Binary Search in Graphs Revisited *

Argyrios Deligkas[1], George B. Mertzios[2], and Paul G. Spirakis[3]

1 **Faculty of Industrial Engineering and Management, Technion, Israel.**
`argyris@technion.ac.il`
2 **School of Engineering and Computing Sciences, Durham University, UK.**
`george.mertzios@durham.ac.uk`
3 **Department of Computer Science, University of Liverpool, UK, University of Patras, Greece, and CTI, Greece.**
`p.spirakis@liverpool.ac.uk`

Abstract

In the classical binary search in a path the aim is to detect an unknown target by asking as few queries as possible, where each query reveals the direction to the target. This binary search algorithm has been recently extended by [Emamjomeh-Zadeh et al., *STOC*, 2016] to the problem of detecting a target in an arbitrary graph. Similarly to the classical case in the path, the algorithm of Emamjomeh-Zadeh et al. maintains a candidates' set for the target, while each query asks an appropriately chosen vertex– the "median"–which minimises a potential Φ among the vertices of the candidates' set. In this paper we address three open questions posed by Emamjomeh-Zadeh et al., namely (a) detecting a target when the query response is a direction to an *approximately shortest path* to the target, (b) detecting a target when querying a vertex that is an *approximate median* of the current candidates' set (instead of an exact one), and (c) detecting *multiple targets*, for which to the best of our knowledge no progress has been made so far. We resolve questions (a) and (b) by providing appropriate upper and lower bounds, as well as a new potential Γ that guarantees efficient target detection even by querying an approximate median each time. With respect to (c), we initiate a systematic study for detecting two targets in graphs and we identify sufficient conditions on the queries that allow for strong (linear) lower bounds and strong (polylogarithmic) upper bounds for the number of queries. All of our positive results can be derived using our new potential Γ that allows querying approximate medians.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems]; G.2.2 [Discrete Mathematics]:[Graph Theory].

Keywords and phrases binary search, graph, approximate query, probabilistic algorithm, lower bound.

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.20

1 Introduction

The classical binary search algorithm detects an unknown target (or "treasure") t on a path with n vertices by asking at most $\log n$ queries to an oracle which always returns the direction from the queried vertex to t. To achieve this upper bound on the number of queries, the algorithm maintains a set of candidates for the place of t; this set is always a sub-path, and initially it is the whole path. Then, at every iteration, the algorithm queries the middle vertex ("median") of this candidates' set and, using the response of the query, it excludes

* Partially supported by ISF 2021296, EPSRC grant EP/P020372/1, EPSRC grant EP/P02002X/1, and Liverpool EEECS initiative NeST

42nd International Symposium on Mathematical Foundations of Computer Science (MFCS).
Editors: Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin; Article No. 20; pp. 20:1–20:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

either the left or the right half of the set. This way of searching for a target in a path can be naturally extended to the case where t lies on an n-vertex tree, again by asking at most $\log n$ queries that reveal the direction in the (unique) path to t [22]. The principle of the binary search algorithm on trees is based on the same idea as in the case of a path: for every tree there exists a separator vertex such that each of its subtrees contains at most half of the vertices of the tree [14], which can be also efficiently computed.

Due to its prevalent nature in numerous applications, the problem of detecting an unknown target in an arbitrary graph or, more generally in a search space, has attracted many research attempts from different viewpoints. Only recently the binary search algorithm with $\log n$ direction queries has been extended to arbitrary graphs by Emamjomeh-Zadeh et al. [10]. In this case there may exist multiple paths, or even multiple shortest paths form the queried vertex to t. The direction query considered in [10] either returns that the queried vertex q is the sought target t, or it returns an arbitrary direction from q to t, i.e. an arbitrary edge incident to q which lies on a shortest path from q to t. The main idea of this algorithm follows again the same principle as for paths and trees: it always queries a vertex that is the "median" of the current candidates' set and any response to the query is enough to shrink the size of the candidates' set by a factor of at least 2. Defining what the "median" is in the case of general graphs now becomes more tricky: Emamjomeh-Zadeh et al. [10] define the median of a set S as the vertex q that minimizes a potential function Φ, namely the sum of the distances from q to all vertices of S.

Apart from searching for upper bounds on the number of queries needed to detect a target t in graphs, another point of interest is to derive algorithms which, given a graph G, compute the *optimal* number of queries needed to detect an unknown target in G (in the worst case). This line of research was initiated in [18] where the authors studied directed acyclic graphs (DAGs). Although computing a query-optimal algorithm is known to be NP-hard on general graphs [4, 8, 16], there exist efficient algorithms for trees; after a sequence of papers [1, 13, 17, 19, 26], linear time algorithms were found in [19, 22]. Different models with queries of non-uniform costs or with a probability distribution over the target locations were studied in [5–7, 15].

A different line of research is to search for upper bounds and information-theoretic bounds on the number of queries needed to detect a target t, assuming that the queries incorporate some degree of "noise". In one of the variations of this model [2, 10, 11], each query independently returns with probability $p > \frac{1}{2}$ a direction to a shortest path from the queried vertex q to the target, and with probability $1 - p$ an arbitrary edge (possibly adversarially chosen) incident to q. The study of this problem was initiated in [11], where $\Omega(\log n)$ and $O(\log n)$ bounds on the number of queries were established for a path with n vertices. This information-theoretic lower bound of [11] was matched by an improved upper bound in [2]. The same matching bound was extended to general graphs in [10].

In a further "noisy" variation of binary search, every vertex v of the graph is assigned a fixed edge incident to v (also called the "advice" at v). Then, for a fraction $p > \frac{1}{2}$ of the vertices, the advice directs to a shortest path towards t, while for the rest of the vertices the advice is arbitrary, i.e. potentially misleading or adversarially chosen [3]. This problem setting is motivated by the situation of a tourist driving a car in an unknown country that was hit by a hurricane which resulted in some fraction of road-signs being turned in an arbitrary and unrecognizable way. The question now becomes whether it is still possible to navigate through such a disturbed and misleading environment and to detect the unknown target by asking only few queries (i.e. taking advice only from a few road-signs). It turns out that, apart from its obvious relevance to data structure search, this problem also appears in artificial intelligence as it can model searching using unreliable heuristics [3, 20, 23]. Moreover this problem also finds applications outside computer science, such as in navigation issues in the context of collaborative transport by ants [12].

Another way of incorporating some "noise" in the query responses, while trying to detect a target, is to have *multiple targets* hidden in the graph. Even if there exist only two unknown targets t_1 and t_2, the response of each query is potentially confusing even if *every* query correctly directs to a shortest path from the queried vertex to one of the targets. The reason of confusion is that now a detecting algorithm does not know to *which* of the hidden targets each query directs. In the context of the above example of a tourist driving a car in an unknown country, imagine there are two main football teams, each having its own stadium. A fraction $0 < p_1 < 1$ of the population supports the first team and a fraction $p_2 = 1 - p_1$ the second one, while the supporters of each team are evenly distributed across the country. The driver can now ask questions of the type "where is the football stadium?" to random local people along the way, in an attempt to visit *both* stadiums. Although every response will be honest, the driver can never be sure which of the two stadiums the local person meant. Can the tourist still detect both stadiums quickly enough? To the best of our knowledge the problem of detecting multiple targets in graphs has not been studied so far; this is one of the main topics of the present paper.

The problem of detecting a target within a graph can be seen as a special case of a two-player game introduced by Renyi [25] and rediscovered by Ulam [27]. This game does not necessarily involve graphs: the first player seeks to detect an element known to the second player in some search space with n elements. To this end, the first player may ask arbitrary yes/no questions and the second player replies to them honestly or not (according to the details of each specific model). Pelc [24] gives a detailed taxonomy for this kind of games. *Group testing* is a sub-category of these games, where the aim is to detect all unknown objects in a search space (not necessarily a graph) [9]. Thus, group testing is related to the problem of detecting multiple targets in graphs, which we study in this paper.

1.1 Our contribution

In this paper we systematically investigate the problem of detecting one or multiple hidden targets in a graph. Our work is driven by the open questions posed by the recent paper of Emamjomeh-Zadeh et al. [10] which dealt with the detection of a single target with and without "noise". More specifically, Emamjomeh-Zadeh et al. [10] asked for further fundamental generalizations of the model which would be of interest, namely (a) detecting a single target when the query response is a direction to an *approximately shortest path*, (b) detecting a single target when querying a vertex that is an *approximate median* of the current candidates' set S (instead of an exact one), and (c) detecting *multiple targets*, for which to the best of our knowledge no progress has been made so far.

We resolve question (a) in Section 2.1 by proving that *any* algorithm requires $\Omega(n)$ queries to detect a single target t, assuming that a query directs to a path with an approximately shortest length to t. Our results hold essentially for any approximation guarantee, i.e. for 1-additive and for $(1 + \varepsilon)$-multiplicative approximations.

Regarding question (b), we first prove in Section 2.2 that, for any constant $0 < \varepsilon < 1$, the algorithm of [10] requires at least $\Omega(\sqrt{n})$ queries when we query each time an $(1 + \varepsilon)$-approximate median (i.e. an $(1 + \varepsilon)$-approximate minimizer of the potential Φ over the candidates' set S). Second, to resolve this lower bound, we introduce in Section 2.3 a new potential Γ. This new potential can be efficiently computed and, in addition, guarantees that, for any constant $0 \le \varepsilon < 1$, the target t can be detected in $O(\log n)$ queries even when an $(1+\varepsilon)$-approximate median (with respect to Γ) is queried each time.

Regarding question (c), we initiate in Section 3 the study for detecting multiple targets on graphs by focusing mainly to the case of two targets t_1 and t_2. We assume throughout that every query provides a correct answer, in the sense that it always returns a direction to a shortest path from the queried vertex either to t_1 or to t_2. The "noise" in this case is that the algorithm does not know whether a query is returning a direction to t_1 or to t_2. Initially we observe in Section 3 that *any* algorithm requires $\frac{n}{2}-1$ (resp. $n-2$) queries in the worst case to detect one target (resp. both targets) if each query directs adversarially to one of the two targets. Hence, in the remainder of Section 3, we consider the case where each query independently directs to the first target t_1 with a constant probability p_1 and to the second target t_2 with probability $p_2 = 1 - p_1$. For the case of trees, we prove in Section 3 that both targets can be detected with high probability within $O(\log n)$ queries.

For general graphs, we distinguish between *biased* queries ($p_1 > p_2$) in Section 3.1 and *unbiased* queries ($p_1 = p_2 = \frac{1}{2}$) in Section 3.2. For biased queries, we observe that we can utilize the algorithm of Emamjomeh-Zadeh et al. [10] to detect the first target t_1 with high probability in $O(\log n)$ queries; this can be done by considering the queries that direct to t_2 as "noise". Thus our objective becomes to detect the target t_2 in a polylogarithmic number of queries. Notice here that we cannot apply the "noisy" framework of [10] to detect the second target t_2 , since now the "noise" is larger than $\frac{1}{2}$. We derive a probabilistic algorithm that overcomes this problem and detects the target t_2 with high probability in $O(\Delta \log^2 n)$ queries, where Δ is the maximum degree of a vertex in the graph. Thus, whenever $\Delta = O(poly \log n)$, a polylogarithmic number of queries suffices to detect t_2. In contrast, we prove in Section 3.2 that, for unbiased queries, *any* deterministic (possibly adaptive) algorithm that detects at least one of the targets requires at least $\frac{n}{2}-1$ queries, even in an unweighted cycle. Extending this lower bound for two targets, we prove that, assuming $2c \geq 2$ different targets and unbiased queries, *any* deterministic (possibly adaptive) algorithm requires at least $\frac{n}{2}-c$ queries to detect one of the targets.

Departing from the fact that our best upper bound on the number of biased queries in Section 3.1 is not polylogarithmic when the maximum degree Δ is not polylogarithmic, we investigate in Section 4 several variations of queries that provide more informative responses. In Section 4.1 we turn our attention to "direction-distance" biased queries which return with probability p_i both the direction to a shortest path to t_i and the distance between the queried vertex and t_i. In Section 4.2 we consider another type of a biased query which combines the classical "direction" query and an edge-variation of it. For both query types of Sections 4.1 and 4.2 we prove that the second target t_2 can be detected with high probability in $O(\log^3 n)$ queries. Furthermore, in Sections 4.3 and 4.4 we investigate two further generalizations of the "direction" query which make the target detection problem trivially hard and trivially easy to solve, respectively.

Due to lack of space, the full paper with all proofs is included in a clearly marked Appendix, to be read at the discretion of the Program Committee.

1.2 Our Model and Notation

We consider connected, simple, and undirected graphs. A graph $G = (V, E)$, where $|V| = n$, is given along with a *weight function* $w : E \to \mathbb{R}^+$ on its edges; if $w(e) = 1$ for every $e \in E$ then G is *unweighted*. An edge between two vertices v and u of G is denoted by vu, and in this case v and u are said to be *adjacent*. The distance $d(v, u)$ between vertices v and u is the length of a shortest path between v and u with respect to the weight function w. Since the graphs we consider are undirected, $d(u, v) = d(v, u)$ for every pair of vertices v, u. Unless specified otherwise, all logarithms are taken with base 2. Whenever an event happens with probability at least $1 - \frac{1}{n^\alpha}$ for some $\alpha > 0$, we say that it happens *with high probability*.

The *neighborhood* of a vertex $v \in V$ is the set $N(v) = \{u \in V : vu \in E\}$ of its adjacent vertices. The cardinality of $N(v)$ is the *degree* $\deg(v)$ of v. The maximum degree among all vertices in G is denoted by $\Delta(G)$, i.e. $\Delta(G) = \max\{\deg(v) : v \in V\}$. For two vertices v and $u \in N(v)$ we denote by $N(v,u) = \{x \in V : d(v,x) = w(vu) + d(u,x)\}$ the set of vertices $x \in V$ for which there exists a shortest path from v to x, starting with the edge vu. Note that, in general, $N(u,v) \neq N(v,u)$. Let $T = \{t_1, t_2, \cdots, t_{|T|}\} \subseteq V$ be a set of (initially unknown) *target vertices*. A *direction query* (or simply *query*) at vertex $v \in V$ returns with probability p_i a neighbor $u \in N(v)$ such that $t_i \in N(u,v)$, where $\sum_{i=1}^{|T|} p_i = 1$. If there exist more than one such vertices $u \in N(v)$ leading to t_i via a shortest path, the direction query returns an arbitrary one among them, i.e. possibly chosen adversarially, unless specified otherwise. Moreover, if the queried vertex v is equal to one of the targets $t_i \in T$, this is revealed by the query with probability p_i.

2 Detecting a Unique Target

In this section we consider the case where there is only one unknown target $t = t_1$, i.e. $T = \{t\}$. In this case the direction query at vertex v always returns a neighbor $u \in N(v)$ such that $t \in N(u,v)$. For this problem setting, Emamjomeh-Zadeh et al. [10] provided a polynomial-time algorithm which detects the target t in at most $\log n$ direction queries. During its execution, the algorithm of [10] maintains a "candidates' set" $S \subseteq V$ such that always $t \in S$, where initially $S = V$. At every iteration the algorithm computes in polynomial time a vertex v (called the *median* of S) which minimizes a potential $\Phi_S(v)$ among all vertices of the current set S. Then it queries a median v of S and it reduces the candidates' set S to $S \cap N(v,u)$, where u is the vertex returned by the direction query at v. The upper bound $\log n$ of the number of queries in this algorithm follows by the fact that always $|S \cap N(v,u)| \leq \frac{|S|}{2}$, whenever v is the median of S.

2.1 Bounds for Approximately Shortest Paths

We provide lower bounds for both additive and multiplicative approximation queries. A *c-additive approximation query* at vertex $v \in V$ returns a neighbor $u \in N(v)$ such that $w(vu) + d(u,t) \leq d(v,t) + c$. Similarly, an $(1+\varepsilon)$-*multiplicative approximation query* at vertex $v \in V$ returns a neighbor $u \in N(v)$ such that $w(vu) + d(u,t) \leq (1+\varepsilon) \cdot d(v,t)$.

It is not hard to see that in the unweighted clique with n vertices any algorithm requires in worst case $n-1$ 1-additive approximation queries to detect the target t. Indeed, in this case $d(v,t) = 1$ for every vertex $v \neq t$, while every vertex $u \notin \{v,t\}$ is a valid response of an 1-additive approximation query at v. Since in the case of the unweighted clique an additive 1-approximation is the same as a multiplicative 2-approximation of the shortest path, it remains unclear whether 1-additive approximation queries allow more efficient algorithms for graphs with large diameter. In the next theorem we strengthen this result to graphs with unbounded diameter.

▶ **Theorem 1.** *Assuming* 1*-additive approximation queries, any algorithm requires at least* $n-1$ *queries to detect the target* t*, even in graphs with unbounded diameter.*

In the next theorem we extend Theorem 1 by showing a lower bound of $n \cdot \frac{\varepsilon}{4}$ queries when we assume $(1+\varepsilon)$-multiplicative approximation queries.

▶ **Theorem 2.** *Let $\varepsilon > 0$. Assuming $(1+\varepsilon)$-multiplicative approximation queries, any algorithm requires at least at least $n \cdot \frac{\varepsilon}{4}$ queries to detect the target t.*

2.2 Lower Bound for querying the Approximate Median

The potential $\Phi_S : V \to \mathbb{R}^+$ of [10], where $S \subseteq V$, is defined as follows. For any set $S \subseteq V$ and any vertex $v \in V$, the potential of v is $\Phi_S(v) = \sum_{u \in S} d(v, u)$. A vertex $x \in V$ is an $(1+\varepsilon)$-approximate minimizer for the potential Φ over a set S (i.e. an $(1+\varepsilon)$-median of S) if $\Phi_S(x) \leq (1+\varepsilon) \min_{v \in V} \Phi_S(v)$, where $\varepsilon > 0$. We prove that an algorithm querying at each iteration always an $(1+\varepsilon)$-median of the current candidates' set S needs $\Omega(\sqrt{n})$ queries.

▶ **Theorem 3.** *Let $\varepsilon > 0$. If the algorithm of [10] queries at each iteration an $(1+\varepsilon)$-median for the potential function Φ, then at least $\Omega(\sqrt{n})$ queries are required to detect the target t in a graph G with n vertices, even if the graph G is a tree.*

2.3 Upper Bound for querying the Approximate Median

In this section we introduce a new potential function $\Gamma_S : V \to \mathbb{N}$ for every $S \subseteq V$, which overcomes the problem occured in Section 2.2. This new potential guarantees efficient detection of t in at most $O(\log n)$ queries, even when we always query an $(1+\varepsilon)$-median of the current candidates' set S (with respect to the new potential Γ), for any constant $0 < \varepsilon < 1$. Our algorithm is based on the approach of [10], however we now query an approximate median of the current set S with respect to Γ (instead of an exact median with respect to Φ of [10]).

▶ **Definition 4** (Potential Γ). Let $S \subseteq V$ and $v \in V$. Then $\Gamma_S(v) = \max\{|N(v, u) \cap S| : u \in N(v)\}$.

▶ **Theorem 5.** *Let $0 \leq \varepsilon < 1$. There exists an efficient adaptive algorithm which detects the target t in at most $\frac{\log n}{1 - \log(1+\varepsilon)}$ queries, by querying at each iteration an $(1+\varepsilon)$-median for the potential function Γ.*

Proof. Our proof closely follows the proof of Theorem 3 of [10]. Let $S \subseteq V$ be an arbitrary set of vertices of G such that $t \in S$. We will show that there exists a vertex $v \in V$ such that $\Gamma_S(v) \leq \frac{|S|}{2}$. First recall the potential $\Phi_S(v) = \sum_{x \in S} d(v, x)$. Let now $v_0 \in V$ be a vertex such that $\Phi_S(v_0)$ is minimized, i.e. $\Phi_S(v_0) \leq \Phi_S(v)$ for every $v \in V$. Let $u \in N(v_0)$ be an *arbitrary* vertex adjacent to v_0. We will prove that $|N(v_0, u) \cap S| \leq \frac{|S|}{2}$. Denote $S^+ = N(v_0, u) \cap S$ and $S^- = S \setminus S^+$. By definition, for every $x \in S^+$, the edge $v_0 u$ lies on a shortest path from v_0 to x, and thus $d(u, x) = d(v_0, x) - w(v_0 u)$. On the other hand, trivially $d(u, x) \leq d(v_0, x) + w(v_0 u)$ for every $x \in S$, and thus in particular for every $x \in S^-$. Therefore $\Phi_S(v_0) \leq \Phi_S(u) \leq \Phi_S(v_0) + (|S^-| - |S^+|) \cdot w(v_0 u)$, and thus $|S^+| \leq |S^-|$. That is, $|N(v_0, u) \cap S| = |S^+| \leq \frac{|S|}{2}$, since $S^- = S \setminus S^+$. Therefore which then implies that $\Gamma_S(v_0) \leq \frac{|S|}{2}$ as the choice of the vertex $u \in N(v_0)$ is arbitrary.

Let $v_m \in V$ be an exact median of S with respect to Γ. That is, $\Gamma_S(v_m) \leq \Gamma_S(v)$ for every $v \in V$. Note that $\Gamma_S(v_m) \leq \Gamma_S(v_0) \leq \frac{|S|}{2}$. Now let $0 \leq \varepsilon < 1$ and let $v_a \in V$ be an $(1+\varepsilon)$-median of S with respect to Γ. Then $\Gamma_S(v_a) \leq (1+\varepsilon)\Gamma_S(v_m) \leq \frac{1+\varepsilon}{2}|S|$. Our adaptive algorithm proceeds as follows. Similarly to the algorithm of [10] (see Theorem 3 of [10]), our adaptive algorithm maintains a candidates' set S, where initially $S = V$. At every iteration our algorithm queries an arbitrary $(1+\varepsilon)$-median $v_m \in V$ of the current set S with respect to the potential Γ. Let $u \in N(v_m)$ be the vertex returned by this query; the algorithm updates S with the set $N(v, u) \cap S$. Since $\Gamma_S(v_a) \leq \frac{1+\varepsilon}{2}|S|$ as we proved above, it follows that the updated candidates' set has cardinality at most $\frac{1+\varepsilon}{2}|S|$. Thus, since initially $|S| = n$, our algorithm detects the target t after at most $\log_{\left(\frac{2}{1+\varepsilon}\right)} n = \frac{\log n}{1-\log(1+\varepsilon)}$ queries. ◀

Notice in the statement of Theorem 5 that for $\varepsilon = 0$ (i.e. when we always query an exact median) we get an upper bound of $\log n$ queries, as in this case the size of the candidates' set decreases by a factor of at least 2. Furthermore notice that the reason that the algorithm of [10] is not query-efficient when querying an $(1 + \varepsilon)$-median is that the potential $\Phi_S(v)$ of [10] can become quadratic in $|S|$, while on the other hand the value of our potential $\Gamma_S(v)$ can be at most $|S|$ by Definition 4, for every $S \subseteq V$ and every $v \in V$. Furthermore notice that, knowing only the value $\Phi_S(v)$ for some vertex $v \in V$ is not sufficient to provide a guarantee for the proportional reduction of the set S when querying v. In contrast, just knowing the value $\Gamma_S(v)$ directly provides a guarantee that, if we query vertex v the set S will be reduced by a proportion of $\frac{\Gamma_S(v)}{|S|}$, regardless of the response of the query. Therefore, in practical applications, we may not need to necessarily compute an (exact or approximate) median of S to make significant progress.

3 Detecting Two Targets

In this section we consider the case where there are two unknown targets t_1 and t_2, i.e. $T = \{t_1, t_2\}$. In this case the direction query at vertex v returns with probability p_1 (resp. with probability $p_2 = 1 - p_1$) a neighbor $u \in N(v)$ such that $t_1 \in N(v, u)$ (resp. $t_2 \in N(v, u)$). Detecting more than one unknown targets has been raised as an open question by Emamjomeh-Zadeh et al. [10], while to the best of our knowledge no progress has been made so far in this direction. Here we deal with both problems of detecting at least one of the targets and detecting both targets. We study several different settings and derive both positive and negative results for them. Each setting differs from the other ones on the "freedom" the adversary has on responding to queries, or on the power of the queries themselves. We will say that the response to a query *directs to* t_i, where $i \in \{1, 2\}$, if the vertex returned by the query lies on a shortest path between the queried vertex and t_i.

It is worth mentioning here that, if an adversary would be free to arbitrarily choose which t_i each query directs to (i.e. instead of directing to t_i with probability p_i), then any algorithm would require at least $\lfloor \frac{n}{2} \rfloor$ (resp. $n-2$) queries to detect at least one of the targets (resp. both targets), even when the graph is a path. Indeed, consider a path $v_1, \ldots, v_n$ where $t_1 \in \{v_1, \ldots, v_{\lfloor \frac{n}{2} \rfloor}\}$ and $t_2 \in \{v_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, v_n\}$. Then, for every $i \in \{1, \ldots, \lfloor \frac{n}{2} \rfloor\}$, the query at v_i would return v_{i+1}, i.e. it would direct to t_2. Similarly, for every $i \in \{\lfloor \frac{n}{2} \rfloor + 1, \ldots, n\}$, the query at v_i would return v_{i-1}, i.e. it would direct to t_1. It is not hard to verify that in this case the adversary could "hide" the target t_1 at any of the first $\lfloor \frac{n}{2} \rfloor$ vertices which is not queried by the algorithm and the target t_2 on any of the last $n - \lfloor \frac{n}{2} \rfloor$ vertices which is not queried. Hence, at least $\lfloor \frac{n}{2} \rfloor$ queries (resp. $n - 2$ queries) would be required to detect one of the targets (resp. both targets) in the worst case.

As a warm-up, we provide in the next theorem an efficient algorithm that detects with high probability both targets in a tree using $O(\log^2 n)$ queries.

▶ **Theorem 6.** *For any constant* $0 < p_1 < 1$*, we can detect with probability at least* $\left(1 - \frac{\log n}{n}\right)^2$ *both targets in a tree with* n *vertices using* $O(\log^2 n)$ *queries.*

Since in a tree both targets t_1, t_2 can be detected with high probability in $O(\log^2 n)$ queries by Theorem 6, we consider in the remainder of the section arbitrary graphs instead of trees. First we consider in Section 3.1 *biased* queries, i.e. queries with $p_1 > \frac{1}{2}$. Second we consider in Section 3.2 *unbiased* queries, i.e. queries with $p_1 = p_2 = \frac{1}{2}$.

3.1 Upper Bounds for Biased Queries

In this section we consider biased queries which direct to t_1 with probability $p_1 > \frac{1}{2}$ and to t_2 with probability $p_2 = 1 - p_1 < \frac{1}{2}$. As we can detect in this case the first target t_1 with high probability in $O(\log n)$ queries by using the "noisy" framework of [10], our aim becomes to detect the second target t_2 with the fewest possible queries, once we have already detected t_1.

For every vertex v and every $i \in \{1, 2\}$, denote by $E_{t_i}(v) = \{u \in N(v) : t_i \in N(v, u)\}$ the set of neighbors of v such that the edge uv lies on a shortest path from v to t_i. Note that the sets $E_{t_1}(v)$ and $E_{t_2}(v)$ can be computed in polynomial time, e.g. using Dijkstra's algorithm. We assume that, once a query at vertex v has chosen which target t_i it directs to, it returns each vertex of $E_{t_i}(v)$ equiprobably and independently from all other queries. Therefore, each of the vertices of $E_{t_1}(v) \setminus E_{t_2}(v)$ is returned by the query at v with probability $\frac{p_1}{|E_{t_1}(v)|}$, each vertex of $E_{t_2}(v) \setminus E_{t_1}(v)$ is returned with probability $\frac{1-p_1}{|E_{t_2}(v)|}$, and each vertex of $E_{t_1}(v) \cap E_{t_2}(v)$ is returned with probability $\frac{p_1}{|E_{t_1}(v)|} + \frac{1-p_1}{|E_{t_2}(v)|}$. We will show in Theorem 8 that, under these assumptions, we detect the second target t_2 with high probability in $O(\Delta \log^2 n)$ queries where Δ is the maximum degree of the graph.

The high level description of our algorithm (Algorithm 1) is as follows. Throughout the algorithm we maintain a candidates' set S of vertices in which t_2 belongs with high probability. Initially $S = V$. In each iteration we first compute an (exact or approximate) median v of S with respect to the potential Γ (see Section 2.3). Then we compute the set $E_{t_1}(v)$ (this can be done as t_1 has already been detected) and we query $c\Delta \log n$ times vertex v, where $c = \frac{7(1+p_1)^2}{p_1(1-p_1)^2}$ is a constant. Denote by $Q(v)$ the multiset of size $c\Delta \log n$ that contains the vertices returned by these queries at v. If at least one of these $O(\Delta \log n)$ queries at v returns a vertex $u \notin E_{t_1}(v)$, then we can conclude that $u \in E_{t_2}(v)$, and thus we update the set S by $S \cap N(v, u)$. Assume otherwise that all $O(\Delta \log n)$ queries at v return vertices of $E_{t_1}(v)$. Then we pick a vertex $u_0 \in N(v)$ that has been returned most frequently among the $O(\Delta \log n)$ queries at v, and we update the set S by $S \cap N(v, u_0)$. As it turns out, $u_0 \in E_{t_2}(v)$ with high probability. Since we always query an (exact or approximate) median v of the current candidates' set S with respect to the potential Γ, the size of S decreases by a constant factor each time. Therefore, after $O(\log n)$ updates we obtain $|S| = 1$. It turns out that, with high probability, each update of the candidates' set was correct, i.e. $S = \{t_2\}$. Since for each update of S we perform $O(\Delta \log n)$ queries, we detect t_2 with high probability in $O(\Delta \log^2 n)$ queries in total.

Algorithm 1 Given t_1, detect t_2 with high probability with $O(\Delta \log^2 n)$ queries

```
S ← V;  c ← 7(1+p_1)^2 / (p_1(1−p_1)^2)
while |S| > 1 do
    Compute an (approximate) median v of S with respect to potential Γ;  Compute E_{t_1}(v)
    Query cΔ log n times vertex v;  Compute the multiset Q(v) of these query responses
    if Q(v) \ E_{t_1}(v) ≠ ∅ then
        Pick a vertex u ∈ Q(v) \ E_{t_1}(v) and set S ← S ∩ N(v, u)
    else
        Pick a most frequent vertex u ∈ Q(v) and set S ← S ∩ N(v, u)
return the unique vertex in S
```

Recall that every query at v returns a vertex $u \in E_{t_1}(v)$ with probability p_1 and a vertex $u \in E_{t_2}(v)$ with probability $1 - p_1$. Therefore, for every $v \in V$ the multiset $Q(v)$ contains at least one vertex $u \in E_{t_2}(v)$ with probability at least $1 - p_1^{|Q(v)|} = 1 - p_1^{|c\Delta \log n|}$. In the next lemma we prove that, every time we update S using Step 8, the updated set contains t_2 with high probability.

▶ **Lemma 7.** *Let $S \subseteq V$ such that $t_2 \in S$ and let $S' = S \cap N(v, u)$ be the updated set at Step 8 of Algorithm 1. Then $t_2 \in S'$ with probability at least $1 - \frac{2}{n}$.*

Proof. Let $\delta = \frac{1-p_1}{1+p_1}$ and $c = \frac{7(1+p_1)^2}{p_1(1-p_1)^2}$ be two constants. Recall that each of the vertices of $E_{t_1}(v) \setminus E_{t_2}(v)$ is returned by the query at v with probability $\frac{p_1}{|E_{t_1}(v)|}$, each vertex of $E_{t_2}(v) \setminus E_{t_1}(v)$ is returned with probability $\frac{1-p_1}{|E_{t_2}(v)|}$, and each vertex of $E_{t_1}(v) \cap E_{t_2}(v)$ is returned with probability $\frac{p_1}{|E_{t_1}(v)|} + \frac{1-p_1}{|E_{t_2}(v)|}$. Observe that these probabilities are the expected frequencies for these vertices in $Q(v)$. Recall that Step 8 is executed only in the case where $Q(v) \subseteq E_{t_1}(v)$. To prove the lemma it suffices to show that, whenever $Q(v) \subseteq E_{t_1}(v)$, the most frequent element of $Q(v)$ belongs to $E_{t_1}(v) \cap E_{t_2}(v)$ with high probability.

First note that, for the chosen value of δ,

$$(1+\delta)\frac{p_1}{|E_{t_1}(v)|} < (1-\delta)\left(\frac{p_1}{|E_{t_1}(v)|} + \frac{1-p_1}{|E_{t_2}(v)|}\right) \tag{1}$$

Let $u \in E_{t_1}(v) \setminus E_{t_2}(v)$, i.e. the query at v directs to t_1 but not to t_2. We define the random variable $Z_i(u)$, such that $Z_i(u) = 1$ if u is returned by the i-th query at v and $Z_i(u) = 0$ otherwise. Furthermore define $Z(u) = \sum_{i=1}^{c\Delta \log n} Z_i(u)$. Since $\Pr(Z_i(u) = 1) = \frac{p_1}{|E_{t_1}(v)|}$, it follows that $E(Z(u)) = c\Delta \log n \frac{p_1}{|E_{t_1}(v)|}$ by the linearity of expectation. Then, using Chernoff's bounds we can prove that

$$\Pr(Z(u) \geq (1+\delta)E(Z(u))) \leq \frac{1}{n^2}. \tag{2}$$

Thus (2) implies that the probability that there exists at least one $u \in E_{t_1}(v) \setminus E_{t_2}(v)$ such that $Z(u) \geq (1+\delta)E(Z(u))$ is

$$\Pr\left(\exists u \in E_{t_1}(v) \setminus E_{t_2}(v) : Z(u) \geq (1+\delta)\frac{p_1}{|E_{t_1}(v)|}\right) < (\Delta - 1)\frac{1}{n^2} < \frac{1}{n}. \tag{3}$$

Now let $u' \in E_{t_1}(v) \cap E_{t_2}(v)$. Similarly to the above we define the random variable $Z'_i(u')$, such that $Z'_i(u') = 1$ if u' is returned by the i-th query at v and $Z'_i(u') = 0$ otherwise. Furthermore define $Z'(u') = \sum_{i=1}^{c\Delta \log n} Z'_i(u')$. Since $\Pr(Z'_i(u') = 1) = \frac{p_1}{|E_{t_1}(v)|} + \frac{1-p_1}{|E_{t_2}(v)|}$, it follows that $E(Z(u)) = c\Delta \log n \left(\frac{p_1}{|E_{t_1}(v)|} + \frac{1-p_1}{|E_{t_2}(v)|}\right)$ by the linearity of expectation. Then we obtain similarly to (2) that

$$\Pr(Z'(u') \leq (1-\delta)E(Z'(u'))) < \frac{1}{n^2} \tag{4}$$

Thus, it follows by the union bound and by (1), (3), and (4) that

$$\Pr(\exists u \in E_{t_1}(v) \setminus E_{t_2}(v) : Z(u) \geq Z'(u')) \leq \frac{2}{n}. \tag{5}$$

That is, the most frequent element of $Q(v)$ belongs to $E_{t_1}(v) \cap E_{t_2}(v)$ with probability at least $1 - \frac{2}{n}$. This completes the proof of the lemma. ◀

With Lemma 7 in hand we can now prove the main theorem of the section.

▶ **Theorem 8.** *Given t_1, Algorithm 1 detects t_2 in $O(\Delta \log^2 n)$ queries with probability at least $(1 - \frac{2}{n})^{O(\log n)}$.*

Note by Theorem 8 that, whenever $\Delta = O(poly \log n)$ we can detect both targets t_1 and t_2 in $O(poly \log n)$ queries. However, for graphs with larger maximum degree Δ, the value of the maximum degree dominates any polylogarithmic factor in the number of queries. The intuitive reason behind this is that, for an (exact or approximate) median v of the current set S, whenever $\deg(v)$ and $E_{t_1}(v)$ are large and $E_{t_2}(v) \subseteq E_{t_1}(v)$, we can not discriminate with a polylogarithmic number of queries between the vertices of $E_{t_2}(v)$ and the vertices of $E_{t_1}(v) \setminus E_{t_2}(v)$ with large enough probability. Although this argument does not give any lower bound for the number of queries in the general case (i.e. when Δ is unbounded), it seems that more informative queries are needed to detect both targets with polylogarithmic queries in general graphs. We explore such more informative queries in Section 4.

3.2 Lower Bounds for Unbiased Queries

In this section we consider unbiased queries, i.e. queries which direct to each of the targets t_1, t_2 with equal probability $p_1 = p_2 = \frac{1}{2}$. In this setting every query is indifferent between the two targets, and thus the "noisy" framework of [10] cannot be applied for detecting any of the two targets. In particular we prove in the next theorem that any deterministic (possibly adaptive) algorithm needs at least $\frac{n}{2} - 1$ queries to detect one of the two targets.

▶ **Theorem 9.** *Let $p_1 = p_2 = \frac{1}{2}$. Then any deterministic (possibly adaptive) algorithm needs at least $\frac{n}{2} - 1$ queries to detect one of the two targets, even in an unweighted cycle.*

In the next theorem we generalize the lower bound of Theorem 9 to the case of $2c \geq 2$ different targets $T = \{t_1, t_2, \ldots, t_{2c}\}$ and the query to any vertex $v \notin T$ is unbiased, i.e. $p_i = \frac{1}{2c}$ for every $i \in \{1, 2, \ldots, 2c\}$.

▶ **Theorem 10.** *Suppose that there are $2c$ targets in the graph and let $p_i = \frac{1}{2c}$ for every $i \in \{1, 2, \ldots, 2c\}$. Then, any deterministic (possibly adaptive) algorithm requires at least $\frac{n}{2} - c$ queries to locate at least one target, even in an unweighted cycle.*

4 More Informative Queries for Two Targets

A natural alternative to obtain query-efficient algorithms for multiple targets, instead of restricting the maximum degree Δ of the graph (see Section 3.1), is to consider queries that provide more informative responses in general graphs. As we have already observed in Section 3.1, it is not clear whether it is possible to detect multiple targets with $O(poly \log n)$ *direction queries* in an arbitrary graph. In this section we investigate natural variations and extensions of the direction query for multiple targets which we studied in Section 3.

4.1 Direction-Distance Biased Queries

In this section we strengthen the direction query in a way that it also returns the value of the distance between the queried vertex and one of the targets. More formally, a *direction-distance query* at vertex $v \in V$ returns with probability p_i a pair (u, ℓ), where $u \in N(v)$ such that $t_i \in N(u, v)$ and $d(v, t_i) = \ell$. Note that here we impose again that all p_i's are constant and that $\sum_{i=1}^{|T|} p_i = 1$, where $T = \{t_1, t_2, \ldots, t_{|T|}\}$ is the set of targets. We will say that the response (u, ℓ) to a direction-distance query at vertex v *directs to* t_i if $t_i \in N(v, u)$ and $\ell = d(v, t_i)$. Similarly to our assumptions on the direction query, whenever there exist more than one such vertices $u \in N(v)$ leading to t_i via a shortest path, the direction-distance query returns an arbitrary vertex u among them (possibly chosen adversarially). Moreover, if the queried vertex v is equal to one of the targets $t_i \in T$, this is revealed by the query with probability p_i. These direction-distance queries have also been used in [10] for detecting one single target in directed graphs.

Here we consider the case of two targets and *biased queries*, i.e. $T = \{t_1, t_2\}$ where $p_1 > p_2$. Similarly to Section 3.1, initially we can detect the first target t_1 with high probability in $O(\log n)$ queries using the "noisy" model of [10]. Thus, in what follows we assume that t_1 has already been detected. We will show that the second target t_2 can be detected with high probability with $O(\log^3 n)$ additional direction-distance queries using Algorithm 2. The high level description of our algorithm is the following. We maintain a candidates' set S such that at every iteration $t_2 \in S$ with high probability. Each time we update the set S, its size decreases by a constant factor. Thus we need to shrink the set S at most $\log n$ times. In order to shrink S one time, we first compute an $(1+\varepsilon)$-median v of the current set S and we query $\log n$ times this vertex v. Denote by $Q(v)$ the set of all different responses of these $\log n$ direction-distance queries at v. As it turns out, the responses in $Q(v)$ might not always be enough to shrink S such that it still contains t_2 with high probability. For this reason we also query $\log n$ times each of the $\log n$ neighbors $u \in N(v)$, such that $(u, \ell) \in Q(v)$ for some $\ell \in \mathbb{N}$. After these $\log^2 n$ queries at v and its neighbors, we can safely shrink S by a constant factor, thus detecting the target t_2 with high probability in $\log^3 n$ queries.

For the description of our algorithm (see Algorithm 2) recall that, for every vertex v, the set $E_{t_1}(v) = \{u \in N(v) : t_1 \in N(v, u)\}$ contains all neighbors of v such that the edge uv lies on a shortest path from v to t_1.

Algorithm 2 Given t_1, detect t_2 with high probability with $O(\log^3 n)$ direction-distance queries

1: $S \leftarrow V$
2: **while** $|S| > 1$ **do**
3:    Compute an (approximate) median v of S with respect to potential Γ; Compute $E_{t_1}(v)$
4:    Query $\log n$ times vertex v; Compute the set $Q(v)$ of different query responses
5:    **if** there exists a pair $(u, \ell) \in Q(v)$ such that $u \notin E_{t_1}(v)$ or $\ell \neq d(v, t_1)$ **then**
6:       $S \leftarrow S \cap N(v, u)$
7:    **else**
8:       **for** every $(u, \ell) \in Q(v)$ **do**
9:          Query $\log n$ times vertex u; Compute the set $Q(u)$ of different query responses
10:          **if** for every $(z, \ell') \in Q(u)$ we have $\ell' = \ell - w(vu)$ **then**
11:             $S \leftarrow S \cap N(v, u)$; Goto line 2
12: **return** the unique vertex of S

▶ **Theorem 11.** *Given t_1, Algorithm 2 detects t_2 in at most $O(\log^3 n)$ queries with probability at least $1 - O\left(\log n \cdot p_1^{\log n}\right)$.*

4.2 Vertex-Direction and Edge-Direction Biased Queries

An alternative natural variation of the direction query is to *query an edge* instead of querying a vertex. More specifically, the direction query (as defined in Section 1.2) queries a vertex $v \in V$ and returns with probability p_i a neighbor $u \in N(v)$ such that $t_i \in N(u, v)$. Thus, as this query always queries a vertex, it can be also referred to as a *vertex-direction query*. Now we define the *edge-direction query* as follows: it queries an ordered pair of adjacent vertices (v, u) and it returns with probability p_i `YES` (resp. `NO`) if $t_i \in N(v, u)$ (resp. if $t_i \notin N(v, u)$). Similarly to our notation in the case of vertex-direction queries, we will say that the response `YES` (resp. `NO`) to an edge-direction query at the vertex pair (v, u) *refers* to t_i if $t_i \in N(v, u)$ (resp. if $t_i \notin N(v, u)$). Similar but different edge queries for detecting one single target on trees have been investigated in [10, 13, 21, 26].

Here we consider the case where both vertex-direction and edge-direction queries are available to the algorithm, and we focus again to the case of two targets and *biased queries*, i.e. $T = \{t_1, t_2\}$ where $p_1 > p_2$. Similarly to Sections 3.1 and 4.1, we initially detect t_1 with high probability in $O(\log n)$ vertex-direction queries using the "noisy" model of [10]. Thus, in the following we assume that t_1 has already been detected.

▶ **Theorem 12.** *Given t_1, it is possible to detect t_2 in at most $O(\log^2 n)$ vertex-direction queries and $O(\log^3 n)$ edge–direction queries with probability at least $1 - O(\log n \cdot p_1^{\log n})$.*

4.3 Two-Direction Queries

In this section we consider another variation of the direction query that was defined in Section 1.2 (or "vertex-direction query" in the terminology of Section 4.2), which we call *two-direction query*. Formally, a two-direction query at vertex v returns an *unordered pair* of (not necessarily distinct) vertices $\{u, u'\}$ such that $t_1 \in N(v, u)$ and $t_2 \in N(v, u')$. Note here that, as $\{u, u'\}$ is an unordered pair, the response of the two-direction query does not clarify which of the two targets belongs to $N(v, u)$ and which to $N(v, u')$.

Although this type of query may seem at first to be more informative than the standard direction query studied in Section 3, we show that this is not the case. Intuitively, this type of query resembles the unbiased direction query of Section 3.2. To see this, consider e.g. the unweighted cycle where the two targets are placed at two anti-diametrical vertices; then, applying many times the unbiased direction query of Section 3.2 at any specific vertex v reveals with high probability the same information as applying a single two-direction query at v. Based on this intuition the next theorem can be proved with exactly the same arguments as Theorem 9 of Section 3.2.

▶ **Theorem 13.** *Any deterministic (possibly adaptive) algorithm needs at least $\frac{n}{2} - 1$ two-direction queries to detect one of the two targets, even in an unweighted cycle.*

4.4 Restricted Set Queries

The last type of queries we consider is when the query is applied not only to a vertex v of the graph, but also to a subset $S \subseteq V$ of the vertices, and the response of the query is a vertex $u \in N(v)$ such that $t \in N(v, u)$ for at least one of the targets t that belong to the set S. Formally, let T be the set of targets. The *restricted-set query* at the pair (v, S), where $v \in V$ and $S \subseteq V$ such that $T \cap S \neq \emptyset$, returns a vertex $u \in N(v)$ such that $t \in N(v, u)$ for at least one target $t \in T \cap S$. If there exist multiple such vertices $u \in N(v)$, the query returns one of them adversarially. Finally, if we query a pair (v, S) such that $T \cap S = \emptyset$, then the query returns adversarially an arbitrary vertex $u \in N(v)$, regardless of whether the edge vu leads to a shortest path from v to any target in T. That is, the response of the query can be considered in this case as "noise". In the next theorem we prove that this query is very powerful, as $|T| \cdot \log n$ restricted-set queries suffice to detect all targets of the set T.

▶ **Theorem 14.** *Let T be the set of targets. There exists an adaptive deterministic algorithm that detects all targets of T with at most $|T| \cdot \log n$ restricted-set queries.*

References

1 Yosi Ben-Asher, Eitan Farchi, and Ilan Newman. Optimal search in trees. *SIAM J. Comput.*, 28(6):2090–2102, 1999.

2 Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 221–230, 2008.

3 Lucas Boczkowski, Amos Korman, and Yoav Rodeh. Searching on trees with noisy memory. *CoRR*, abs/1611.01403, 2016.

4 Renato Carmo, Jair Donadelli, Yoshiharu Kohayakawa, and Eduardo Sany Laber. Searching in random partially ordered sets. *Theor. Comput. Sci.*, 321(1):41–57, 2004.

5 Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Marco Molinaro. On the complexity of searching in trees and partially ordered structures. *Theor. Comput. Sci.*, 412(50):6879–6896, 2011.

6 Ferdinando Cicalese, Tobias Jacobs, Eduardo Sany Laber, and Caio Dias Valentim. The binary identification problem for weighted trees. *Theor. Comput. Sci.*, 459:100–112, 2012.

7 Constantinos Daskalakis, Richard M. Karp, Elchanan Mossel, Samantha Riesenfeld, and Elad Verbin. Sorting and selection in posets. *SIAM J. Comput.*, 40(3):597–622, 2011.

8 Dariusz Dereniowski. Edge ranking and searching in partial orders. *Discrete Applied Mathematics*, 156(13):2493–2500, 2008.

9 Dingzhu Du and Frank K. Hwang. *Combinatorial Group Testing and its Applications.* World Scientific, Singapore, 1993.

10 Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 519–532, 2016.

11 Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994.

12 Ehud Fonio, Yael Heyman, Lucas Boczkowski, Aviram Gelblum, Adrian Kosowski, Amos Korman, and Ofer Feinerman. A locally-blazed ant trail achieves efficient collective navigation despite limited information. *eLife*, page 23 pages, 2016.

13 Ananth V. Iyer, H. Donald Ratliff, and Gopalakrishnan Vijayan. Optimal node ranking of trees. *Inf. Process. Lett.*, 28(5):225–229, 1988.

14 C. Jordan. Sur les assemblages de lignes. *Journal f"ur die reine und angewandte Mathematik*, 70:195–190, 1869.

15 Eduardo Sany Laber, Ruy Luiz Milidiú, and Artur Alves Pessoa. On binary searching with non-uniform costs. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 855–864, 2001.

16 Tak Wah Lam and Fung Ling Yue. Edge ranking of graphs is hard. *Discrete Applied Mathematics*, 85(1):71–86, 1998.

17 Tak Wah Lam and Fung Ling Yue. Optimal edge ranking of trees in linear time. *Algorithmica*, 30(1):12–33, 2001.

18 Nathan Linial and Michael E. Saks. Searching ordered structures. *J. Algorithms*, 6(1):86–103, 1985.

19 Shay Mozes, Krzysztof Onak, and Oren Weimann. Finding an optimal tree searching strategy in linear time. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 1096–1105, 2008.

20 Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill Pub. Co., 1971.

21 Robert Nowak. Noisy generalized binary search. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1366–1374. Curran Associates, Inc., 2009.

22 Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 379–388, 2006.

23 Judea Pearl. *Heuristics - intelligent search strategies for computer problem solving.* Addison-Wesley series in artificial intelligence. Addison-Wesley, 1984.

24 Andrzej Pelc. Searching games with errors - fifty years of coping with liars. *Theor. Comput. Sci.*, 270(1-2):71–109, 2002.

25 Alfred Renyi. On a problem in information theory. *Magyar Tud. Akad. Mat. Kutato Int. Kozl*, 6(B):505–516, 1961.

26 Alejandro A. Sch"affer. Optimal node ranking of trees in linear time. *Information Processing Letters*, 33(2):91–96, 1989.

27 Stanislaw Ulam. *Adventures of a Mathematician.* University of California Press, 1991.