific information across the Internet and retrieving it) [16]. There are numerous examples of such agents in use today, including the Internet search engines, like Yahoo, Lycos, etc.

In the present paper we consider the problem of searching for an item in a distributed network in the presence of "liars." The objective is to design a mobile agent that travels along the network links in order to locate the item. Although the location of the item in the network is initially unknown, information about its whereabouts can be obtained by querying the nodes of the network. The nodes have databases providing the first edge on a shortest path to the item sought. The agent queries the nodes; the queried nodes respond either by providing a link adjacent to them that is on a shortest path to the node that holds the item or if the desired item is at the node itself then the node answers by providing it to the agent. However certain nodes in the network may be liars, e.g., due to out-of-date network information in their databases. The liars are unknown to the mobile agent that must still find the item despite the fact that queries to responses may be wrong. In this paper we give deterministic algorithms for searching in a distributed network with a bounded number of liars that has the topology of a complete network, ring, torus, hypercube, or trees under three models of liars.

A variant of the above *searching* model, was introduced in [9], where the network topologies considered were the *ring* and the *torus* and the nodes respond to queries with a bounded probability of being incorrect. Additional investigations under the same model of "searching with uncertainty" were carried out for fully interconnected networks in [10]. Models with faulty information in the nodes have been considered before for the problem of routing (see [1, 3, 7, 8, 14]). However, in this problem it is assumed that the identity of the node that contains the information is known, and what is required is to reach this node following the best possible route. Search problems in graphs, where the identity of the node that contains the information sought is not known, have been considered before. These include *deterministic search games*, where a fugitive that possesses some properties hides in the nodes or edges of a graph [5, 12, 13]), and the problem of exploring an *unknown* graph [2, 11, 15]. Our model is similar in spirit to the model in [4] where the authors propose algorithms to search for a point on a line or on a lattice drawn on the plane. While in the models they consider there is limited if any knowledge of the location of the point, the nodes along the way do not provide new location information at each step as in our case.

1.1 Preliminaries and Definitions

In order to present the problem more precisely, we must define the search model in a given network. Since a mobile agent does not know if a network has liars, we suppose it assumes the number of liars is bounded by k. This assumption may affect the moves of the mobile agent. Thus, the complexity of our results will depend on the actual distance d of the mobile agent to the destination as well as the number of liars assumed.

The mobile agent is basically a software program running an algorithm that requires a certain amount of memory, storing relevant information about its current position in the network, e.g. in a binary tree the distance to the root, in a ring the distance from the starting node, etc. We will see later that the algorithm depends on the topology of the network and we will consider different trade-offs between the amount of memory required by the mobile agent and the number of steps, i.e the number of moves of the mobile agent.

A network of n nodes is represented as a connected undirected graph $G = (V, E)$ where V is the set of vertices or nodes and E the set of edges or links. Let s denote the item the mobile agent is searching for and assume there is a unique node in G containing s. A *query* $Q_u(s)$ returns either s if the node u contains

s or a subset of edges, incident to u, belonging to a shortest path leading to the item s. If $Q_u(s)$ returns an edge that does not belong to a shortest path to s, the node u is called a *liar*, otherwise a *truthteller*. The path $p = u_0u_1 \cdots u_\alpha$ is a sequence of nodes followed by the mobile agent until item s is found. The number of edges followed by the path p is called the *number of steps* of the mobile agent, which is denoted by α. If there are no liars we expect that the mobile agent will follow an optimal path, i.e. if $k = 0$, it is obvious that $\alpha = d$ where d is the distance between the starting node of the mobile agent and the node containing the item. δ (resp. Δ) is the minimal (resp. maximal) degree of a given network. By convention, we consider that the nodes can be labelled by the set $\{1, 2, \dots, n\}$.

1.2 Models

We consider three models of responses to queries:

One advice per node with co-ordination (CO) Model: In this case, a query returns a unique edge. We assume some preprocessing was done when building the databases stored in each node u of V. Let v be the node containing s and choose a fixed shortest path tree with destination v. For a given node u, $Q_u(s) = e$ where e is the (unique) outgoing edge incident to u chosen in this shortest path tree. If a node indicates an edge on another shortest path this node is considered to be a liar. The mobile agent is assumed to have knowledge as to how the shortest path trees were originally constructed. For example, we assume they always report first a row and then a column in the case of the torus. The truthtellers are *co-ordinated* in that the set of edges they report leads to the construction of a particular shortest path spanning tree. An adversary may decide which nodes are liars but has no influence over which edges are to be reported by the truthtellers.

One advice per node without co-ordination (NCO) Model: In this model an adversary does not only decide which nodes are liars but also which correct edge the truthtellers will report whenever there is a choice of shortest path edges.

One advice per edge (ECO) Model: In this model a truthteller returns $Q_u(s)$ equal to the set of all incident edges to u belonging to a shortest path tree. A liar may return any (presumably non-empty) subset of the edges incident to u. Again the adversary has no input as to what is returned by a truthteller.

1.3 Results

In this paper, we consider searching for an item under the above models and for different topologies: complete graph, ring, torus, hypercube, trees. In each case, we assume that the mobile agent knows the topology of the network and suspects a bounded number k of liars. We assume that the responses of the nodes are set before the start of the algorithm according to the model considered and that they do not change throughout the running of the algorithm. The cost measures we consider for a given algorithm is the number of steps (i.e, edges traversed) and the amount of memory required by the mobile agent. Proofs are left out and only sketches of algorithms are presented due to space limitations.

2 Complete Graphs

In this section, we present two algorithms. The first one prioritizes the number of steps and the second one the amount of memory. We also establish two lower bounds on the number of steps for the complete graph and for any graph. Algorithm SEARCHCOMPLETE(S) works as follow: starting from a node u , we follow its advice to node u' unless we have already visited u' in which case we select any node not previously visited and go there.

Theorem 1. *In any complete graph of n vertices with k liars, a mobile agent can find an item in at most $k+1$ steps with $k \log n$ bits of memory .*

Theorem 2. *Let $D(u,p)$ be the set of nodes at a distance p from u and B_p be the set of nodes at a distance at most p. For any graph so that $|D(u,p)| > 1, |B_p| \leqslant k$ and $|B_{p+1}| > k$ then a mobile agent starting from u may require at least $d+k$ steps to find an item, where d is the distance between the starting node and s.*

We may be interested in a trade-off between the memory and the number of steps required by a mobile agent to find an item. Algorithm SEARCHCOMPLETE2 illustrates this idea: follow advice of nodes labeled $1, 2, \ldots, k+1$ until you find the item, i.e. if node labeled i gives a bad advice and sends you to a node u then go to node labeled $i+1$.

Theorem 3. *In any complete graph of n vertices and k liars, a mobile agent can find an item in at most $2k+3$ steps with $\log k$ bits of memory.*

3 Ring and Torus

For the ring, each vertex is of degree two and we may consider a global orientation known by each processor. Each node has a left and a right edge labelled respectively Left and Right , i.e. the query $Q_u(s)$ returns Left or Right. Algorithm SEARCHRING(s,k) works as follow: (1) choose a direction to follow, (2) move in this direction until either s is found or $k+1$ query responses in the opposite direction are given and then move in the opposite direction.

Theorem 4. *In a ring of n vertices with k liars, a mobile agent can find an item in at most $d+4k+2$ steps with $O(\log k)$ bits of memory.*

Theorem 5. *There exists a distribution of k liars in the ring of n vertices for which the number of steps is at least $d + 2k$.*

We present three algorithms to find the item in a torus of n vertices. As for the ring, we suppose there exists a global orientation of the edges known by each node and its four incident edges are labelled L, R, U, D for the left, right, up and down direction. We also use the notation $\leftarrow, \rightarrow, \uparrow, \downarrow$ for the edges. u represents the current location.If dir is a direction, $\bar{dir}$ indicates the opposite direction: $\leftarrow = \bar{\rightarrow}$ and $\uparrow = \bar{\downarrow}$. The *advice of a block* or of a rectangle consists in the set of directions $\{a_1, ..., a_t\}$ so that each a_i has been given at least $k + 1$ times.

<u>**CO Model**</u>: For this model, we assume truthtellers always report first a row and then a column. The algorithm SearchRingII(s, m, l) travels in a set, called *block*, of l consecutive nodes along the direction m and returns the number of query responses for each direction $\leftarrow, \rightarrow, \uparrow, \downarrow$. The *advice of a block B* corresponds to the direction indicated by the majority of B.

The sketch of the algorithm SearchTorus(s) is the following: (1) follow the advice of a block of size $2k + 1$ until two blocks B, B' are found with the opposite advice, (2) locate the column of s by a walk [1] in a square containing B, B',(3) find s in the column c using a search algorithm in a ring.

Theorem 6. *In any torus of n vertices and k liars, a mobile agent of $O(k \log k)$ bits of memory can find an item in at most $d + O(k)$ steps.*

<u>**NCO and ECO Models**</u>: In the NCO Model, the walk of SearchTorus does not work. Indeed, a row of truthtellers may indicate different columns for the item. We propose a new strategy to choose a starting direction in SearchTorusII, we make a search within a square of area $O(k)$ instead of a segment of $O(k)$ nodes along a given direction (in SearchTorusIII, we will use the previous method). We propose a variant of an algorithm which can be found in [9] to choose a starting direction in a square:

SearchSquare(s, u, l): (a) For each direction dir, $a_{dir} = 0$, let $m = \{\}$; (b) mobile agent searches for the desired item s by testing all nodes in a square B of area l centered at node u; for each node of advice dir, $a_{dir} = a_{dir} + 1$; (c) return $\{a_{dir}\}$;

The idea of SearchTorusII is the following: (1) we first locate s in a band of columns (or rows) $c_1, \ldots, c_w$ of width $w = O(\sqrt{k})$ finding two adjacent squares S, S' of area $4k + 1$ with different horizontal or vertical advice, (2) we find the vertical (horizontal) direction to follow by a walk in a rectangle R of size $O(\sqrt{k}) * (2k + 1)$ containing S, S' (3) we search for s in in the direction given by R in consecutive rectangles of size $O(\sqrt{k})$ in the direction given by R.

[1] this walk not described here finds c in $O(k)$ steps

Roughly speaking, since each iteration of Step 3[2] takes $O(k)$ steps and moves the mobile agent $\Omega(\sqrt{k})$ closer to the item, we have the following result:

Theorem 7. *In any torus of n vertices and k liars, a mobile agent can find an item in at most $O(d\sqrt{k})$ steps with $O(\log k)$ bits of memory.*

If $d = \Omega(\sqrt{k} \log k)$, another strategy illustrated by SearchTorusIII may be interesting: (1) we first locate s in a band of columns (or rows) $c_1, \dots, c_w$ finding two consecutive blocks B, B' of size $4k+1$ with different advice[3], (2) The next steps consist of applying a variant of the dichotomy principle in rectangles of size $O(k) \times O(k)$ to find the column of s.

Theorem 8. *In any torus of n vertices and k liars, a mobile agent of $O(\log k)$ bits of memory can find an item in at most $O(d + k \log k)$ steps.*

In the ECO Model, the mobile agent may use the same algorithms as for CO Models. Indeed, the mobile agent can do the co-ordination itself choosing one edge per node. Since we have a lower bound of $d + \Omega(k)$ steps for any model, the upper bounds in ECO Model does not change a lot if we do not pay particular attention to the constants.

4 Hypercube

For the hypercube, we show that the ECO model has an advantage over the CO model. Let C_n be an hypercube of 2^n vertices. Each node u is coded by $(x_n, \dots, x_1)$ with $x_i \in \{0, 1\}$. As for the torus, we assume there exists a global orientation of edges known by each node such that two nodes are adjacent along the direction i, labelled $\rightarrow_i$, if they agree in all but the position i. For $n \geqslant 2$, C_n is hamiltonian (see [6]) and it follows that any subgraph of C_n isomorphic to $C_{n'}$ with $n' < n$ is hamiltonian. Moreover, there exists in C_n an hamiltonian circuit.

In this model, the co-ordination works in the following way : each node always reports first the direction 1, then direction 2, ... , direction n. In other words, if the advice of a node $u = (x_n, \dots, x_1)$ is $\rightarrow_i$, it indicates that the destination v should have at least $i - 1$ identical coordinates $x_{i-1} \dots x_1$.

Let us consider the starting node $u = (x_n, \dots, x_1)$ and the node $v = (y_n, \dots, y_1)$ is the node containing s. The idea of the algorithm to find the coordinates of v is the following : let $i = 1$; (1) we choose a subgraph $Q' = C_{\lceil 2k+1 \rceil} \in C_n$ such that all nodes of Q' have same coordinates $x_i, y_{i-1}, \dots, y_1$ (2) we follow an hamiltonian path in Q' and compute, for the first $2k + 1$ nodes, the number m of responses $\rightarrow_i$ of Q'; (3) if $m > k$ then $y_i = 1 - x_i$ else $y_i = x_i$; (4) repeat Step 1 until the item is found.

[2] it may happen that we found s in Step 2 but the walk in the rectangle R is a spiral to obtain the same result

[3] This can be done with $O(k)$ extra steps

Theorem 9. *In an hypercube C_n of 2^n vertices with k liars, a mobile agent of $O(n + \log k)$ bits of memory can find an item in at most $d(2k + 1)$ steps.*

In the ECO Model, a node u gives a response $Q_u = (a_{n-1}, \dots, a_0)$. The position of s is given using the majority among $2k + 1$. Immediately, an easy upper bound of $d + 4k + 2$ steps can be obtained by following $2k + 1$ nodes in a hamiltonian path in C_n. This result can be improved if we consider only a hamiltonian path in a subgraph of C_n isomorphic to $C_{\lceil \log (2k+1) \rceil}$.

Theorem 10. *In a hypercube C_n of 2^n vertices with k liars, a mobile agent can find an item in at most $d + 2k + 1 + \lceil \log (2k + 1) \rceil$ steps with $O(n \log k)$ bits of memory.*

5 Trees

We pay particular attention on the CO Model for a tree. Indeed, the shortest path is unique and so, we do not have a question of co-ordination of the nodes. We present one algorithm for bounded degree trees. In this case, the ECO model would lead to the same result as for the CO model.

We suppose that we are starting from a node u_1, considered as a root of the tree. Node u_1 gives an orientation of the edges. Each node, except the root, has $\Delta - 1$ incident edges, corresponding to the directions upward, downward 1, downward 2, etc. and labelled $\uparrow$, $\downarrow_1, \dots, \downarrow_{\Delta-1}$. By convention, the edge pointing upward is the edge leading to the root. A node u is a *suspect* if its response is upward and if its parent's response is downward. SearchTree(s, k) works as follow: (1) follow the downward advice until either a suspect u_l or s is found (2) traverse the all subtree rooted in u_l of depth $2k$ and choose to follow the k first edges belonging to the path to leaves with the maximum of downward responses, (3) iterate first step. Analyzing SearchTree, we obtain :

Theorem 11. *In a tree of bounded degree Δ of n vertices and k liars, a mobile agent can find an item in at most $d + O((\Delta - 1)^{2k+1})$ steps with $O(k \log \Delta)$ bits of memory.*

It is the first example where the number of steps is exponential in k. However, the next result indicates that the gap between the upper bound and lower bound is not so large:

Theorem 12. *For $k < \log_{\delta-1} n$, there exists a distribution of k liars in the tree of bounded degree δ with n vertices so that the number of steps required to find s is at least $d + \Omega((\delta - 1)^k)$.*

References

[1] Y. Afek, E. Gafni, and M. Ricklin, Upper and lower bounds for routing schemes in dynamic networks, in: Proc. 30th Symposium on Foundations of Computer Science, (1989), 370–375.

[2] S. Albers and M. Henzinger, Exploring unknown environments, *in Proc. 29th Symposium on Theory of Computing*, (1999), 416–425.

[3] B. Awerbuch, B. Patt-Shamir, and G. Varghese, Self-stabilizing end-to-end communication, *Journal of High Speed Networks* **5** (1996), 365–381.

[4] R.A. Baeza-Yates, J.C. Culberson and G.J.E Rawlins, Searching in the plane, *Information and Computation* **106(2)** (1993), 234–252.

[5] D. Bienstock and P. Seymour, Monotonicity in graph searching, *Journal of Algorithms* **12** (1991), 239–245.

[6] P.J. Cameron. *Topics, Techniques, Algorithms.* Cambridge University Press, 1994.

[7] R. Cole, B. Maggs and R. Sitaraman, Routing on butterfly networks with random faults, *in: Proc. 36th Symposium on Foundations of Computer Science*, (1995), 558–570.

[8] S. Dolev, E. Kranakis, D. Krizanc and D. Peleg, Bubbles: Adaptive routing scheme for high-speed networks, *SIAM Journal on Computing*, to appear.

[9] E. Kranakis and D Krizanc, Searching with uncertainty, *in: Proc. 6th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, (1999), C. Gavoille, J.-C, Bermond, and A. Raspaud, eds., pp, 194-203, Carleton Scientific, 1999.

[10] L.M. Kirousis, E. Kranakis, D. Krizanc, and Y.C. Stamatiou. Locating information with uncertainty in fully interconnected networks. unpublished paper, (1999).

[11] E. Kushilevitz and Y. Mansour, Computation in noisy radio networks, *in Proc. 9th Symposium on Discrete Algorithms*, 1998, 236–243.

[12] L. Kirousis and C. Papadimitriou, Interval graphs and searching, *Discrete Mathematics* **55** (1985), 181–184.

[13] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou, The complexity of searching a graph, *Journal of the ACM* **35** (1988), 18–44.

[14] T. Leighton and B. Maggs, Expanders might be practical, *in: 30th Proc. Symposium on Foundations of Computer Science*, (1989), 384–389.

[15] P. Panaite and A. Pelc, Exploring unknown undirected graphs, *in: Proc. 9th Symposium on Discrete Algorithms*, (1998), 316–322.

[16] Mobile Agents, W. R. Cockayne and M. Zyda, editors, Manning Publications Co., Greenwitch, Connecticut, 1997.
http://www.manning.com/Cockayne/Contents.html