Locating Information with Uncertainty in Fully Interconnected Networks with Applications to World Wide Web Information Retrieval

Alexis C. Kaporis[*,**] Lefteris M. Kirousis [†] Evangelos Kranakis[‡††]

Danny Krizanc[§††] Yannis C. Stamatiou[¶††] Elias C. Stavropoulos[‖‡‡]

April 12, 2001

Abstract

In this paper we examine the problem of searching for some information item in the nodes of a fully interconnected computer network, where each node contains information relevant to some topic as well as links to other network nodes that also contain information, not necessarily related to locally kept information. These links are used to facilitate the Internet users and mobile software agents that try to locate specific pieces of information. However, the links do not necessarily point to nodes containing information of interest to the user or relevant to the aims of the mobile agent. Thus an element of *uncertainty* is introduced. For example, when a Internet user or some search agent lands on a particular network node, they see a set of links that point to information that is, *supposedly*, relevant to the current search. Therefore, we can assume that a link points to relevant information with some *unknown* probability p that, in general, is related to the number of nodes in the network (intuitively, as the network grows, this probability tends to 0 since adding more nodes to the network renders some extant links less accurate or obsolete). Consequently, since there is uncertainty as to whether the links contained in a node's web page are correct or not, a search algorithm cannot rely on following the links systematically since it may

[*]University of Patras, Department of Computer Engineering and Informatics, GR-265 00, Patras, Greece. e-mail: kaporis@ceid.upatras.gr

[†]University of Patras, Department of Computer Engineering and Informatics, GR-265 00, Patras, Greece. e-mail: kirousis@ceid.upatras.gr

[‡]Carleton University, School of Computer Science, Ottawa, ON, K1S 5B6, Canada. e-mail: kranakis@scs.carleton.ca

[§]Wesleyan University, Department of Mathematics, Middletown, CT 06459. e-mail: dkrizanc@wesleyan.edu

[¶]University of Patras, Department of Computer Engineering and Informatics, GR-265 00, Patras, Greece also Computer Technology Institute Kolokotroni 3, GR-262 21 Patras, Greece. e-mail: stamatiu@ceid.upatras.gr

[‖]University of Patras, Department of Computer Engineering and Informatics, GR-265 00, Patras, Greece. e-mail: estavrop@ceid.upatras.gr

[**]Research supported by the University of Patras Research Committee (Project Caratheodory, under contract no. 2445).

[††]Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grant and by MITACS project CANCCOM (Complex Adaptive Networks for Computing and Communication). The fifth author was also supported by the Greek Ministry of National Economy through a NATO scholarship for conducting postdoctoral studies (contract number 106384/ΔΟΟ 1222/2-7-98).

[‡‡]Research supported by the Hellenic Ministry of Development, General Secretariat for Research and Technology (Project ΠΕΝΕΔ 99, under contract no. 99 ΕΔ 232). Also supported by the University of Patras Research Committee (Project Caratheodory, under contract no. 1939).

end up spending too much time visiting nodes that contain irrelevant information. In this work, we will describe and analyze a search algorithm that is only allowed to transfer a fixed amount of memory along communication links as it visits the network nodes. The algorithm is, however, allowed to use one bit of memory at each node as an "already visited" flag. In this way the algorithm has its memory distributed to the network nodes, avoiding overloading the network links as it moves from node to node searching for the information. We work on fully interconnected networks for simplicity reasons and moreover, because, according to some recent experimental evidence, such networks can be considered to be a good approximation of the current structure of the World Wide Web.

Key Words and Phrases: Graph search with uncertainty, Random walks in graphs, Distributed systems, World Wide Web search.

1 Introduction

The World Wide Web (or, for short, Web) grows daily by a million electronic pages that are added to the hundreds of millions pages already on-line. The rapid and chaotic growth of the Web makes information retrieval a challenging task. Gathering reliable information relevant to a specific topic of interest or taking accurate statistics about the web pages is difficult due to the lack of apparent structure and organization. The Web can be viewed as a huge directed graph where each node represents a page and each edge from one node to another represents a hyperlinks from a page to another. Recent experimental studies suggest that this Web graph is composed by four components: first, there is a giant strongly connected component where every two nodes are reachable from one another through some path in the graph (a sequence of links). This is the *main* Web, the part of the Web that is the *most well known*. Then there is the *old* part of the Web whose pages are reachable from the main part but its nodes cannot reach the main part. Also there is the *new* part, containing pages that can reach the main part but are not reachable from the main part. Finally, there is another component containing the rest of the pages, other small connected components and pages that link to the old part or that are linked by the new part. The main part and the old part of the web graph are the, so called, *indexable Web*. This is the part that can be easily explored by most Web users and search engines and contains the most useful content of the Web. This model of the Web has been adopted and used for the approximation of aggregate and selection queries about Web pages via random walks, mainly in the indexable Web [5]. Small modifications can be made to the natural Web graph to obtain an undirected, regular, and strongly connected graph. A crucial observation is that the *minimum* number of hops for moving from a node to any other node in the Web is bounded from above by a small constant [1]. Since this constant is very small compared to the size of the Web, it would not be unrealistic to view the indexable Web as a *fully interconnected graph* or *clique*. Moreover, the structure of the Web graph exhibits the *small world* properties defined in [22] that measure the separation between two nodes of the graph (*characteristic path length*) and the cliquishness of a typical neighborhood of it (*clustering coefficient*). These properties can be used to obtain more relevant and accurate documents as a response to a query and have been recently exploited by search engines to assess the connectedness between communities in the Web (see [1, 12]).

Led by the above considerations, our starting point is that searching for some information item in the Web can be modeled as searching for information in a fully interconnected network. Thus, in this paper we study the problem of searching for an information item in a completely interconnected computer network when, in addition, the search algorithms are constrained to transfer along communication links only a fixed amount of memory, no matter how large the network is. This is a realistic assumption since it is not desirable to have mobile search agents that transfer unbounded amounts of memory along communication links since this would increase the flow of data and overload the network. To aid the algorithms' search, each network node contains a number of links that, when followed by the algorithms, will lead to some node of the network that contains some, hopefully, useful information. However, some of the links may not be properly updated or they may lead to information that is not useful with respect to a specific query. Therefore, we can assume that the links lead to the desired information with some bounded probability p that may depend on the number of nodes in the network. This has as a consequence that the responses of some of the nodes are misleading and, ideally, should not be followed by the search algorithms. However, a natural assumption is that the algorithms are unaware not only of which nodes possess useful links but also of how many of them exist in the network. This "searching with uncertainty" was introduced in [13] for ring interconnected and toroidal interconnection networks. The authors of that paper gave some high-probability results about the number of steps required to reach the information node and had proposed as a research question to study the problem on fully interconnected networks, a question that was addressed in [11]. In this latter paper, various classes of algorithms were considered for searching in the complete network and they were classified according to whether they have unbounded or fixed memory and whether they can use randomness or not. A general lower bound of $1/p$ was proved for the expected number of steps for any search algorithm in the complete network and it was shown that no deterministic, fixed memory algorithm can terminate in a finite expected number of steps. It was also shown that deterministic and randomized algorithms require the same expected number of steps, namely $O(1/p)$. In all variants however, as opposed to the variant considered in this paper, the search algorithms transferred their memory along the network links. Since the amount of this memory can be as large as the number of nodes, this may cause increased link traffic.

Search problems that incorporate some degree of uncertainty or nondeterminism have been considered extensively in the past. One of the best known and most thoroughly studied is the routing problem for various computer network topologies in the presence of *faulty* nodes (see [2, 4, 8, 9, 14]). The routing problem differs from the problem we consider here in that in the former problem the identity of the target node is known in advance and what is required is to follow a shortest path to reach it, when some of the nodes or the interconnections may be faulty. On the other hand, graph search problems where the identity of the target node is not known have been considered in the past in *deterministic search games*, where a fugitive with some characteristics that define the search variant (e.g. it may be moving constantly or it may move only when some searcher is about to visit the node at which it resides) hides in the nodes of the graph and the aim is to capture it using the smallest possible number of searchers (see, for example, [6, 10, 15]). Deterministic algorithms that explore graphs with unknown topology have been studied in [3, 17]. There also exists a class of games called *stochastic games* in which the opponents employ move strategies involving a probability transition matrix (see [20]). A description of games that involve searching for hidden subsets of a set, called *Geometric Games*, is given in [21]. Moreover, searching with uncertainty (by asking an oracle) in sets of distinct items has also been considered (see [7, 16, 18, 19], for example).

In what follows, we will describe an algorithm for locating an information item that resides in one of the nodes of a fully interconnected network. The algorithm exploits the nodes' links to obtain help in the identification of this node. We will assume that a link in a node returns a correct link to the information node with probability p which, in general, may depend on the number of nodes. At each step, and as long as the information node has not been found, the algorithm queries the currently visited node. Then the algorithm sets a "already visited" flag in the current node and follows the link returned. If at a later step it visits again the same node, it refrains from following the same wrong link by reading the "already visited" flag and selects the next node at random. This is also what an internet surfer would do if it has followed a wrong link and after some steps it returns back to the web page that contained the wrong link.

In the next section we first describe the network model we adopt and how the nodes that provide incorrect information are determined. Then we give the algorithm and analyze its behaviour. Since in practice the number of the links may differ from a node to another, the analysis that follows corresponds to the worst case. Also, for convenience in the analysis, for all the nodes we have the same probability p that their links lead to the desired information. This corresponds to choosing, as a worst case, the *smallest* probability over all the nodes in the network (assuming $p > 0$).

2 The computer network model with the uncertainty element

The computer network is modeled as a complete graph on n nodes, i.e. there exist communication links between every pair of nodes. Each node contains (i) an identity number and (ii) a link that provides the identity (network address) of the node that stores some information item. However, the link points to the *right* node (or, *wrong* node) only with some probability p (or, $q = 1 - p$, respectively) that may depend on the number of nodes in the network. To model this uncertainty, we do the following: (i) Initially, a node is chosen at random and is considered to possess the information item. The links of the node are set to point to the node itself. (ii) Each of the remaining nodes are chosen with probability p to contain a correct link to the information node. Those who are not chosen, select at random a node other than the information node and themselves and set their link to point to it. No dynamic changes are permitted as the search algorithm operates, after the above have been determined.

For convenience in the analysis of our algorithm, we assume that the information node is an extra node outside the complete interconnected network (which contains n nodes). Under this assumption, a node of the network is characterized either as *honest* or *liar*. A node is called *honest* if its link provides a correct link to the information node; otherwise the node is called *liar*. This computer network is equivalent to the previous one: locating the information node is now equivalent to reaching an honest node. Notice that this reformulation is equivalent for our algorithm, not for any arbitrary one. An important consequence of this simplification is that we require that at least *one* honest node exists in the network since, otherwise, the algorithm would never terminate. This is, however, not too restrictive since in what follows we will require $p = \omega(1/n)$ so that the expected number of honest nodes is $pn >> 1$.

3 A distributed memory algorithm

In this section we will present an algorithm whose memory is distributed among the network nodes. The only help it receives when it reaches a node is a single bit which is initially set to 0 (we say that it visits a *white* node) and the algorithm changes to 1 (a *red* node) when it visits the node for the first time. Therefore, whenever the algorithm visits a red node, it knows it should not follow the node's (wrong) directions to the information node. The whole procedure ends when an honest node is found. Of course the algorithm still has to follow the link to finish (using one more step), but this will not affect our analysis.

Algorithm: Search with Distributed Memory Bits (SDMB)
Input : A complete network (V, E) and a node designated as the information node
Aim: Find the information node

```
1.  begin
2.     reds ← 0 /* Counts the number of visited nodes. */
3.     current = RANDOM(V) /* The currently visited node. */
4.     steps ← 0
5.     repeat
6.       if current(color) = 0 begin /* A node not previously visited – "white" node. */
7.          current(color) ← 1
8.          reds ← reds + 1
9.          current ← current(link)
10.         steps ← steps + 1
11.      end
12.      else begin /* An already visited node – don't follow the link. */
13.         current ← RANDOM(V-current)
14.         steps ← steps + 1
15.      end
16.    until current(information) = true /* Information node found. */
17. end
```

When lines 10 or 14 are executed, the current step is considered complete and the variable *steps* is, thus, increased. When we say "at step $i > 0$" we refer to the stage of the execution of the algorithm where lines 9 or 13 have just been executed. Also observe that the number of steps taken by algorithm SDMB is unbounded. However, the *expected* number of steps is bounded and close to $1/p$.

Let us denote by q_i the *failure probability at step* $i > 0$, conditional upon failure in all previous step values, $0, 1, \ldots, i-1$. We also denote by r_i the number of "red" nodes *after steps* has assumed the value i. Obviously, $r_0 = 0$, $r_1 = 1$, and $r_2 = 2$. We also define the following two (mutually exclusive) events:

- F_w^i: Failure at step i *after* the execution of lines 7–9 (i.e. failure after following the link given by a previously white node).
- F_r^i: Failure at step i *after* the execution of line 13 (i.e. failure after encountering a red node and randomly selecting one of the remaining ones).

By "$\text{Step}_i = \mathit{white}$ (red)" we denote that after the execution of the i-th step the algorithm has visited a white (red) node while by "$\text{Step}_i \to \mathit{liar}$ (honest)" we denote that at the execution of the i-th step the algorithm will visit a liar (honest) node. Let also F_i denote failure at step i and E_i failure at steps $0, 1, \ldots, i$. (When $i = 0$ then $E_i = \emptyset$, the empty event.) Then:

$$\begin{aligned} q_i &= \mathbf{Pr}[\text{failure at step} = i \mid \text{failure at steps } 0, 1, \ldots, i-1] \\ &= \mathbf{Pr}[F_i \mid E_{i-1}] = \mathbf{Pr}[F_w^i \mid E_{i-1}] + \mathbf{Pr}[F_r^i \mid E_{i-1}]. \end{aligned} \tag{1}$$

Let us examine the first probability that appears in (1). By definition, the probability of the event F_w^i, under the condition E_{i-1}, is equal to the probability that at step $i-1$ the algorithm has reached a white node and this node is a liar (it gives an incorrect link to the information node). That is,

$$\mathbf{Pr}[F_w^i \mid E_{i-1}] = \mathbf{Pr}[\text{Step}_{i-1} = \mathit{white} \mid E_{i-1}] \cdot \mathbf{Pr}[\text{Step}_i \to \mathit{liar} \mid \text{Step}_{i-1} = \mathit{white}, E_{i-1}].$$

A liar node may be either one of the already red nodes of the network or one of the white ones. Thus,

$$\begin{aligned}
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] \\
= \; & \mathbf{Pr}[\mathrm{Step}_i \to \mathit{red\ liar} \mid \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] + \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{white\ liar} \mid \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] \\
= \; & \mathbf{Pr}[\mathrm{Step}_i \to \mathit{red} \mid \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] \cdot \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_i = \mathit{red}, \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] + \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{white} \mid \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] \cdot \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_i = \mathit{white}, \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] \\
= \; & \frac{r_{i-1} - 1}{n-1} \cdot 1 + \frac{n - 1 - (r_{i-1} - 1)}{n-1} \cdot q \, .
\end{aligned}$$

Therefore:

$$\mathbf{Pr}[F_w^i \mid E_{i-1}] = \mathbf{Pr}[\mathrm{Step}_{i-1} = \mathit{white} \mid E_{i-1}] \cdot \left[\frac{r_{i-1} - 1}{n-1} + q \left(1 - \frac{r_{i-1} - 1}{n-1} \right) \right].$$

The second probability in (1), under the condition E_{i-1}, is equal to the probability that at step $i-1$ the algorithm has reached a red node and this node is a liar, i.e.,

$$\mathbf{Pr}[F_r^i \mid E_{i-1}] = \mathbf{Pr}[\mathrm{Step}_{i-1} = \mathit{red} \mid E_{i-1}] \cdot \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}]$$

Using the same analysis as above, we have that

$$\begin{aligned}
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}] \\
= \; & \mathbf{Pr}[\mathrm{Step}_i \to \mathit{red\ liar} \mid \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}] + \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{white\ liar} \mid \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}] \\
= \; & \mathbf{Pr}[\mathrm{Step}_i \to \mathit{red} \mid \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}] \cdot \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_i = \mathit{red}, \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}] + \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{white} \mid \mathrm{Step}_{i-1} = \mathit{red}, E_{i-1}] \cdot \\
& \mathbf{Pr}[\mathrm{Step}_i \to \mathit{liar} \mid \mathrm{Step}_i = \mathit{red}, \mathrm{Step}_{i-1} = \mathit{white}, E_{i-1}] \\
= \; & \frac{r_{i-1} - 1}{n-1} \cdot 1 + \frac{n - 1 - (r_{i-1} - 1)}{n-1} \cdot q \, .
\end{aligned}$$

Therefore:

$$\mathbf{Pr}[F_r^i \mid E_{i-1}] = \mathbf{Pr}[\mathrm{Step}_{i-1} = \mathit{red} \mid E_{i-1}] \cdot \left[\frac{r_{i-1} - 1}{n-1} + q \left(1 - \frac{r_{i-1} - 1}{n-1} \right) \right].$$

By adding the above probabilities we have,

$$\begin{aligned}
q_i \; = \; & \mathbf{Pr}[\mathrm{Step}_{i-1} = \mathit{white} \mid E_{i-1}] \cdot \left[\frac{r_{i-1} - 1}{n-1} + q \left(1 - \frac{r_{i-1} - 1}{n-1} \right) \right] + \\
& \mathbf{Pr}[\mathrm{Step}_{i-1} = \mathit{red} \mid E_{i-1}] \cdot \left[\frac{r_{i-1} - 1}{n-1} + q \left(1 - \frac{r_{i-1} - 1}{n-1} \right) \right] \\
= \; & q + \frac{r_{i-1} - 1}{n-1} \, (1 - q). \qquad (2)
\end{aligned}$$

We will use this expression in the proof of Theorem 4.2 where we will need the fact that q_i is a nondecreasing function of i since r_i is nondecreasing.

Expression (2) also quantifies the intuition that during the first steps, where r_i is small compared to n, the distributed memory algorithm fails with probability q. At later steps, as r_i increases towards n, the probability of failure gradually increases from q to 1, as expected.

4 The dynamic behaviour of the distributed memory algorithm

Let X be the random variable that counts the number of liar nodes in the complete network. Obviously, the random variable X follows the binomial distribution with parameters n and q. Consider now the conditional subspace of all completely interconnected networks containing $X = k$ liars, $k < n$. We shall show that if we apply the SDMB algorithm in a network with k liars, the probability of failure of the algorithm at each step does not depend on the evolution of the number of red nodes.

Initially, we randomly choose one node from the n possible nodes of the network as the starting node of the algorithm (this is step 0). The probability of the starting node being a liar is k/n. Let S be the random variable that counts the numbers of steps of the algorithm, $S = 1, 2, \ldots$ (exactly the same as the variable *steps* in the algorithm, only shorter for convenience). Then the following holds:

Theorem 4.1 *Suppose that the complete network contains $X = k$ liars, $2 \leq k < n$. Then the probability that the algorithm terminates after t steps, $t \geq 2$, is:*

$$\mathbf{Pr}[S = t \mid X = k] = \frac{k}{n} \left(\frac{k-1}{n-1}\right)^{t-2} \left(1 - \frac{k-1}{n-1}\right),$$

while the above probability is equal to $(1 - k/n)$ when $t = 1$. Moreover,

$$\mathbf{Pr}[S = t \mid X = 0] = \begin{cases} 1 \ , & for\ t = 1, \\ 0 \ , & for\ t > 1, \end{cases}$$

and

$$\mathbf{Pr}[S = t \mid X = 1] = \begin{cases} 1 - 1/n \ , & for\ t = 1, \\ 1/n \ , & for\ t = 2, \\ 0 \ , & for\ t > 2. \end{cases}$$

Proof The cases $k = 0, 1$ are easy. The case where $k \geq 2$ and $t = 1$ is easy too. Therefore, we will be concerned with the case where $k \geq 2$ and $t \geq 2$.

The probability that the algorithm terminates after t steps is equal to the probability that in the $(t-1)$-th step the algorithm reaches an honest node for the first time, i.e., a node that points to the information node. Thus,

$$\begin{aligned} \mathbf{Pr}[S = t \mid X = k] &= \mathbf{Pr}[\text{Step}_0 \to liar \wedge \cdots \wedge \text{Step}_{t-2} \to liar \wedge \text{Step}_{t-1} \to honest \mid X = k] \\ &= \mathbf{Pr}[\text{Step}_0 \to liar \mid X = k] \cdot \\ &\quad \mathbf{Pr}[\text{Step}_1 \to liar \mid \text{Step}_0 = liar, X = k] \cdot \ldots \cdot \\ &\quad \mathbf{Pr}[\text{Step}_{t-2} \to liar \mid \text{Step}_0 = liar \wedge \cdots \wedge \text{Step}_{t-3} = liar, X = k] \cdot \\ &\quad \mathbf{Pr}[\text{Step}_{t-1} \to honest \mid \text{Step}_0 = liar \wedge \cdots \wedge \text{Step}_{t-2} = liar, X = k]. \end{aligned}$$

Since at Step_0 the starting node is chosen uniformly at random, the probability this node being a liar is $\mathbf{Pr}[\text{Step}_0 \to \mathit{liar} \mid X = k] = k/n$. For the computation of $\mathbf{Pr}[\text{Step}_1 \to \mathit{liar} \mid \text{Step}_0 = \mathit{liar}, X = k]$, notice that since the node mentioned in the conditional is white, the algorithm has to follow it's link. Now each liar node had, initially, set its link to point to a randomly selected node of the network. The probability that the selected node was a liar is therefore $\frac{k-1}{n-1}$. Thus, $\mathbf{Pr}[\text{Step}_1 \to \mathit{liar} \mid \text{Step}_0 = \mathit{liar}, X = k] = \frac{k-1}{n-1}$. To compute $\mathbf{Pr}[\text{Step}_2 \to \mathit{liar} \mid \text{Step}_0 = \mathit{liar} \wedge \text{Step}_1 = \mathit{liar}]$ observe that at Step_1 there exists one red (i.e. already visited) node (the starting node) and the current node is thus liar (from the conditional) and white. Since the node is white, the algorithm again follows the node's link. Therefore $\mathbf{Pr}[\text{Step}_2 \to \mathit{liar} \mid \text{Step}_{0,1} = \mathit{liar}, X = k] = \frac{k-1}{n-1}$.

Consider now the case where the algorithm is at step 3. The complication is that the current node may be now either white or red. Therefore the probability $\mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k]$ must be decomposed as follows:

$$\begin{aligned}
& \mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k] \\
= \ & \mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \wedge \text{Step}_2 = \mathit{white} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k] + \\
& \mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \wedge \text{Step}_2 = \mathit{red} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k] \\
= \ & \mathbf{Pr}[\text{Step}_2 = \mathit{white} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k] \\
& \cdot \mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, \text{Step}_2 = \mathit{white}, X = k] + \\
& \mathbf{Pr}[\text{Step}_2 = \mathit{red} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k] \\
& \cdot \mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, \text{Step}_2 = \mathit{red}, X = k].
\end{aligned}$$

When a white node is encountered, the algorithm follows the node's link. Since this node is a liar, the link is not correct and the node will point to one of the remaining $n-1$ nodes of the network $k-1$ of which are liars. Thus,

$$\mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, \text{Step}_2 = \mathit{white}, X = k] = \frac{k-1}{n-1}.$$

On the other hand, when a red node is reached, the algorithm doesn't follow the node information and uniformly it chooses a node among the rest $n-1$ nodes, to visit. Since the current node is a liar,

$$\mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, \text{Step}_2 = \mathit{red}, X = k] = \frac{k-1}{n-1}.$$

Notice also that the probabilities $\mathbf{Pr}[\text{Step}_2 = \mathit{white} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k]$ and $\mathbf{Pr}[\text{Step}_2 = \mathit{red} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k]$ are complementary and their sum is equal to 1. By adding the above two probabilities, we have that

$$\mathbf{Pr}[\text{Step}_3 \to \mathit{liar} \mid \text{Step}_{0,1,2} = \mathit{liar}, X = k] = \frac{k-1}{n-1}.$$

It is easy to see that this observation is also true for the next $t-1$ steps of the algorithm.

We will now compute the probability that the algorithm reaches an honest node at the t-th step, an event that occurs when the test in line (16) of the algorithm is successful. This probability is decomposed as follows:

$$\mathbf{Pr}[\text{Step}_t \to \mathit{honest} \mid \text{Step}_{0,1,\ldots,t-1} = \mathit{liar}, X = k]$$

$$
\begin{aligned}
= \quad & \mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \wedge \mathrm{Step}_{t-1} = \mathit{white} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, X = k] + \\
& \mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \wedge \mathrm{Step}_{t-1} = \mathit{red} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, X = k] \\
= \quad & \mathbf{Pr}[\mathrm{Step}_t = \mathit{white} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, X = k] \\
& \cdot \mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, \mathrm{Step}_{t-1} = \mathit{white}, X = k] + \\
& \mathbf{Pr}[\mathrm{Step}_t = \mathit{red} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, X = k] \\
& \cdot \mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, \mathrm{Step}_{t-1} = \mathit{red}, X = k].
\end{aligned}
$$

The probability the algorithm reaches an honest node starting from a white node that is liar is:

$$
\begin{aligned}
\mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, \mathrm{Step}_{t-1} = \mathit{white}, X = k] &= \frac{(n-1)-(k-1)}{n-1} \\
&= 1 - \frac{k-1}{n-1}.
\end{aligned}
$$

When the node is red, the probability of reaching a honest node remains the same:

$$
\begin{aligned}
\mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \mid \mathrm{Step}_{0,1,\ldots,t-1} = \mathit{liar}, \mathrm{Step}_{t-1} = \mathit{red}, X = k] &= \frac{(n-1)-(k-1)}{n-1} \\
&= 1 - \frac{k-1}{n-1}.
\end{aligned}
$$

Thus,

$$
\mathbf{Pr}[\mathrm{Step}_t \to \mathit{honest} \mid \mathrm{Step}_{0,1,\ldots t-1} = \mathit{liar}, X = k] = 1 - \frac{k-1}{n-1}.
$$

This completes the proof of the theorem. ■

In the following theorem, we will prove that if $t = o(1/p)$ and $p = \omega(\frac{1}{n})$ then SDMB locates the information node in at most t steps, almost surely.

Theorem 4.2 *Let $p = \omega(\frac{1}{n})$ and $p \to 0$. Then, the probability that SDMB succeeds in at most t steps, $t \geq 2$, tends to 1 when $t = \omega(1/p)$ and to 0 when $t = o(1/p)$.*

Proof The probability that the SDMB algorithm succeeds in at most t steps is 1 minus the probability that it does not succeed in any of the steps $0, 1, \ldots, t$:

$$
\begin{aligned}
& \mathbf{Pr}[\text{SDMB succeeds in at most } t \text{ steps}] = \\
& \quad 1 - \mathbf{Pr}[\text{SDMB fails at steps } 0, 1, \ldots t] = \\
& \quad 1 - \mathbf{Pr}[\text{SDMB fails at steps } 0, 1, \ldots t \mid X = 0] \cdot \mathbf{Pr}[X = 0] \\
& \quad\quad - \mathbf{Pr}[\text{SDMB fails at steps } 0, 1, \ldots t \mid X = 1] \cdot \mathbf{Pr}[X = 1] \\
& \quad\quad - \sum_{k=2}^{n-1} \mathbf{Pr}[\text{SDMB fails at steps } 0, 1, \ldots t \mid X = k] \cdot \mathbf{Pr}[X = k].
\end{aligned}
$$

Observe that

$$
\mathbf{Pr}[\text{SDMB fails at steps } 0, 1, \ldots t \mid X = 0] = 0
$$

and

$$
\begin{aligned}
& \mathbf{Pr}[\text{SDMB fails at steps } 0, 1, \ldots t \mid X = 1] \\
& = \mathbf{Pr}[\text{SDMB fails at step } 0 \mid X = 1] \cdot \\
& \quad \mathbf{Pr}[\text{SDMB fails at step } 1 \mid \text{SDMB failed at step } 0, X = 1] \cdot \ldots \\
& = 1/n \cdot 0 \cdot \ldots = 0.
\end{aligned}
$$

Thus,

$$\begin{aligned}
&\mathbf{Pr}[\text{SDMB succeeds in at most } t \text{ steps}] = \\
&1 - \sum_{k=2}^{n-1} \mathbf{Pr}[\text{Step}_0 \to \mathit{liar} \wedge \cdots \wedge \text{Step}_{t-1} \to \mathit{liar} \wedge \text{Step}_t \to \mathit{liar} \mid X = k] \cdot \mathbf{Pr}[X = k] = \\
&1 - \sum_{k=2}^{n-1} \frac{k}{n} \left(\frac{k-1}{n-1}\right)^{t-2} \binom{n}{k} q^k (1-q)^{n-k}. \qquad (3)
\end{aligned}$$

In general, the number of liar nodes is a fraction $\alpha \in (0,1)$ of the total number of nodes of the network, i.e. $\alpha = k/n$. By applying Stirling's approximation for the factorial function, it can be easily proved that $\binom{n}{k} = \binom{n}{\alpha n} \sim \frac{1}{\sqrt{2\pi n \alpha(1-\alpha)}} \cdot \left[\frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}}\right]^n$. We set $k = \alpha n$ in (3) and obtain the following:

$$\begin{aligned}
&\mathbf{Pr}[\text{SDMB succeeds in at most } t \text{ steps}] \sim \\
&1 - \sum_a \alpha \alpha^{t-2} \frac{1}{\sqrt{2\pi n \alpha(1-\alpha)}} \left[\left(\frac{q}{\alpha}\right)^\alpha \left(\frac{1-q}{1-\alpha}\right)^{1-\alpha}\right]^n. \qquad (4)
\end{aligned}$$

Our goal is to show that if the number of steps performed by the algorithm is $t = \omega(\frac{1}{p})$ then the probability in (4) converges to 1. To this end, it suffices to have the sum in (4) converge to 0. We observe that this sum contains n terms (that is, *polynomially many*) of the form

$$\alpha \alpha^{t-2} \frac{1}{\sqrt{2\pi n \alpha(1-\alpha)}} \left[\left(\frac{q}{\alpha}\right)^\alpha \left(\frac{1-q}{1-\alpha}\right)^{1-\alpha}\right]^n$$

for various values of α in the interval $(0,1)$. Thus, for the sum in (4) to converge to 0 it suffices that all the terms of the above form vanish exponentially fast. For the exponential (in n) factors it holds that

$$\left[\left(\frac{q}{\alpha}\right)^\alpha \left(\frac{1-q}{1-\alpha}\right)^{1-\alpha}\right]^n \to \begin{cases} 0, & \alpha \neq q \\ 1, & \alpha = q. \end{cases}$$

Therefore, factors with $\alpha \neq q$ vanish *exponentially* fast, guaranteeing convergence to 0 of the terms that contain them. For factors with $\alpha = q$, it suffices to set t in a way such that the other factor, α^t, of the term vanishes:

$$\alpha^t = q^t = (1-p)^t \leq e^{-pt} \to 0,$$

using the assumption $t = \omega(\frac{1}{p})$. This establishes the convergence to 0 of the terms with $\alpha = q$ and, consequently, of all terms of the sum in (4).

¿From the expression for q_i in Equation (2), $q_i = q + \frac{r_{i-1}-1}{n-1}\,(1-q)$, it follows that $q_i \geq q$ for all i. Therefore,

$$\mathbf{Pr}[\text{SDMB succeeds in at most } t \text{ steps}] = 1 - \prod_{i=0}^{t} q_i \leq 1 - q^t = 1 - (1-p)^t. \qquad (5)$$

Since $e^{-x} \sim 1 - x$ when $x = o(1)$ we have that $(1-p)^t \sim e^{-pt}$. Assuming that $t = o(\frac{1}{p})$ we have $e^{-pt} \to 1$ which, in turn, drives the probability in (5) to 0 completing the proof of the theorem. ∎

We next prove that the expectation of the number of steps of the algorithm SDMB is $\Theta(1/p)$.

Theorem 4.3 *The expected number of steps for algorithm SDMB is, asymptotically,* $\Theta(\frac{1}{p})$ *when* $p = \omega(\frac{1}{n})$.

Proof The expectation of the random variable S than counts the number of steps in lines 10 and 14 of the algorithm is the following:

$$\begin{aligned}
\mathbf{E}[S] &= \mathbf{E}[S \mid X = 0] \cdot \mathbf{Pr}[X = 0] + \\
&\quad \mathbf{E}[S \mid X = 1] \cdot \mathbf{Pr}[X = 1] + \cdots + \\
&\quad \mathbf{E}[S \mid X = n-1] \cdot \mathbf{Pr}[X = n-1], \qquad (6)
\end{aligned}$$

with

$$\mathbf{Pr}[X = k] = \binom{n}{k} q^k (1-q)^{n-k}.$$

¿From Theorem 4.1, when $k = 0, 1$, it holds that $\mathbf{E}[S \mid X = 0] = 1$ and $\mathbf{E}[S \mid X = 1] = 1 + 1/n$. Moreover, from the same theorem, when $X = k, 2 \le k < n$, we obtain the following:

$$\begin{aligned}
\mathbf{E}[S \mid X = k] &= \sum_{i=1}^{\infty} i \mathbf{Pr}[S = i \mid X = k] = 1 - \frac{k}{n} + \sum_{i=2}^{\infty} i \mathbf{Pr}[S = i \mid X = k] \\
&= 1 - \frac{k}{n} + \sum_{i=2}^{\infty} i \frac{k}{n} \left(\frac{k-1}{n-1}\right)^{i-2} \left(1 - \frac{k-1}{n-1}\right) \\
&= 1 - \frac{k}{n} + \left(\frac{k}{n} + \frac{n-1}{n} \frac{k}{n-k}\right) \\
&= 1 + \frac{k}{n-k+1} + O(\frac{1}{n}). \qquad (7)
\end{aligned}$$

Therefore, the expectation of S is as follows:

$$\begin{aligned}
\mathbf{E}[S] &= (1-q)^n + \left(1 + \frac{1}{n}\right) nq(1-q)^{n-1} \\
&+ \sum_{k=2}^{n-1} \left(1 + \frac{k}{n-k+1} + O(\frac{1}{n})\right) \binom{n}{k} q^k (1-q)^{n-k} \\
&= (1-q)^n + \left(1 + \frac{1}{n}\right) nq(1-q)^{n-1} \\
&+ \sum_{k=2}^{n-1} \binom{n}{k} q^k (1-q)^{n-k} + \sum_{k=2}^{n-1} \left(\frac{k}{n-k+1} + O(\frac{1}{n})\right) \binom{n}{k} q^k (1-q)^{n-k}. \qquad (8)
\end{aligned}$$

After some straightforward algebraic manipulations and using the fact that $p = \omega(1/n)$ in order to have some terms converge to 0, Expression (8) can be written as follows (remember $p = 1 - q$):

$$\begin{aligned}
& 1 + \sum_{k=2}^{n-1} \frac{k}{n-k+1} \binom{n}{k} q^k (1-q)^{n-k} + O(\frac{1}{n}) \\
&= 1 + \sum_{k=2}^{n-1} \binom{n}{k-1} q^k (1-q)^{n-k} + O(\frac{1}{n}) \\
&= 1 + \frac{q}{1-q} + O(\frac{1}{n}) = \frac{1}{p} + O(\frac{1}{n}).
\end{aligned}$$

Thus,

$$\mathbf{E}[S] = \frac{1}{p} + O(\frac{1}{n})$$

which, since $p = \omega(1/n)$, completes the proof of the theorem. ∎

5 Conclusions and directions for further research

Searching for a piece of information in the World Wide Web can be considered as searching in a fully interconnected network with each node representing a web page and a directed edge between two different nodes modeling a hyperlink from one node to the other. To model and investigate this situation, in this paper we studied the problem of searching for an information item stored in a node of a completely interconnected network with n nodes each of which, when queried, points to the node that actually possesses the information with some bounded probability p (generally a function of n). The algorithms considered are only allowed to transfer a fixed amount of memory along communication links (in contrast to [11]). However, they can use a small amount of memory (1 bit) at each network node (i.e. their memory is distributed to network nodes) simply to remember whether the node was visited before or not so that a wrong link is not followed twice. Assuming that $p = \omega(1/n)$ in order to avoid the situation where with high probability no node in the network points to the information node (a fact stemming from a straightforward application of Markov's inequality), we described and analyzed an algorithm that locates the information node in $\Theta(1/p)$ expected number of steps. Finally, it would be interesting to consider the problem of locating an information item in more general classes of graphs and, in particular, d-regular interconnection networks.

References

[1] L. Adamic (1999) The Small World Web. In *Proc. of the European Conference on Digital Libraries*, Paris, France, September 22-24, 443–452.

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

[3] S. Albers and M. Henzinger (1999) Exploring unknown environments. In *Proc. 29th Symposium on Theory of Computing*, El Paso, Texas, USA, 416–425.

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

[5] Ziv Bar-Yossef, Alexander Berg, Steve Chien, Jittat Fakcharoenphol, and Dror Weitz (2000) Approximating Aggregate Queries about Web Pages via Random Walks. In *Proc. 26th International Conference on Very Large Databases*, 535–544.

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

[7] F. Cicalese and U. Vaccaro (1999) Optimal strategies against a liar. *Theoretical Computer Science*, **230** , 167-193.

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

[9] S. Dolev, E. Kranakis, D. Krizanc and D. Peleg (1999) Bubbles: Adaptive routing scheme for high-speed networks. *SIAM Journal on Computing*, **29 (3)**, 804-833.

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

[11] L.M. Kirousis, E. Kranakis, D. Krizanc, and Y.C. Stamatiou (2000) Locating Information with Uncertainty in Fully Interconnected Networks. In *Proc. 14th International Symposium on Distributed Computing (DISC 2000)*, Toledo, Spain, October 2000, 283–296.

[12] J. Kleinberg (1999) The Small-World Phenomenon: An Algorithmic Perspective. Cornell Computer Science Technical Report 99-1776.

[13] E. Kranakis and D. Krizanc (1999) Searching with uncertainty. In *Proc. 6th International Colloquium on Structural Information and Communication Complexity (SIROCCO'99)*, 194–203.

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

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

[16] S. Muthukrishnan (1994) On optimal strategies for searching in the presence of errors. In *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 680–688.

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

[18] A. Pelc (1989) Searching with known error probability. *Theoretical Computer Science*, **63**, 185–202.

[19] J. Spencer (1992) Ulam's searching game with a fixed number of lies. *Theoretical Computer Science*, **95**, 307–321.

[20] T.E.S. Raghavan, T.S. Ferguson, T. Parthasapathy, and O.J. Vrieze (eds) (1991) *Stochastic games and related topics.* Kluwer Academic Publishers.

[21] W.H. Ruckle (1983) *Geometric games and their applications.* Research Notes in Mathematics, Pitman Publishing Inc.

[22] D.J. Watts and S.H. Strogatz (1998) Collective dynamics of 'small-world' networks. *Nature*, **393**, 440–442.