Alcom-FT Technical Report Series
ALCOMFT-TR-01-105

Locating Information with Uncertainty in Fully Interconnected Networks: The Case of Non-Distributed Memory

Lefteris M. Kirousis[1⋆] Evangelos Kranakis[2⋆⋆] Danny Krizanc[3⋆⋆⋆] and Yannis C. Stamatiou[4†]

[1] University of Patras, Department of Computer Engineering and Informatics, Rio 26500, Patras, Greece. e-mail: kirousis@ceid.upatras.gr
[2] Carleton University School of Computer Science, Ottawa, ON, K1S 5B6, Canada. email: kranakis@scs.carleton.ca
[3] Wesleyan University, Department of Mathematics, Middletown, CT 06459. e-mail: dkrizanc@wesleyan.edu
[4] University of Patras, Department of Computer Engineering and Informatics, Rio 26500, Patras, Greece and Computer Technology Institute, Kolokotroni 3, GR-262 21 Patras, Greece. e-mail: stamatiu@ceid.upatras.gr

Abstract. We consider the problem of searching for a piece of information in a fully interconnected computer network (or *clique*) by exploiting advice about its location from the network nodes. Each node contains a database that "knows" what kind of documents or information are stored in other nodes (e.g. a node could be a Web server that answers queries about documents stored on the Web). The databases in each node, when queried, provide a pointer that leads to the node that contains the information. However, this information is up-to-date (or correct) with some bounded probability. While, in principle, one may always locate the information by simply visiting the network nodes in some prescribed ordering, this requires a time complexity in the order of the number of nodes of the network. In this paper, we provide algorithms for locating an information node in the complete communication network, that take advantage of *advice* given from network nodes. The nodes may either give correct advice, by pointing directly to the information node, or give wrong advice by pointing elsewhere. On the lower bounds side, we show that no finite memory deterministic algorithm may locate the information node in finite expected number of steps. In addition, if the probability that a node gives correct advice is p and p tends to 0 while it is asymptotically larger than $1/n$, we show that no algorithm may locate the information node in expected number of steps less than $1/p$. In order to study how the expected number of steps is affected by the amount of memory allowed to the algorithms, we give a memoryless randomized algorithm with expected number of steps $4/p$ and a 1-bit randomized algorithm requiring on the average at most $2/p$ steps. Finally, we give an optimal, unlimited memory deterministic algorithm with expected number of steps equal to $1/p$.

Key Words and Phrases: Search, Information Retrieval, Complete Network, Uncertainty, Random Walks.

1 Introduction

Suppose that we have a network of computers interconnected in some particular topology (e.g. ring, mesh, clique etc.) and one of the computers possesses a piece of information. The objective is to design a *software agent* that is able to travel along the communication links of the network and locate the information as fast as possible. An additional element to this picture,

⋆ Partially supported by the Computer Technology Institute.
⋆⋆ 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).
⋆⋆⋆ 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).
† Supported by the Greek Ministry of National Economy through a NATO scholarship (contract number 106384/ΔΟΟ 1222/2-7-98) and by the MITACS project CANCCOM. Also partially supported by the Computer Technology Institute.

that differentiates the search from the usual random search, is that some of the computers, when queried, will respond with the name of the computer that holds the information. However, some other faulty computers, when queried, will give consistently wrong advice as to which computer has the information. (We do not consider intermittent faults that may appear and then disappear in an unpredictable fashion.) In this paper we address the problem of locating a piece of information in the *complete network* using, possibly incorrect, advice from the nodes, where each computer may communicate directly with any other computer.

This variant of *searching with uncertainty*, was introduced in [12], where the network topologies were the *ring* and the *torus*. Models with faulty information in the nodes have been considered before for the problem of routing (see [1, 3, 6, ?,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 in the past. We have the *deterministic search games*, where a fugitive that possesses some properties (it is agile or inert) hides in the nodes or edges of a graph and the aim is to locate it using as few searchers as possible (for definitions and relevant theory see [5, 11, 16]). Also, the problem of exploring an *unknown* graph has been considered (see [2, 17, 18], for example). Closer to the spirit of our work are the search problems that are defined and studied in [4] where the authors consider problems of locating points in the plane using incomplete knowledge about their position. On another front, a number of algorithms have appeared that search a graph relying on some sort of *random walk* along its edges (for a comprehensive and very readable survey, see [15]). We have a very interesting class of games called *stochastic games* in which opponents' strategies incorporate some sort of probability transition matrix (see [21]). Also, in [22] one may find a detailed treatment of *geometric games* where the aim is to locate some element of a hidden random set.

In this work, we show how to locate a piece of information on the complete network of n nodes using the *advice* that they give when queried. The advice is correct, i.e. it directs to the information node, with some bounded probability p that is a function of n. This problem was studied in [13] in the context of World Wide Web searching for memoryless search algorithms that are only allowed to use a single bit per network node to remember whether they have encountered the node before. Such algorithms impose minimal overhead on communication links as they visit the network nodes and they are well suited for information retrieval in communication topologies resembling large cliques.

The class of algorithms we will consider can execute one of the following operations: (i) query a node of the complete network about the location of the information node (ii) follow the advice given by a queried node and (iii) select the next node to visit using some probability distribution function on the network nodes (that may be, for example, a function of the number of previously seen nodes or the number of steps up to now). The algorithms may be either *memoryless*, have *limited memory* or have *unlimited memory*. An algorithm is memoryless, if it may not store any information before it moves from a node to the next one. It is of limited memory, if it can only store a *fixed* amount of information (i.e. independent from the size of the complete network). Finally, an algorithm is of unlimited memory, if it may store any amount of information, usually the identities of the nodes encountered in the past. In addition, an algorithm may use randomization or be deterministic. We show that no deterministic algorithm with fixed memory can locate the information node in a finite expected number of steps. The reason for this is that it may fall into cycles in the space of its states, for sufficiently large cliques. However, we give an optimal deterministic algorithm with unlimited memory

that achieves an expected number of steps asymptotically equal to $\frac{1}{p}$. When randomization is allowed, we give a memoryless randomized algorithm that achieves an expected number of steps bounded from above by $4/p$ while if one bit of memory is allowed we give a randomized algorithm that needs at most $2/p$ steps on the average. Proving a lower bound of $4/p$ for all memoryless algorithms or finding one with a better complexity figure is an interesting question that we believe merits further investigation.

2 The computer network model

In order to present and analyze the various algorithmic approaches to the problem of searching for a piece of information in the complete network, we will define a model of the computer network that captures the fact that some computers give a valid advice (i.e. they point to the node containing the information) while some others give faulty advice by pointing elsewhere.

The computer network is modeled as a clique on n nodes where there is a direct link between any pair of nodes. Each node also contains the following information:

- *Node identity number.* This is a number within the range $1 \dots n$ that uniquely identifies each node.
- *Advice.* This is also a number within the range $1 \dots n$ and it is interpreted as the advice of the node as to which node of the network contains the information.

In order to model the fact that the advice of a node may be incorrect with some probability as well as the fact that the information may reside in any node in a random fashion, we introduce randomness to the model in the following way:

- A node is randomly and uniformly selected from among the n nodes that contain the information and its identity, say s, becomes known to the other nodes. Node s sets its *advice* to be equal to itself, i.e. to contain s. In this way, any algorithm may distinguish a node that contains the information from one that does not.
- Each node, except s, flips a coin that shows heads with probability p and if head shows-up, then the node sets its *advice* equal to s. These nodes give correct advice about the location of the information. If tails appears, then the node randomly and uniformly selects a number from within the set $\{1 \dots n\} - \{s, \mathrm{id}\}$ where *id* is the node's identity number. These nodes are the *faulty* nodes.

After this procedure is carried out, *no changes* are permitted to any piece of information that was determined by the procedure. That is, the sets of faulty and non-faulty nodes remain the same and the wrong advice given by the faulty nodes never changes.

3 Some lower bounds for searching in the clique

Let $K_n = (V_n, E_n)$ be the clique graph on n vertices. In this section, we will prove that if $p = \omega(1/n)$, then no algorithm that searches for a piece of information in K_n (under the model we have defined) can do, asymptotically, better than executing $1/p$ steps on the average. To this end, we will use the fact that given a random set of node $S \subset V_n$ of cardinality k an *adaptive* random walk that starts from a node in $V_n - S$ will hit some node in S in expected time that converges to $\Theta(\frac{n}{k})$, as n tends to infinity. An adaptive random walk is simply a random walk in which after each step, the currently visited node, along with its edges, is deleted from the clique.

Our aim is to show that the problem of locating the information node is at least as difficult as the problem of locating one of k non faulty nodes using a random walk that starts from a randomly chosen initial node. The following proposition holds:

Proposition 1. *For a randomly selected set S of non-faulty nodes, locating the information node $t \in S$ is at least as hard, on the average, as hitting the random set S.*

It is easy to see why this proposition is valid by observing that *any* algorithm that locates the information node in some number of steps may also be used to hit the random set to which the information node belongs in the same number of steps.

Now given a node v in V_n, we denote by $r_v(u)$ the probability of the adaptive random walk choosing vertex $u \neq v$ when it leaves from node v. In the uniform adaptive random walk on K_n, $r_v(u) = \frac{1}{n-1}$ for every v, u. The following can be proved:

Lemma 1. *The uniform adaptive random walk is an optimal algorithm for hitting a randomly selected set of k clique nodes.*

Proof Suppose that an algorithm uses a non-uniform distribution function $r_v(u)$ (that may also be a function of the current step) and that it is about to select the next vertex to visit. Then the probability of failure to select at step i a node from the random set of size k is

$$\begin{aligned} P_v &= \sum_{u \text{ adjacent to } v} r_v(u) \left(1 - \frac{k}{n-i}\right) \\ &= \left(1 - \frac{k}{n-i}\right) \sum_{u \text{ adjacent to } v} r_v(u) = \left(1 - \frac{k}{n-i}\right), \end{aligned}$$

since nothing has been revealed about the initial random set of k nodes (it is still random after the deletion of i clique nodes). Therefore, we see that the probability of failure from any node is independent of the probability distribution chosen for the node by the algorithm. Therefore, the uniform distribution is an optimal one.

Now no algorithm can have at the ith step a probability of failure less than the probability of failure of the adaptive random walk. This is because for such an algorithm the set of the k selected nodes is still random and the clique is essentially the clique K_{n-i}. Therefore, the probability of not hitting this set, using *any* distribution on the outgoing edges of the currently visited node, is $\frac{k}{n-i}$. ∎

According to the lemma above, the number of steps taken by the adaptive random walk is a lower bound for the number of steps required by any algorithm to locate one of the randomly chosen nodes. The expected number of steps required by the uniform adaptive random walk is the following:

$$\sum_{i=1}^{n-k+1} i \prod_{l=1}^{i-1} \left(1 - \frac{k}{n-l+1}\right) \frac{k}{n-i+1}.$$

As n tends to infinity, the above expression is bounded from below by the expectation of the geometric distribution with probability of success $\frac{k}{n}$. The following lemma will help us establish this fact:

Lemma 2. *The following inequality holds:*

$$\frac{k}{(n-k)^k} \int_k^n \frac{(n+1-z)(z-k)^k}{z} dz + \frac{n-k+1}{(k+1)^{n-k}}$$

$$\leq \sum_{i=1}^{n-k+1} i \prod_{l=1}^{i-1} \left(1 - \frac{k}{n-l+1}\right) \frac{k}{n-i+1}.$$

Proof We write

$$\sum_{i=1}^{n-k+1} i \prod_{l=1}^{i-1} \left(1 - \frac{k}{n-l+1}\right) \frac{k}{n-i+1} =$$
$$\sum_{i=1}^{n-k} i \prod_{l=1}^{i-1} \left(1 - \frac{k}{n-l+1}\right) \frac{k}{n-i+1} + \frac{n-k+1}{(k+1)^{n-k}}.$$

Then for the product it holds that

$$\begin{aligned}
\prod_{l=1}^{i-1} \left(1 - \frac{k}{n-l+1}\right) &= \exp\left(\sum_{l=1}^{i-1} \ln(1 - \frac{k}{n-l+1})\right) \\
&\geq \exp\left(\int_1^i \ln(1 - \frac{k}{n-l+1}\right) dl \\
&= \left(1 - \frac{k}{n}\right)^n \frac{1}{(n-k)^k} \frac{(n-i+1)^{n-i+1}}{(n-i+1-k)^{n-i+1-k}}.
\end{aligned}$$

Therefore, using again approximations of sums with integrals, we obtain the following:

$$\begin{aligned}
&\sum_{i=1}^{n-k+1} i \prod_{l=1}^{i-1} \left(1 - \frac{k}{n-l+1}\right) \frac{k}{n-i+1} \\
&\geq \left(1 - \frac{k}{n}\right)^n \frac{k}{(n-k)^k} \sum_{i=1}^{n-k} i \frac{(n-i+1)^{n-i}}{(n-i+1-k)^{n-i+1-k}} + \frac{n-k+1}{(k+1)^{n-k}} \\
&= \left(1 - \frac{k}{n}\right)^n \frac{k}{(n-k)^k} \sum_{z=k+1}^{n} \frac{n+1-z}{(z-k)^{z-k}} z^{z-1} + \frac{n-k+1}{(k+1)^{n-k}} \\
&= \left(1 - \frac{k}{n}\right)^n \frac{k}{(n-k)^k} \sum_{z=k+1}^{n} \frac{(n+1-z)(z-k)^k}{(1-\frac{k}{n})^n z} + \frac{n-k+1}{(k+1)^{n-k}} \\
&\geq \frac{k}{(n-k)^k} \int_k^n \frac{(n+1-z)(z-k)^k}{z} dz + \frac{n-k+1}{(k+1)^{n-k}} \qquad (1)
\end{aligned}$$

as claimed. ■

Setting the denominator of the integrand in (1) equal to n, we conclude the following:

Theorem 1. *The expected number of steps taken by the uniform adaptive random walk before hitting a random set of k nodes in a complete network of n nodes is bounded from below by the following expression:*

$$E(n,k) = \frac{k}{k+1} \frac{n+2}{k+2} \left(1 - \frac{k}{n}\right) + \frac{n-k+1}{(k+1)^{n-k}}.$$

In order to use Theorem 1 in the context of searching with uncertainty in the complete network, the following lower bound to $E(n,k)$ will be convenient:

$$L(n,k) = \frac{k}{k+1} \frac{n+2}{k+2} - \left(1 + \frac{2}{n}\right).$$

Then we can prove the following:

Theorem 2. *Suppose that each node in the complete network on n nodes decides to be non-faulty independently of the others with probability $p = \omega(1/n)$. Then the expected number of steps required to locate the information node is bounded from below by an expression that is $\Theta(1/p)$.*

PROOF The expected number of steps before locating the information node can be bounded from below as follows:

$$\begin{aligned} E'(n,p) &= \sum_{k=1}^{n} \binom{n}{k} E(n,k) p^k (1-p)^{n-k} \geq \sum_{k=1}^{n} \binom{n}{k} L(n,k) p^k (1-p)^{n-k} \\ &= \sum_{k=1}^{n} \binom{n}{k} \left(1 - \frac{1}{k+1}\right) \frac{n+2}{k+2} p^k (1-p)^{n-k} - \left(1 + \frac{2}{n}\right)(1-(1-p)^n) \\ &\geq \sum_{k=1}^{n} \binom{n}{k} \frac{n+2}{k+2} p^k (1-p)^{n-k} - \sum_{k=1}^{n} \binom{n}{k} \frac{1}{k+1} \frac{n+2}{k+2} p^k (1-p)^{n-k} - 3. \end{aligned} \tag{2}$$

It can be shown that the first sum is equal to the following expression:

$$\frac{n}{n+1}\frac{1}{p} - \frac{1}{(n+1)p^2} + \frac{2}{(n+1)p} + \left[\frac{1}{(n+1)p^2} - \frac{2}{p} + \frac{1}{n+1} - n - 2\right](1-p)^n.$$

Using the fact that $p = \omega(1/n)$, this expression is equal to

$$\frac{n}{n+1}\frac{1}{p} - o\left(\frac{1}{p}\right) + o(1). \tag{3}$$

Also, the second sum can be bounded from above by the following expression:

$$1 - \frac{1}{p}\frac{n}{n+1}(1-p)^n = 1 - o(1). \tag{4}$$

Therefore, from (2), (3) and (4) we conclude that

$$\begin{aligned} E'(n,p) &\geq \frac{n}{n+1}\frac{1}{p} - o\left(\frac{1}{p}\right) - 4 + o(1) = \frac{1}{p} - \frac{1}{(n+1)p} - o\left(\frac{1}{p}\right) - 4 + o(1) \\ &= \frac{1}{p} - o\left(\frac{1}{p}\right) - 4 + o(1) \end{aligned}$$

which proves the required. ■

Therefore, the expected number of steps of any algorithm that locates the information node is bounded from below by an expression whose dominating term is $\frac{1}{p}$. Thus no algorithm may achieve an expected number of steps given by an expression with dominating term $\alpha\frac{1}{p}$, for $\alpha < 1$.

4 Deterministic search algorithms

We will now give an argument for showing that no bounded memory deterministic algorithm can achieve a finite expected number of steps. In other words, randomization is *necessary* if memory is bounded.

Theorem 3. *No fixed memory deterministic algorithm can locate the information node in the complete network of n nodes in a finite expected number of steps.*

PROOF Let A be a deterministic algorithm with a fixed amount of memory, that locates the information node in cliques of any size. A deterministic algorithm that performs one of the operations we described, can be thought of as a function $f : (S, V) \mapsto (S, V \cup \{0\})$, where S is the finite set of different states A may assume, $V = \{v_1, \ldots, v_n\}$ is the set of nodes and the number 0 means that the algorithm follows the advice of the currently visited clique node. By *states of the algorithm*, we mean the different possible contents of its memory combined with its internal states. The set of its internal states is necessarily finite.

It is easy to see, that there can be no deterministic algorithm that locates the information node in cliques of any size, that does not follow the advice of at least one of the nodes it encounters. For let K_n be one of the cliques that the algorithm handles correctly. Then the set of pairs (s, v) with s a state of the algorithm and $v \in V_n$ a vertex of the clique with n nodes is finite since the algorithm has finite memory. If n' is an integer larger than $|S \times V_n|$, then consider the operation of the algorithm on $K_{n'}$ (in what follows, we consider K_n as a subgraph of $K_{n'}$). Since the information node is chosen randomly, there is a nonzero probability that it will be one of the nodes in $V_{n'} - V_n$, say in v'. Since the algorithm is deterministic, there is no pair (s, v), with $v \in K_n$, that is mapped onto (s', v') for some state s'. Then the algorithm fails to locate the information node, which contradicts our assumption that it works correctly for cliques of any size. Therefore, there must be some state/node pair that is mapped onto $(s', 0)$ by the algorithm, for some state s' and, moreover, the algorithm must encounter at least one of these pairs, say (s, v), at some point of its operation on K_n (otherwise it would, again, not operate properly with $K_{n'}$). Now there is a nonzero probability that the following events occur simultaneously:

- The node v is faulty.
- The (wrong) advice it gives sends the algorithm to a previously encountered node (but, possibly, the algorithm is now in another state).

The algorithm must again follow the advice of some node u in K_n at some point for otherwise there exists again a sufficiently large clique that cannot be handled correctly. Again, the two events above occur simultaneously for u with nonzero probability. Since the advice of the faulty nodes never changes, the event that the algorithm will be constantly directed to previously seen state/node pairs (for K_n) is fixed and nonzero. Since the state/node pairs are finite, the algorithm will eventually reach a previously encountered pair (s, v), with some fixed nonzero probability. If this occurs, then the algorithm will repeat infinitely the same steps without ever locating the information node in K_n. Therefore, the expected number of steps required by the algorithm cannot be finite. ■

The essence of the above argument is that a finite memory deterministic algorithm should always follow the advice of the nodes it encounters. However, in doing this, it will eventually enter a cycle in its state space with fixed and nonzero probability. It can also be proved that if we alter our model so that when a faulty node is queried again it determines its wrong advice at random and does not necessarily give the same wrong advice, then the "always follow the advice" policy results in an algorithm with expected number of steps equal to $\frac{1}{p}$.

5 Randomized algorithms

5.1 A memoryless randomized algorithm

The algorithm we will describe is memoryless and at in order to decide at each step whether it will follow the advice of the current node or it will choose the next node to visit at random, it flips a fair coin.

Algorithm: Memoryless Search
Input : A clique (V, E) with a node designated as the information holder
Aim: Find the node containing the information

```
1.  begin
2.     current = RANDOM(V)
3.     l ← 1
4.     while current(information) ≠ true
5.        l ← l + 1
6.        c = FLIP(1/2)
7.        if c = HEADS
8.           current = RANDOM(V − current)
9.        else
10.          current ← advice(current)
11.    end while
12. end
```

Theorem 4. *If $p = \omega(1/n)$, the algorithm* Memoryless Search *locates the information node in expected number of steps at most equal to $4/p + O(1)$.*

Proof The algorithm succeeds if at the last decision it makes (random selection in line 8 or follow advice in line 10) it lands on the information node. In attempting to analyze probabilistically the complexity of the algorithm we are faced with the difficulty that some of its steps *expose* information about the initially random set of faulty and non-faulty nodes. For example, consider the last two flips of a sequence of coin flips and assume that they are both TAILS. Assuming, further, that the first flip leads the algorithm to a faulty node so that the algorithm does not terminate at the second flip, what is the probability of success at the second flip? It is equal to the probability that the node reached after the first flip is non-faulty. One would say that this probability is p but this is not correct. If the node was encountered by the algorithm at some previous step, then the node is certainly faulty, otherwise the algorithm would have terminated at that step. Thus, now that the node is encountered a second time, the probability that it is non-faulty is equal to 0 implying that at the second T flip of our scenario the probability of success is also 0. The situation changes if the node reached after the first flip was never met before. Then the probability of success after the second flip results in TAILS would indeed be equal to p. For these reasons, in the analysis that follows we will concentrate on the realization of the success event from sequences of coin flips ending with HT (HEADS, TAILS) as it is only for such sequences that we may claim that the probability of success at the T step is equal to p since the previous step essentially "randomizes" the T step that follows. Of course this means that we will only obtain an upper bound to the expected number of steps of the algorithm. In addition, we will ignore sequences ending in HH and TH as they have negligible effect on the expectation (as n tends to infinity, success in such sequences is becoming increasingly rare).

Given a sequence of coin flips ending with HT we will divide it into *epochs* as follows: The first epoch, e_1, starts from step $l = 1$ and ends at the step where TAILS appeared right after after HEADS (i.e. the maximal string of coin flips starting from the first flip and ending with HT). Let X_1 be the number of coin flips of epoch e_1. The ith epoch, $i > 1$, is defined similarly starting at the next flip after the end of epoch $i - 1$. Also, let X_i be the number of coin flips in epoch e_i. Each time an epoch ends, we consider that we perform an experiment with probability of success k/n which in the context of our algorithm is the probability of

having reached a non-faulty node at the HEADS step before the TAILS (the two flips that mark the end of epoch). Let S be the number of epochs formed at the end of the algorithm. Then the number of coin flips X required for success in the experiment, given that the number of epochs is $S = s$, is the following:

$$E[X/s] = E[\sum_{i=1}^{s} X_i] = \sum_{i=1}^{s} E[X_i] = sE[X_1], \tag{5}$$

as the random variables X_i are independent and identically distributed. It is not hard to see that the expectation of X_i is the following:

$$E[X_1] = \sum_{l=2}^{\infty} l\frac{l-1}{2^l} = 4. \tag{6}$$

From (5) and (6) it follows that $E[X/s] = 4s$. Therefore, the following holds:

$$E[X] = \sum_{s=1}^{\infty} E[X/s]\Pr[S = s] = \sum_{s=1}^{\infty} 4s\left(1 - \frac{k}{n}\right)^{s-1}\frac{k}{n} = \frac{4n}{k}. \tag{7}$$

In order to find an upper bound to the expected value of the number of steps in our model where each network node decides to be non-faulty with probability p, we work as follows:

$$\begin{aligned}
E[\#\text{ of steps}] &\leq \sum_{k=1}^{n} E[X/k]\binom{n}{k}p^k(1-p)^{n-k} = 4n\sum_{k=1}^{n}\frac{1}{k}\binom{n}{k}p^k(1-p)^{n-k} \\
&\leq 4n\sum_{k=1}^{n}\left(\frac{1}{k+1}\right)\binom{n}{k}p^k(1-p)^{n-k} = 4n\sum_{k=1}^{n}\frac{1}{k+1}\binom{n}{k}p^k(1-p)^{n-k} + 1 - (1-p)^n \\
&= 4\frac{n}{n+1}\sum_{k=1}^{n}\binom{n+1}{k+1}p^k(1-p)^{n-k} + 1 - (1-p)^n \\
&= 4\frac{n}{n+1}\frac{1}{p} + 1 + o(1) \leq \frac{4}{p} + O(1).
\end{aligned} \tag{8}$$

which gives the required upper bound to the expected number of steps of the algorithm. ■

5.2 One bit of memory helps

In this section, we will show that one bit of memory suffices in order to reduce the expected number of steps to locate the information node from $\frac{4}{p}$ (memoryless algorithm) to $\frac{2}{p}$.

The algorithm given below, simply alternates between following the advice of the currently visited node and selecting a random node as the next node to visit. It only needs to remember if at the last step it followed the advice or not. Although one bit suffices, the algorithm is stated as if it knew the number of steps it has taken. Then it simply checks if this number is odd or even. However, one bit would be sufficient.

Algorithm: Fixed Memory Search
Input : A clique (V, E) with a node designated as the information holder
Aim: Find the node containing the information
1. **begin**
2. current = RANDOM(V)

```
3.    l ← 1
4.    while current(information) ≠ true
5.       l ← l + 1
6.       if l mod 2 = 0
7.          current = current(advice)
8.       else
9.          current ← RANDOM(V − current)
10.   end while
11. end
```

Theorem 5. *If $p = \omega(1/n)$, the algorithm* Fixed Memory Search *locates the information node in expected number of steps at most equal to $2/p + o(1)$.*

Proof At step l, failure can occur in one of the following ways:

- If $l = 1$, the algorithm fails if it randomly selects one node other than the node containing the information. The probability for this happening is $q_1 = (1 - \frac{1}{n})$.
- If l is even, the algorithm fails if the currently visited node is faulty. The probability of this event is $q_l = q = 1 - p$.
- If l is odd, the algorithm fails if it randomly selects a node other than the currently visited one, that does not contain the information. The probability of this event is $q_l = (1 - \frac{1}{n-1})$.

Then the expected number of steps (expectation of the random variable l) is the following (where $q_l = 1 - p_l$):

$$\begin{aligned}
E[\#\text{ of steps}] &= \sum_{l=1}^{\infty} l \cdot \Pr[\text{first success in } l\text{-th step}] = \sum_{l=1}^{\infty} l \cdot q_1 \cdots q_{l-1} p_l \\
&= \left(1 - \frac{1}{n}\right) \sum_{l \geq 2,\ \text{even}} l q^{\frac{l}{2}-1} \left(1 - \frac{1}{n-1}\right)^{\frac{l}{2}-1} (1-q) \\
&+ (1 - \frac{1}{n}) \sum_{l \geq 3,\ \text{odd}} l q^{\frac{l+1}{2}-1} \left(1 - \frac{1}{n-1}\right)^{\frac{l-1}{2}-1} \frac{1}{n-1} + \frac{1}{n} \\
&= \left(1 - \frac{1}{n}\right) \sum_{l \geq 0} 2(l+1) q^{l} \left(1 - \frac{1}{n-1}\right)^{l} (1-q) \\
&+ \left(1 - \frac{1}{n}\right) \sum_{l \geq 0} [2(l+1)+1] q^{l+1} \left(1 - \frac{1}{n-1}\right)^{l} \frac{1}{n-1} + \frac{1}{n}.
\end{aligned}$$

After some algebraic manipulations, we obtain the following:

$$E[\#\text{ of steps}] = \left(1 - \frac{1}{n}\right) \frac{2n - p - 1}{1 + pn - 2p} + \frac{1}{n}.$$

Since $p = \omega(1/n)$, from the above expression it follows that $E[\#\text{ of steps}] \leq \frac{2}{p} + o(1)$. ∎

6 Unlimited memory: randomized and deterministic algorithms

We will now give an unlimited memory randomized algorithm for locating the information node in a clique. More specifically, the algorithm can store the previously visited nodes using

$O(n \log n)$ memory bits and use this knowledge in order to decide its next move. In this way, the algorithm avoids visiting again previously visited nodes. Such an algorithm always terminates within $n-1$ steps in the worst case.

Algorithm: Unlimited Memory Randomized Search
Input : A clique (V, E) with a node designated as the information holder
Aim: Find the node containing the information

```
1.  begin
2.     l ← 1
3.     current = RANDOM(V)
4.     M ← {current} // M holds the up to now visited nodes
5.     while current(information) ≠ true
6.        read(current(advice))
7.        if current ∉ M
8.           M ← M ∪ {current}
9.           current = advice
10.       end if
11.       else
12.          current ← RANDOM(V − M)
13.       l ← l + 1
14.    end while
15. end
```

We can now prove the following:

Theorem 6. *If $p = \omega(\frac{1}{n})$, then the expected number of steps required by the algorithm* Unlimited Memory Randomized Search *in order to locate the information node is, asymptotically, equal to $1/p$.*

Proof The algorithm fails at step $l \geq 1$, due to one of the following events:

1. Event A_1: the answer v to the query is one of the previously visited values.
2. Event A_2: the answer v is a node that lies outside the previously visited set but the piece of information is not there.

Let E_l be the event that at steps $1 \ldots l$ the algorithm failed to find the information. Then the probability of failure at the lth step, given that the algorithm has failed at steps $1, \ldots, l-1$, is equal to $q_1 q_2 \cdots q_l$, where

$$\begin{aligned} q_l &= \Pr[\text{failure in } l\text{-th step} | E_{l-1}] = \Pr[A_1 | E_{l-1}] + \Pr[A_2 | E_{l-1}] \\ &= q \frac{l}{n-2} \cdot \frac{n-l-2}{n-l-1} + q \frac{n-l-2}{n-2} \\ &= q \frac{n-l-2}{n-2} \left(1 + \frac{l}{n-l-1}\right) = \frac{n-l-2}{n-l-1} \cdot \frac{n-1}{n-2}. \end{aligned}$$

It follows that the expected number of steps of the above algorithm is given by the formula

$$E[\#\text{ of steps}] = \sum_{l=1}^{n-2} l \cdot \Pr[\text{first success in } l\text{-th step}] = \sum_{l=1}^{n-2} l \cdot q_0 q_1 \cdots q_{l-1} p_l$$

$$
\begin{aligned}
&= \sum_{l=1}^{n-2} l \cdot q_0 q_1 \cdots q_{l-1} - \sum_{l=1}^{n-2} l \cdot q_0 q_1 \cdots q_{l-1} p_l \\
&= \sum_{l=1}^{n-2} q_0 q_1 \cdots q_{l-1} - (n-2) q_0 q_1 \cdots q_{n-2} \\
&= \sum_{l=1}^{n-2} q_0 q_1 \cdots q_{l-1} = \sum_{l=1}^{n-2} q^l \left(\frac{n-1}{n-2}\right)^l \frac{n-l-1}{n-1} \\
&= \frac{1}{n-1} \sum_{l=1}^{n-2} q^l \left(\frac{n-1}{n-2}\right)^l (n-l-1) \\
&= \frac{n}{n-1} \sum_{l=1}^{n-2} \left(q \cdot \frac{n-1}{n-2}\right)^l - \frac{1}{n-1} \sum_{l=1}^{n-2} (l+1) \left(q \cdot \frac{n-1}{n-2}\right)^l \\
&= \frac{n}{n-1} \frac{1-a^{n-1}}{1-a} - \frac{1}{n-1} \frac{-a^{n+1} + na^n - na^{n-1} + a}{(1-a)^2},
\end{aligned}
$$

where $a = q\frac{n-1}{n-2}$. To obtain the last formula we used the well-known expansion of the geometric progression. It follows that asymptotically in n,

$$
\begin{aligned}
\frac{1-a^{n-1}}{1-a} &\sim \frac{1-q^{n-1}e}{p} \\
\frac{-a^{n+1} + na^n - na^{n-1} + a}{(1-a)^2} &\sim \frac{1}{n-1} \left(\frac{-q^{n+1}e + nq^n e - nq^{n-1}e + q}{p^2}\right).
\end{aligned}
$$

Substituting this in the previous formula we obtain that

$$
E[\#\text{ of steps}] = \frac{1}{p} - \frac{q^{n-1}e}{p} - \frac{1}{n-1} \left(\frac{-q^{n+1}e + nq^n e - nq^{n-1}e + q}{p^2}\right).
$$

As $\frac{1}{p} = o(n)$ (from the hypothesis that $p = \omega(1/n)$), we obtain that, asymptotically, $E[\#\text{ of steps}] = 1/p$, which completes the proof of the theorem. ■

As we have proved, no fixed memory deterministic algorithm has a finite expectation in the number of steps to locate the information node. However, the unlimited memory randomized algorithm we described in this section can be converted easily into an unlimited memory deterministic one by only changing line 12 to be as follows:

$$
\text{current} = \text{the lexicographically smallest node in } V - M.
$$

This change does not affect the analysis of the randomized algorithm because the probability that the chosen node is faulty or not, or that it is the information node, is the same as if the next node had been chosen at random.

7 Conclusions

In this paper we have considered the problem of searching for an information node in the complete network using information about its location obtained from the nodes of the network, where the information is correct with some bounded probability p. This problem was introduced in [12] where the ring and toroidal interconnection networks were considered. One possible

direction for future research is to consider searching with uncertainty on classes of graphs more general than the cliques. One complication that immediately arises is that a node giving correct advice does not point to the information node directly, since it may not be adjacent to it, but it points to a node that lies on a shortest path to that node (see [12]). On another front, it would be interesting to derive lower bounds on the expected number of steps for more general graphs. However, perhaps it is now not possible to apply the random walk argument that we applied to the cliques. Recall that the proof of the $\frac{1}{p}$ lower bound was based on the analysis of the expected number of steps required by a random walk on the complete network before *hitting* a randomly chosen set of nodes that give *correct* advice. Following a similar line of thought for other classes of graphs seems to be difficult in view of the discussion in [15] about the analysis of the expected hitting times in random walks on graphs. In [15] it is stated that hitting (or access) times can remain bounded, i.e. independent of the number of nodes of the graph, even for regular graphs. In the same paper, it is also stated that the only graphs for which a nonconstant lower bound on the expected hitting time can be proved are the graphs with *transitive* automorphism group. Therefore, deriving lower bounds for the problem of searching with uncertainty on more general classes of graphs may require more complex arguments. Finally, it would be interesting to know whether the $4/p$ upper bound for the memoryless randomized algorithm and the $2/p$ upper bound for the 1-bit randomized algorithm are in fact the best possible or they can be further improved.

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. 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.
7. X. Deng and C. Papadimitriou, Exploring an unknown graph, *in Proc: 31st Symposium on Foundations of Computer Science*, 1990, 356–361.
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. W. Evans and N. Pippenger, Lower bounds for noisy boolean decision trees, *in Proc. 28th Symposium on Theory of Computing*, (1996), 620–628.
10. U. Fiege, D. Peleg, P. Raghavan, and E. Upfal, Computing with uncertainty, *in Proc. 22nd Symposium on Theory of Computing*, (1990), 128–137.
11. L. Kirousis and C. Papadimitriou, Interval graphs and searching, *Discrete Mathematics* **55** (1985), 181–184.
12. 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.
13. A.C Kaporis, L.M. Kirousis, E. Kranakis, D. Krizanc, Y.C. Stamatiou, and E.C. Stavropoulos, Locating information with uncertainty in fully interconnected networks with applications to World Wide Web information retrieval, to appear in *Computer Journal*.
14. T. Leighton and B. Maggs, Expanders might be practical, *in: 30th Proc. Symposium on Foundations of Computer Science*, (1989), 384–389.
15. L. Lovász, Random Walks on Graphs: A Survey, *Combinatorics* **2** (1993), 1–46.
16. 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.

17. P. Panaite and A. Pelc, Exploring unknown undirected graphs, *in: Proc. 9th Symposium on Discrete Algorithms*, (1998), 316–322.
18. C.H. Papadimitriou and M. Yannakakis, Shortest paths without a map, *Theoretical Computer Science* **84(1)** (1991), 127–150.
19. N. Pippenger, On networks of noisy gates, *in: Proc. 26th Symposium on Foundations of Computer Science*, (1985), 30–36.
20. P. Raghavan, Robust algorithms for packet routing in a mesh, *in: Proc. 1st Symposium on Parallel Algorithms and Architectures*, (1989), 344–350.
21. T.E.S. Raghavan, T.S. Ferguson, T. Parthasapathy, and O.J. Vrieze eds., *Stochastic games and related topics*. Kluwer Academic Publishers, 1991.
22. W.H. Ruckle, *Geometric games and their applications*. Research Notes in Mathematics, Pitman Publishing Inc., 1983.