Locating a Target with an Agent Guided by Unreliable Local Advice
Nicolas Hanusse, David Ilcinkas, Adrian Kosowski, Nicolas Nisse

To cite this version:
Nicolas Hanusse, David Ilcinkas, Adrian Kosowski, Nicolas Nisse. Locating a Target with an Agent Guided by Unreliable Local Advice. PODC 2010, Jul 2010, Zurich, Switzerland. ACM New York, NY, USA, pp.355-364, 2010, <10.1145/1835698.1835781>. <hal-00516695>
Locating a Target with an Agent Guided by Unreliable Local Advice

How to Beat the Random Walk when you have a Clock?

Nicolas Hanusse
CNRS, LaBRI/INRIA,
Université de Bordeaux 1
Bordeaux, France
nicolas.hanusse@labri.fr

David Ilcinkas
CNRS, LaBRI/INRIA,
Université de Bordeaux 1
Bordeaux, France
david.ilcinkas@labri.fr

Adrian Kosowski
Gdańsk University of Technology
Gdańsk, Poland
adrian@kaims.pl

Nicolas Nisse
MASCOTTE, INRIA,
I3S(CNRS/UNS)
Sophia Antipolis, France
nicolas.nisse@sophia.inria.fr

ABSTRACT

We study the problem of finding a destination node $t$ by a mobile agent in an unreliable network having the structure of an unweighted graph, in a model first proposed by Hanusse et al. [21, 20]. Each node is able to give advice concerning the next node to visit so as to go closer to the target $t$. Unfortunately, exactly $k$ of the nodes, called liars, give advice which is incorrect. It is known that for an $n$-node graph $G$ of maximum degree $\Delta \geq 3$, reaching a target at a distance of $d$ from the initial location may require an expected time of $2^{\Omega(\min\{d,k\})}$, for any $d, k = O(\log n)$, even when $G$ is a tree.

This paper focuses on strategies which efficiently solve the search problem in scenarios in which, at each node, the agent may only choose between following the local advice, or randomly selecting an incident edge. The strategy which we put forward, called $R/A$, makes use of a timer (step counter) to alternate between phases of ignoring advice ($R$) and following advice ($A$) for a certain number of steps. No knowledge of parameters $n$, $d$, or $k$ is required, and the agent need not know by which edge it entered the node of its current location. The performance of this strategy is studied for two classes of regular graphs with extremal values of expansion, namely, for rings and for random $\Delta$-regular graphs (an important class of expanders). For the ring, $R/A$ is shown to achieve an expected searching time of $2d + k^{\Theta(1)}$ for a worst-case distribution of liars, which is polynomial in both $d$ and $k$. For random $\Delta$-regular graphs, the expected searching time of the $R/A$ strategy is $O(k^{3} \log^{3} n)$ a.a.s. The polylogarithmic factor with respect to $n$ cannot be dropped from this bound; in fact, we show that a lower time bound of $\Omega(\log n)$ steps holds for all $d, k = \Omega(\log \log n)$ in random $\Delta$-regular graphs a.a.s. and applies even to strategies which make use of some knowledge of the environment.

Finally, we study oblivious strategies which do not use any memory (in particular, with no timer). Such strategies are essentially a form of a random walk, possibly biased by local advice. We show that such biased random walks sometimes achieve drastically worse performance than the $R/A$ strategy. In particular, on the ring, no biased random walk can have a searching time which is polynomial in $d$ and $k$.

Categories and Subject Descriptors
G.2.2 [Graph Theory]: Graph algorithms

General Terms
Algorithms, Theory

Keywords
Distributed Computing, Mobile Agent, Random Walks, Expanders, Faulty Networks

1. INTRODUCTION:

THE SEARCH PROBLEM

Walking in the streets of Paris, you decide to visit the famous Musée du Louvre, but do not know where it is situated. You first ask some people who say you should go to the North. After a short walk, you ask a policeman who is almost sure it is to the East. At the next intersection, you are told to go to the South. At least one of the persons you have met is mistaken. What is the best strategy to quickly find the museum?

Locating an item (a piece of information, data, services, etc.) is one of the most common tasks in a distributed environment. This is, for instance, the role of search engines in
the World Wide Web. One idea for a user of a network to locate an item at some node is to send agents out to search for the desired item [30, 27, 25]. If the mobile agent is provided complete information about the network and the location of the item, it can quickly find it by following a shortest path from its current position to the node hosting the item. Another way for the agent to find an item without being provided any information consists in exploring the whole network until the desired item is encountered. Several works have been devoted to the problem of exhaustive network exploration and numerous algorithms, both deterministic (e.g., Breadth First Search, Universal Traversal Sequences [1, 36], Universal Exploration Sequences [29, 35]) and randomized (e.g., random walks [2, 31, 26], biased random walks [4, 6]), have been designed. In the context of large scale networks like the World Wide Web or the Internet, it is impractical to have full knowledge of the network because of its size and dynamicity. It is also impossible to fully explore the network since such an approach requires a time complexity (at least) in the order of the number of nodes of the network.

Another important constraints to the complete exploration of a network are the memory of the agent (e.g., see [15]) and the notion of sense-of-direction [12, 13, 14].

In this paper, we consider the problem of locating an item hosted by some node of the network when each of the nodes maintains a database storing the first edge on a shortest path to the node hosting the desired item (the destination). The search is performed by a mobile agent with a limited perception of the environment and with little memory which starts from some initial node, the source. When occupying a node, the mobile agent can perform a query to the node's database that reveals to it an edge that is the beginning of a shortest path from the current node to the item. We assume, however, that some nodes may provide wrong information, that is, a node $v$ may indicate an edge that does not belong to any shortest path from $v$ to the destination. This is motivated by the fact that inaccuracies occur in the nodes' databases because nodes may malfunction or be malicious, or may store out-of-date information due to the movement of items or the dynamicity of the network. This is also the case when you are searching some place in a city by asking your way to someone you meet. A node providing wrong information is called a liar, otherwise it is a truth-teller.

The problem is then to deal with the potentially incorrect information and to find the desired item. That is, the mobile agent can decide to follow the edge pointed by its current node's database or not. The performance of the search is measured by comparing the searching time (a.k.a. hitting time), i.e., the length of the walk followed by the agent from the source to the destination, with the length $d$ of a shortest path between these nodes.

In this paper, we investigate the search problem in the class of regular graphs in the presence of a bounded number $k$ of liars. Our main contribution is the design and study of a randomized algorithm, called $R/A$ that alternates phases of pure random walk ($R$) with phases in which the agent follows the advice ($A$). We show that Algorithm $R/A$ improves upon previous algorithms for the search problem in paths and random $\Delta$-regular graphs. In particular, in these classes, we prove that the Algorithm $R/A$ achieves searching time much smaller than $\Omega(d + 2^k)$, which is the lower bound for general regular graphs [20]. Note that the graph classes we consider capture the two extreme types of behaviour in terms of expansion, since random $\Delta$-regular graphs are good expanders, while the other classes are highly symmetric graphs with poor expansion. Note, however, that Algorithm $R/A$ is generic and works for any topology.

### 1.1 Related Work

The search problem in the presence of liars was first investigated by Kranakis and Krizanc [30]. In this seminal work, they designed algorithms for searching in distributed networks with ring or torus topology, when a node has a constant probability of being a liar [30]. The case when the number $k$ of liars is bounded was first considered in [21], where deterministic algorithms were designed for particular topologies like the complete graph, ring, torus, hypercube, and bounded degree trees. In particular, in bounded degree trees, it is proved that the search time is lower-bounded by $\Omega(d + \gamma_{\text{min}}(r,d))$ [21]. Simple randomized and memoryless algorithms are designed in [20] for the case of bounded degree graphs, where the mobile agent follows the advice with some fixed probability $p > 1/2$. In this class of graphs, the authors showed that the expected distance covered before reaching the destination is upper-bounded by $O(d + r^k)$, where $r = \frac{1}{1-p}$ [20]. Moreover, this bound is tight since they proved a lower bound of $\Omega(d + r^k)$ in the torus [20]. While this bound is a bit disappointing, it can be improved for particular graph classes. In this paper, we focus on some particular and widely used topologies.

#### Biased random walks.

Roughly speaking, the algorithms we present in this paper consist of alternation of phases of given duration: either the agent keeps on following the advice provided by the nodes or it walks choosing the next visited node uniformly at random in the neighborhood of the current position. This is closely related to biased random walks which are random walks in which nodes have a statistical preference to shift the walker towards the target, or more generally, prevent the walker from staying too long in one vicinity [4]. More formally, biased random walks are used in network exploration in order to speed up the time required to visit the whole network without an a-priori knowledge of the topology and without an edge/node labeling requirement. For instance, Ieda et al. proved in [23] that, assuming the knowledge of the degrees of the neighbors, a biased random walk can explore (cover) any graph within $O(n^2 \log n)$ edge traversals whereas a uniform random walk takes $\Theta(n^3)$ steps for some graphs. In our context, however, the bias may be erroneous due to the presence of liars.

#### Expanders and random regular graphs.

Expander graphs are highly connected sparse graphs that play an important role in computer science and the theory of communication networks (see [22] for a survey). Formally, a graph $G = (V, E)$ is a $c$-expander if, for any $X \subseteq V$ with $|X| \leq |V|/2$, then $|N(X) \setminus X| \geq c|X|$ where $N(X)$ is the set of neighbours of $X$. Expanders arise in questions about designing networks that connect many users while using only a small number of switches (e.g., see [3]). They also arise in constructions of error-correcting codes with efficient encoding and decoding algorithms, derandomisation of random algorithms, embeddings of finite metric spaces, etc.

Expanders and random regular graphs have been exten-
sively studied for the design of optimal networks and algorithms for routing [33, 28, 17]. Because of their low diameter and high connectivity, random regular graphs are also of interest in Peer-to-Peer networks (e.g., see [18]). More generally, it can be observed that many interaction networks like peer to peer overlay networks, small worlds and scale-free networks are expanders despite this is not proved in the original papers. For instance, Bourassa and Holt [8] proposed a fully decentralized protocol based on random walks for the nodes to join and leave the network. They conjectured that their protocol produces random regular graphs, which was proved formally in [9]. On the other hand, Cooper [10] et al. show that random regular graphs are expanders.

1.2 Terminology and the model

Throughout the paper, a distributed network is for our purposes an undirected \( n \)-node graph \( G = (V, E) \). There are two distinguished nodes in the graph, the \( source \) and the \( destination \) \( t \in V \), hosting the desired item. The distance \( d(u,v) \) between two nodes \( u \) and \( v \) corresponds to the number of edges of a shortest path from \( u \) to \( v \). We set \( d(u,t) = d \). The number of liars is denoted by \( k \geq 0 \). For any \( v \in V \) and \( r \geq 0 \), let \( N_r(v) \) denote the distance- \( r \)-neighborhood of \( v \), i.e., the set of all nodes \( u \in V \) such that \( dist(u,v) \leq r \). We will call the subgraph \( B_r(v) = G[N_r(v)] \) the ball with center \( v \) and radius \( r \). The \( deg(u) \) edges incident to any node \( u \) are labeled by \( port \) numbers, from 1 to \( deg(u) \), so that the searchers can distinguish the different edges incident to a node. There is no sense-of-direction [12, 13, 14], meaning that the local labeling of the edges satisfies no global consistency constraints (e.g., right/left in a path, or North/South/East/West in a grid). In most cases, we assume that the agent does not know the label of the port by which it entered the current node. Note that in a \( \Delta \)-regular graph, all nodes are indistinguishable for the agent.

At each step of the execution of an algorithm, the agent performs a query to its current node \( v \in V \). If \( v = t \), the item is found and the mobile agent stops. Otherwise, a piece of advice \( a \in \{1, \ldots, deg(v)\} \) is given to the agent, representing the port number \( a \) of the next edge \( e \) incident to \( v \) which should be crossed in order to reach \( t \). If \( v \) is a \textit{truth-teller}, \( e \) belongs to a shortest path between \( v \) and \( t \). Otherwise, \( v \) is called a \textit{liar}. Finally, the mobile agent chooses some edge incident to \( v \) and traverses it. In the following, the set of liars is denoted by \( \mathcal{L} \) and the set of truth-tellers by \( \mathcal{T} \). \textit{A priori}, a search algorithm can take into account the set of advice encountered so far to choose the next edge to cross. In particular, this means that a node should always provide the same advice, otherwise it would easily be identified as a liar. However, for practical applications, it is natural to limit the agent’s memory. Here, we consider that the agent only has a timer (whose size is specified below). Hence, a lying node may or may not provide always the same advice (but a node cannot change its status: it is either consistently a liar or a truth-teller). Finally, the agent will also be assumed to have no global knowledge about the size of the network, the number of liars, and the value of \( d \).

We are mainly interested in the expected number of edge traversals, named the \textit{searching time}, taken by the mobile agent to reach the target \( t \), and in comparing it with \( d \).

1.3 Results and structure of the paper

We design and study Algorithm R/A [\( \tau_u, \tau_A \) defined as follows. The mobile agent alternates between phases in which it performs a random walk for \( \tau_g \geq 0 \) steps and then follows advice for \( \tau_A \geq 0 \) steps (Algorithm 1). Note that this algorithm can fundamentally work for any topology. However, a cautious design of duration is necessary. For instance, if the duration of the phase \( A \) is much larger than that of phase \( R \), the mobile agent could be stuck forever in the same area full of liars. By carefully parameterizing the durations, we prove that:

- in the path (similarly in the ring), an agent using a counter of \( O(\log k) \) bits and following Algorithm \textit{R/A} ends up at the target \( t \) within \( 2d + O(k^3) + o(d) \) moves with probability \( 1 - \Theta(1/k^c-3) \), where \( c \) is a constant (Section 2), and
- in random \( \Delta \)-regular graphs, an agent using a counter of \( O(\log k + \log \log n) \) bits and following Algorithm \textit{R/A} ends up at the target \( t \) within \( O(c \cdot k^3 \log n) \) steps with probability \( 1 - 1/2^{kR(v)} \), where \( c \) is a constant (Section 3).

In both results, no knowledge of \( n, d, \) or \( k \) is required. Finally, in the case of random \( \Delta \)-regular graphs, we provide a lower bound of \( \Omega(\log n) \) which holds for all \( d, k = \Omega(\log \log n) \) and applies even to strategies which are in some sense not oblivious with respect to the environment.

Tables 1 and 2 establish a comparison between the performances of Algorithm \textit{R/A}, and the performances of the \textit{Biased Random Walk} (BRW). In the BRW strategy, the agent flips a biased coin and accepts the advice with probability \( p \) and rejects it with probability \( 1 - p \), in which case it also selects any of the remaining incident edges with uniform probability in the number of remaining incident edges. Note that when the searched graph is a line, if \( p = 1/2 \), BRW is the pure Random Walk.

2. SEARCHING THE PATH

In this section, let us assume that the initial graph is a path \( P \) of \( n \) nodes. For this topology, we look at the behavior of the generic algorithms, which we later study for expanders. Recall that \( k \) denotes the number of liars.

Note that whereas it is of course possible to design extremely simple and efficient algorithms specifically for searching a path, such algorithms will usually not perform well in general. Consider for example the 1D \textit{Cow Path} strategy proposed in [5], which does not take into account the advice. For \( r \in [1, \lfloor \log_2 d \rfloor] \), it just consists in going \( 2^r \) steps in one given direction, then in going \( 2^{r+1} \) steps in the opposite direction and repeats the sequence until the target is found. This algorithm is efficient in the case of the path, always reaching its target in at most \( 9d \) steps [5]; however, this is not
the end of Phase R, the mobile agent first executes a random walk during the case when the expansion of the graph is large (e.g., the exponential in the number of liars [20]). Regardless of the value of $p$, the searching time of BRW in the path is always lower-bounded by $\Omega(d + 2^\min(k,d))$.

In this section, we study the performance of Algorithm R/A on a path. Recall that, following this algorithm, the search consists of rounds of length $2L > 0$. During each round, the mobile agent first executes a random walk during $L$ steps (phase $R$), and then it follows the advice during the next $L$ steps (phase $A$).

Let $u_0, u_1, \ldots, u_L$ be the sequence of nodes traversed by the mobile agent in a phase $R$ and let $v_0 = u_0, v_1, \ldots, v_L$ be the sequence of nodes traversed by the mobile agent in a phase $A$. We are interested in the gain $X = X_A + X_R$ of $u_L$ with respect to $u_0$ defined by $X = d(u_0, t) - d(v_L, t)$, $X_R$ (resp. $X_A$) being the gain during phase $R$ (resp. $A$).

In the following, we will make use of an assumption: $u_L$ is almost always close to $u_0$. More formally, let $C_i$ be the event defined by $u_L$ is at distance at most $\sqrt{2L}$ from $u_0$, i.e., $u_L \in B_{u_0}(\sqrt{2L})$. We get:

**Lemma 1** (Random Phase). Let $c > 0$ and $k > 0$. At the end of Phase $R$, we have:

1. With probability at least $1 - 2/k^c$, event $C_{c \log k}$ holds;
2. With probability at least $1 - 2/e$, event $C_1$ holds.

Let $|i| \leq L$ with same parity as $L$.

3. $\Pr(d(u_0, u_i) = i) \leq p_{\max} = 0.8/\sqrt{L}$;
4. If $|i| \leq \sqrt{2L}$, $\Pr(d(u_0, u_i) = i) \geq p_{\min} = 0.1/\sqrt{L}$.

**Proof.** Let $X_1, X_2, \ldots, X_L$ be independent random variables such that $\Pr(X_i = 1) = \Pr(X_i = -1) = \frac{1}{2}$. The gain $X_R = d(u_0, t) - d(u_L, t)$ during phase $R$ is $X_R = \sum_{i=1}^{L} X_i$, that is, $u_L$ is at distance $|X_R|$ from $u_0$. From Chernoff’s bound, we have:

$$\Pr(|X_R| \geq a) \leq 2 e^{-a^2/4L}, \quad a > 0.$$ 

For the first statement (resp. the second), take $a = \sqrt{2L} \log k$ (resp. $a = \sqrt{2L}$).

Let $-L \leq i \leq L$ such that $L - i \equiv 0[2]$. The probability that $X_R = i$ is exactly $(L/(2L+|i|))/2L$. We recall that we should have $L/2 + i/2$ moves in one given direction and $L/2 - i/2$ moves in the opposite direction. We prove the lemma in the case $L = 2L$, the case $L$ odd is similar. The function $f(2) = (L)/(2L)$ is a decreasing function ($0 \leq j \leq L$). Hence, for all $0 \leq j < L$, $f(2) \leq f(0) = (L) / (2L)$. By the Stirling Formula, $f(0) \sim \sqrt{\frac{2}{\pi}} L \leq 0.8/\sqrt{L}$.

The last statement is similar. Indeed, since the function $f$ is decreasing, we get that, for any $i = 2j$ with $|i| \leq \sqrt{2L}$, $f(i) \geq f(2 \sqrt{L}/2) \geq f(2 \sqrt{L})$. Moreover, $f(2 \sqrt{L}) = (L + \sqrt{L})/2L = \sqrt{(L)(L+2\sqrt{L})/4L^2} + o(L^{-3/2}).$ Hence, $f(2 \sqrt{L}) \geq 0.1/\sqrt{L}$. □

Consider the subpath of $P \setminus \{t\}$ containing $u_0$. It can be decomposed into at most $2k+1$ maximal subpaths of consecutive nodes such that each subpath is composed only of nodes of same state, that is liars or truth-tellers. We assume that the subpaths (or sets of nodes) are ordered from

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Expected searching time (a.a.s.)</th>
<th>Memory</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRW $[p &lt; 1/2]$</td>
<td>$2^{\Omega(d)}$</td>
<td>–</td>
<td>[31]</td>
</tr>
<tr>
<td>BRW $[p = 1/2]$</td>
<td>$\Omega(\log n)$</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>BRW $[p &gt; 1/2]$</td>
<td>$\Omega(d + 2^{\Theta(k)})$</td>
<td>–</td>
<td>[20]</td>
</tr>
<tr>
<td>$R/A$</td>
<td>$2d + k^{\Theta(1)} + o(d)$</td>
<td>timer $(O(\log t))$</td>
<td>Thm. 2</td>
</tr>
</tbody>
</table>

Table 1: Searching in a path with liars. The listed strategies have no knowledge of inbound ports used by the agent. The number of rounds required to complete the search is denoted by $t$.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Expected searching time</th>
<th>Memory</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^*$ lower bound</td>
<td>$\Omega(\min((\Delta - 1)^k, (\Delta - 1)^d, \log \Delta \log n))$</td>
<td>–</td>
<td>Thm. 6</td>
</tr>
<tr>
<td>BRW $[p &gt; 1/2]$</td>
<td>$\Theta(\min((\Delta - 1)^k, n^{\Theta(1)}))$</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>BRW $[p = 1/2]$</td>
<td>$\Omega(\log^2 \Delta \log n)$</td>
<td>–</td>
<td>[10]</td>
</tr>
<tr>
<td>BRW $[p &lt; 1/2]$</td>
<td>$\Omega(\log k \log \log \Delta \log n)$</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>$^* R/A/E$</td>
<td>$O(k \log n)$</td>
<td>$\Theta(\log k + \log \log \Delta \log n)$</td>
<td>Thm. 3</td>
</tr>
<tr>
<td>$R/A$</td>
<td>$O(k^3 \log^3 n)$</td>
<td>timer $(O(\log t))$</td>
<td>Thm. 4</td>
</tr>
</tbody>
</table>

Table 2: Searching with liars in random $\Delta$-regular graphs. Results marked with (‘$*$’) allow the agent to make use of labels of inbound ports and to have knowledge of $n$ and $k$. The number of rounds required to complete the search is denoted by $t$. |
the subpath furthest from \( t \) to the subpath closest to \( t \). Define \( L_i \) (resp. \( T_i \)) as the \( i \)-th set of liars (resp. truth-tellers) of the path.

**Lemma 2 (Advice Phase).** Let \( c \geq 1 \) such that \( \sqrt{c+1} - \sqrt{c} > 1/\sqrt{2\log k} \), and \( L > 32k^3 \). Let \( k' \) be the number of liars at distance at most \( \sqrt{2(c+1)\log k} \) from \( u_0 \). Condition on \( C_{c \log k} \), we have:

1. \(-k' \leq X_A < 0 \) with probability at most \( p_{\max} \cdot k' = O(k'/\sqrt{L}) \);
2. \( X_A \geq \sqrt{2L}/k' - 1 \) with probability at least \( \sqrt{2L}p_{\min}(1-2/e) = \Theta(1) \);
3. \( \mathbb{E}(X) = \mathbb{E}(X_A) \) and \( \mathbb{E}(X_A) \geq \frac{\sqrt{L}}{k' - 1} \).

Proof. By hypothesis, \( u_c \) belongs to \( B_{u_0}(\sqrt{2c\log k}) \), the path of \( 2\sqrt{2c\log k} + 1 \) consecutive nodes centered at node \( u_0 \).

We first prove that after phase \( A \), the distance can only increase by at most \( k' \) units. If \( u_L \) is a liar belonging to \( L_i \), we know that during phase \( A \), the distance to \( t \) can increase by at most \( |L_i| \leq k \) units, i.e., \( X_A = d(u_0, t) - d(u_L, t) \leq d(u_0, t) - d(u_L, t) + |L_i| \). In this case, since \( d(u_c, u_L) \leq k \), we have that \( u_L \in B_{u_0}(\sqrt{(2c+1)\log k}) \) because, by the choice of \( L \) and \( c \), \( \sqrt{2(c+1)\log k} - \sqrt{2c\log k} > k \). Hence, \( |L_i| \leq k' \).

If \( u_L \) is a truth-teller, the distance to \( t \) decreases by a number of units which is the minimum of \( L \) and the distance to the next set of liars (towards \( t \)).

Let us focus on the gain \( X_A \) during phase \( A \). \( X_A < 0 \) if and only if \( u_L \) is a liar. Otherwise \( X_A \geq 0 \). \( X_A = 0 \) if possible as soon as \( u_c \) is a truth-teller being a neighbor of a liar and \( L \) is even.

From Lemma 1, we know that for any given node \( v \) at distance at most \( L \) from \( u_0 \) and such that \( d(u_0, v) \) has same parity as \( L \), \( \Pr(u_L = v) \leq p_{\max} \). Moreover, by hypothesis, \( u_L \) belongs to \( B_{u_0}(\sqrt{2c\log k}) \) that contains at most \( k' \) liars. Hence, with probability at most \( p_{\max}k' \), \( u_L \) is a liar and thus \( -k' \leq X_A < 0 \).

Whenever \( u_L \) is a truth-teller, \( u_L \) belongs (or not) to \( B_{u_0}(\sqrt{2L}) \). By Lemma 1, any nodes of \( B_{u_0}(\sqrt{2L}) \) has almost the same probability to be chosen (there is a multiplicative factor of at most \( p_{\max}/p_{\max} \)). \( X_A = i \) if there are \( i \) consecutive truth-tellers before the next liar. Let us prove that \( X_A = \Omega(\sqrt{L}/k') \) with constant probability. We just focus on truth-tellers belonging to \( B_{u_0}(\sqrt{2L}) \).

For \( y \geq 1 \), let \( t_y \) be the number of disjoint sequences of consecutive truth-tellers of length \( y \) of \( B_{u_0}(\sqrt{2L}) \). Note that \( \sum_{y \geq 1} t_y \leq k' \).

By definition, we have:

\[
\sum_{t_y \geq 1} y \cdot t_y \geq 2\sqrt{2L} - k' + 1
\]

Since at least \( \sum_{y \geq 2} (y-x)t_y \) vertices in \( B_{u_0}(\sqrt{2L}) \) are at distance at least \( x \) from the next liar in the direction of \( t \), we have

\[
\Pr(X_A \geq x) \geq p_{\min}(1 - 2/e) \sum_{y \geq x} (y-x)t_y
\]

Taking \( x = \sqrt{2L}/k' - 1 \), we have \( \sum_{y \geq 2} (y-x)t_y \leq \sqrt{2L} - k' \). From Eq. (1), it follows that \( \sum_{y \geq 2} (y-x)t_y \geq \sqrt{2L} + 1 \) and \( \Pr(X_A \geq \sqrt{2L}/k' - 1) \geq \sqrt{2L}p_{\min}(1 - 2/e) = \Theta(1) \).

The expectation of \( X_A \) is

\[
\mathbb{E}(X_A) \geq -k' \Pr(X_A < 0)
\]

\[
+ \left( \frac{\sqrt{2L}}{k'} - 1 \right) \Pr(X_A \geq \sqrt{2L}/k' - 1)
\]

\[
\geq -k'p_{\max}
\]

\[
+ \left( \frac{\sqrt{2L}}{k'} - 1 \right) \sqrt{2L}p_{\min}(1 - 2/e)
\]

\[
\geq - \frac{0.8k'^2}{\sqrt{L}} + 0.1\sqrt{2(1 - 2/e)}\left( \frac{\sqrt{2L}}{k'} \right)
\]

Since \( L > 32k^3 \), we get:

\[
\mathbb{E}(X_A) \geq \frac{\sqrt{L}}{k'}(-1/40 + 1/20) - 1
\]

\[
\geq \frac{\sqrt{L}}{40k^2} - 1
\]

Since \( X = X_R + X_A \) and \( \mathbb{E}(X_R) = 0 \), we get \( \mathbb{E}(X) = \mathbb{E}(X_A) \).

We now need some definitions. The *advice cone* \( C_u \) of a liar \( u \) is the set of nodes \( v \) such that there is a path \( v_0 = v, v_1, \ldots, v_i = u \) from \( v \) to \( u \) with \( v_j = \text{Adv}(v_{j-1}) \). For \( r \geq 1 \), we note the *r-restricted advice cone* of a liar \( L_u \) that contains all sets of all vertices at distance at most \( r \) from \( C_u \) (including \( C_u \)). Finally, a *r-box* is a connected component of the subgraph induced by all r-zones.

In other words, consider one of the two components of \( P \setminus \{t\} \), and let \( \{u_1, \ldots, u_k\} \) be the liars in this component, ordered by decreasing distance to \( t \). The cone of \( u_1 \) consists of \( u_1 \) plus the component of \( P \setminus \{u_1\} \) that does not contain \( t \). For any \( i > 1 \), \( C_{u_i} \) consists of the subpath of truth-tellers between \( u_{i-1} \) and \( u_i \), plus \( u_i \). It follows that, on the path, a r-box containing \( k' \geq 1 \) liars \( \{u_1', \ldots, u_{k'}'\} \) consists of a subpath \( P' = \{v_1, v_2, \ldots, v_{r-1}\} \), such that the neighbors of \( v_1 \) and \( v_{r-1} \) are truth-tellers, \( u_1' = v_{2r+1}, u_{k'}' = v_{r-1} \), and for any \( 1 < i < k' \), \( u_i' \) is between \( u_{i-1}' \) and \( u_{i+1}' \), and for any \( 1 < i < k' \), \( d(u_i, u_{i+1}) \leq 2r \). In particular, a r-box containing \( k' \) liars is a subpath of length at most \( 3r + 1 \), and at most \( (2k' + 1)r + k' \).

**Lemma 3 (Inside a Box).** Let \( L = \Omega(k^2) \), \( c \geq 1 \), and \( r = \sqrt{c\log k} \). Let us assume that event \( C_{c \log k} \) always occurs. Given a r-box \( B \) of \( k' \leq k \) liars, the agent, initially located within any node of \( B \), leaves the box after \( O(k^2\sqrt{c\log k}) \) expected rounds of Algorithm \( R[A, L, L] \) in the direction of the target. The exit time is \( O(\max\{c \log k, k^2\sqrt{c \log k}\}) \) with probability \( 1 - 2/k^c \).

Proof. First, let us make some easy remarks.

- If, at the end of a phase \( A \), the mobile agent still stands in \( B \), then it is on a liar or on the neighbor of a liar in its cone (indeed, in this case, at the end of phase \( A \), the agent oscillates between these two nodes).
- Condition on event \( C_{c \log k} \), either the agent exits from the box in the direction of the target or stay within the box. Once the agent is outside the box after one phase \( R \), during phase \( A \), it increases its distance from \( B \) of at least \( r + 1 \) (the distance minimum to the first liar of the next box) and at most \( L \) (if no liars are encountered).
we get that $\Theta(2k + 1)$ steps with probability $1 - 2k^{-c}$. Hence, $X[i] \geq \Theta((2k + 1)r + k')$ and the mobile agent leaves the box with high probability.

To conclude this section, we prove that the knowledge of $k$ is not necessary to achieve the same performance as the one of Theorem 1.

**Theorem 2.** The iterated execution of Algorithm R/A with parameter $i$ starting at $i = 0$ and increasing by one unit every $t_i = i \cdot 2^{2i}$ phases, finds the target within $2d + O(k^2 \log k) + o(d)$ steps with probability $1 - \Theta(1/k^{c-3})$, without assuming knowledge of $k$.

**Proof.** Whenever the number of iterations $(2d/L$ on average) required to cross safe areas is larger than the ones to traverse the boxes of liars (roughly $40k^3\sqrt{\log k}$ iterations), the time to reach the target is dominated by the time of traversal of the safe areas.

For $i = 5/3 + \log_2 k$, the $i$-th execution of the proposed algorithm corresponds to the execution of $R/A[L_i]$ with $L = 2^{5/3 + \log_2 k} = 32k^3$ during $O(i \cdot 2^{2i}) = O(k^2 \log k)$ iterations. According to Theorem 1, this finds the target with probability $1 - \Theta(1/k^{c-3})$.

Now, the total number of steps of the proposed algorithm until it terminates its $(5/3 + \log_2 k)^{th}$ iteration is

$$O\left(\sum_{l=1}^{5/3 + \log_2 k} L_{l,1} \cdot i \cdot 2^{2i} \right) = O\left(\sum_{i=1}^{\log_2 k} i \cdot \left(2^{5/3} \cdot 2^{2i} \right)\right) = O\left(2^{5/3 + \log_2 k} \log k\right) = O(5k \log k)$$
Either the target is reached during these steps \((d < \frac{11}{32} \cdot 4^k \cdot \sqrt{\log_2 n})\) or the agent spend more time in the safe areas and the target is reached within \(2d + o(d)\) steps. □

3. SEARCHING IN RANDOM \(\Delta\)-REGULAR GRAPHS

3.1 Preliminaries: Properties of Random \(\Delta\)-regular graphs

A particular class of graphs with powerful expansion properties is that of random \(\Delta\)-regular graphs. A random \(\Delta\)-regular graph \(G_{n,\Delta}\) is an element of \(\mathcal{G}(n,\Delta)\), the set of \(\Delta\)-regular graphs with \(n\) nodes viewed as a probability space with uniform probability \([24]\). It is well known that a random regular graph does not have the same properties as those of the standard random graphs model, namely the Erdős-Rényi random graphs \(\mathcal{G}(n,p)\) with parameter \(p\) \(^1\), even when the parameter \(p\) is chosen so that both graphs have the same expected number of edges \((p = \Delta/n)\). For \(\Delta = o(\log n)\), \(G_{n,\Delta}\) is connected asymptotically almost surely (a.a.s.), whereas this is not the case for \(\mathcal{G}(n,p)\). Random \(\Delta\)-regular graphs were proved to be very powerful expanders, a.a.s. \([34, 18]\). Moreover, there exists a quick randomized algorithm for generating such graphs \([34]\).

We recall here some properties of random \(\Delta\)-regular graphs that will be useful in the paper.

**Lemma 5 (Diameter \([7]\)).** There exists a constant \(D\) such that the diameter of \(G_{n,\Delta}\) a.a.s. satisfies the relation \(|diam\ G_{n,\Delta} - (\log_{\Delta-1} n + \log_{\Delta-1} \log n)| < D\).

**Lemma 6 (Tree-like neighbourhood \([32, 11]\)).** There exists a constant \(c > 0\), the subgraph induced by \(B_n(c\log n)\) is a.a.s. a tree.

**Lemma 7 (Mixing time \([16, 10]\)).** For any starting node \(u\) of \(G_{n,\Delta}\) and after following a random walk of \(M_{n,\Delta} = \frac{\log n}{\log(\Delta / 2)}\) steps, for every node \(v\), the probability that the walk ends at \(v\) is at least \(1/n - 1/n^3\).

This last lemma implies that we will restrict considerations to the case \(\Delta > 4\) in the following discussion.

3.2 Upper Bounds

Before discussing the behavior of the \(R/A\) algorithm for random regular graphs, we introduce a new algorithm called \(R/A/E\) (Algorithm 2). Algorithm \(R/A/E\) is formulated in a way which assumes that we have knowledge of the port by which we entered each node in order to locally explore a ball in the exploration phase. Then, we show how to apply an analogous analysis to the \(R/A\) algorithm, for an agent with no knowledge of inbound ports.

The values of parameters \(t_R, t_A\) and \(r_E\), which are used for determining the duration of phase \(R\) (resp. \(A\) and \(E\)), are all set deterministically \textit{a priori} and will be described later.

Let \(r_1 = \log_{\Delta-1} n - 1\), let \(r_2 = \log_{\Delta-1} n + \log_{\Delta-1} \log n + D\), where \(D\) is the constant from Lemma 5, and let the target \(t\) be an arbitrarily chosen node in graph \(G \in G_{n,\Delta}\).

\(^1\)In the Erdős-Rényi model, a random graph is built in the following way: for any pair of nodes, flip a biased coin and it exists an edge between these two nodes with probability \(p\).

**Algorithm 2 Algorithm \(R/A/E\) with parameters \(t_R, t_A, r_E\).**

**Algorithm \(R/A/E\) [\(t_R, t_A, r_E\):** Repeat the following sequence of three phases until the target is reached:

1. **Random phase (R):** the mobile agent performs a pure unbiased random walk for \(t_R\) steps.
2. **Advice phase (A):** the mobile agent follows the advice for \(t_A\) steps.
3. **Exploration phase (E):** the mobile agent explores locally, that is, visits all the nodes at distance up to a given radius \(r_E\) from its location at the start of the phase.

**Theorem 3.** For a graph \(G \in G_{n,\Delta}\), the \(R/A/E\) algorithm with parameters: \(t_R = M_{n,\Delta}\), \(t_A = r_1 - \log_{\Delta-1} k\), and \(r_E = \log_{\Delta-1} k + (r_2 - r_1)\), completes any search in expected time \(O(k \log n)\), a.a.s.

**Proof.** We will carry out an execution of three successive phases (random walk, advice, exploration) an iteration of the algorithm. The number of steps within each iteration is deterministically bounded by the sum of the durations of its three phases:

\[ t_R + t_A + O((\Delta - 1)^3) \leq O((\Delta - 1)^3) + O((\Delta - 1)^3) + O(k \log n) \]

\[ = O(k \log n). \]

To achieve \(O(k \log n)\) expected search time, it now suffices to prove that the probability of locating the target in any iteration is \(\Theta(1)\).

First, observe that regardless of the initial location of the agent, after completion of the random walk (R) phase, the agent is located at any node \(s\) of the graph with probability at least \(1/n - O(1/n^3)\) by the definition of the mixing time \(M_{n,\Delta}\). We remark that by Lemma 5, the distance between \(s\) and the target \(t\) is bounded by \(d(s, t) \leq diam(G) < (\Delta - 1)^2\), a.a.s.

The proof is completed by showing that if indeed \(d(s, t) < r_2\), then the probability of locating the target in any iteration of algorithm \(R/A/E\) is \(\Theta(1)\). Let \(\kappa\) denote the random variable describing the number of liars encountered by the agent during the advice (A) phase. Let \(s'\) be the location of the agent at the end of the advice phase. Observe that if \(\kappa = 0\), then either the target is reached in the advice phase, or in each of the \(t_A\) steps of the advice phase, the agent moves towards the target by a distance of 1, and consequently:

\[ d(s', t) = d(s, t) - t_A \leq r_2 - (r_1 - \log_{\Delta-1} k) = r_E. \]

Then, the target will be reached in the exploration (E) phase. So, it suffices to show that the event \(\kappa = 0\) occurs with probability \(\Theta(1)\). Observe that a sufficient condition for this event to occur is that the neighborhood \(N_s(t_A)\) does not contain any liars, or equivalently, that we have \(s \in V \setminus \bigcup_{L \subseteq L} N_s(t_A)\), where \(L \subseteq V\) is the set of liars. Since the graph is of maximum degree \(\Delta\), we have for any node \(l\):

\[ |N_l(t_A)| \leq (\Delta - 1)^{t_A} \leq \frac{1}{k} (\Delta - 1)^{r_1} \leq \frac{n}{2k}. \]

So, \(|\bigcup_{L \subseteq L} N_s(t_A)| \leq |L| \frac{n}{2k} = \frac{n}{2k}\), and \(|V \setminus \bigcup_{L \subseteq L} N_s(t_A)| \geq \frac{n}{2k}\).

Recall that \(s\) is chosen as any node from \(V\) with probability \(1/n - O(1/n^3)\), hence the event \(s \in V \setminus \bigcup_{L \subseteq L} N_s(t_A)\) occurs
with probability \( (\pi/2)(1/n - O(1/n^2)) = 1/2 - O(1/n^2) = \Theta(1) \), which completes the proof.

**Theorem 4.** For a graph \( G \in \mathcal{U}_{n,\Delta} \), the R/A algorithm with parameters: \( t_n = M_{s,\Delta} + \log_{\Delta} k + (r_2 - r_1) \) and \( t_\Delta = r_1 - \log_{\Delta} k \), completes any search in expected time \( O(k^2 \log^2 n) \), a.a.s.

**Proof.** Observe that the only difference between algorithms R/A/E and R/A is only in the computation of 4 consecutive steps of the optimal binary search tree having the target node \( \Theta(1) \), which completes the proof. Hence, by Lemma 8, the target has been found with probability \( 1 - 1/2^{2\Omega(c)} \). It suffices to note that the \( j \)-th step is completed within \( \Theta(j^2) \) phases, or equivalently within \( \Theta(j^2) = \Theta(c^2 \cdot k^3 \log^2 n) \) steps.

**3.3 Lower Bound**

Before proving the main theorem, we put forward a lower bound for a search scenario in a tree, which is a modification of the formulation of the known lower bound for trees from [20].

**Proposition 1.** Let \( T = (V_T, E_T) \) be a complete \((\Delta - 1)\)-ary tree having \( l \) levels with a root at node \( s \), let \( a \) be a known value, and let \( t \) be a target node chosen uniformly at random from the set of nodes of \( T \) such that \( d(s,t) = a \). Assume that for all nodes of \( T \), the advice is directed towards \( s \), and the direction of the advice of \( s \) is arbitrarily chosen. Then in expectation any search strategy requires \( \Omega((\Delta - 1)^3) \) steps to reach target \( t \) from source \( s \).

**Proof.** The distribution of the advice in the given tree is independent of the location of the target \( t \). Hence, the expected search time from \( s \) to \( t \) cannot be improved with respect to strategies in a model with no advice.

**Theorem 6.** For a graph \( G \in \mathcal{U}_{n,\Delta} \), with \( k \) lies and source-target distance \( d \), any search strategy requires \( \Omega(\min((\Delta - 1)^3, (\Delta - 1)^d) \log(\log n)) \) steps in expectation, a.a.s.

**Proof.** Let us denote \( r = \lfloor \frac{1}{2} \log_{\Delta} n \rfloor \), where \( c \) is the constant from Lemma 6. The number of required steps is not less than the source-target distance, and so, if \( d \geq r \), since \( r = \Theta(\log n) \), the claim clearly holds. We can thus assume that \( d < r \).

Consider a searching process in which the source vertex \( s \) is such that \( B_s(2r) \) is a complete \((\Delta - 1)\)-tree (such a vertex can be found a.a.s. by Lemma 6), and let the target vertex \( t \) be chosen uniformly at random subject to the constraint \( d(s,t) = d \). Since \( B_s(d) \) is a \((\Delta - 1)\)-tree, there exists a unique shortest path \( P \) of length \( d \) between \( s \) and \( t \). If \( d < d_-1 \), let the \( k \) nodes of path \( P \) which are closest to \( s \) be lies, whose advice is directed along path \( P \) to vertex \( s \). If \( d \geq d_-1 \), let all vertices of \( P \) (excluding \( s \) and \( t \)) be likewise defined as lies, and place the remaining lies at arbitrary nodes of \( G \) outside \( N_s(r) \).

Observe that, for any \( v \in N_s(r) \), we have \( d(v, t) \leq d(s, v) + d(s, t) \leq 2r \), and a path of length at most \( 2r \) connects \( v \) and \( t \) in \( B_s(r) \). This path is the unique shortest path connecting \( v \) and \( t \) in \( G \), since otherwise there would have to be another path in \( G \) of length at most \( 2r \) connecting \( v \) with \( t \) and the tree \( B_s(2r) \) would contain a cycle, a contradiction. We will say that a node \( v \in V \) gives useless advice if the following conditions are fulfilled: \( v \in N_s(r), v \neq t \), and \( v \) directs the agent along the path in the tree \( B_s(2r) \) leading from \( v \) to \( s \). Since for all \( v \in N_s(r) \) \( \setminus P \), the unique shortest path from \( v \) to \( t \) in \( G \) leads via \( s \), we obtain that all nodes \( v \in N_s(r) \setminus P \) give useless advice. Moreover, all of the lies located on path \( P \) give useless advice by construction of the set of advice for lies.

To complete the proof, we will consider two possibilities:

- The search strategy reaches the target \( t \), visiting only nodes which give useless advice. Then, until \( t \) is discovered, the visited subgraph must be a subtree of \( B_s(r) \). The searching time for \( t \) is the same as in a
corresponding set-up for the complete \((\Delta-1)\)-ary tree with useless advice, and by Proposition 1, we obtain that in expectation the number of required steps is
\[\Omega((\Delta-1)^{d(s,v)}).\]

- Some of the nodes encountered in the search do not give useless advice. Let \(v\) be the first such encountered node. If \(v \notin N_r(r)\), then \(d(s,v) > r\), so the search process takes \(\Omega(r) = \Omega(\log_{\Delta-1} n)\) steps. Otherwise, the search process before encountering node \(v\) must be confined to the tree \(N_r(r)\), and node \(v\) must lie on the path \(P\). Node \(v\) is necessarily the node of \(P\) located closest to \(s\) which is not a liar, i.e., \(d(s,v) = k + 1\). By applying Proposition 1 analogously as before, we have that the number of steps of the search which are performed in expectation before reaching \(v\) is at least
\[\Omega((\Delta-1)^{k(s,v)}).\]

The claimed bound is the minimum of the obtained bounds, taken over the considered cases.

4. CONCLUSIONS

We have shown that there exists a simple, generic searching strategy which, for network topologies such as the path (and ring) or random regular graphs, allows an anonymous agent to locate the target efficiently. The number of moves is polynomial in the number of faults which appear in the network. The proposed R/A strategy is based on alternating phases in which the agent follows advice and performs a random walk. The precise duration of the phases has to be fine-tuned, depending on the graph class in which the search is performed (Theorems 2 and 4). As a matter of fact, by a minor modification to the proofs, it is possible to select a phase duration which gives good results for both of the specific graph classes studied in this paper.

The persistent memory of the agent following the R/A strategy is limited to a timer which counts the number of performed steps. Somewhat surprisingly, the memory requirement cannot be decreased to 0 without affecting the performance of the algorithm even on the path.

It is natural to ask if the proposed R/A strategy or its variants may be applied to other graph classes, with a searching time polynomial in the number of liars and the distance to the target. Whereas the approach applied for paths in Section 2 generalizes e.g. to some cases of 2-dimensional grids, we leave this question open, e.g., for the much wider class of graphs of bounded doubling dimension.

5. ACKNOWLEDGMENTS

Works of N. Hanusse and D. Ilicinkas are partially supported by ANR ALADDIN and N. Nisse is partially supported by ANRs AGAPE and DIMAGREEN. These three authors are partially supported by the DCR project.

6. REFERENCES


