


A Framework for Searching in Graphs in the Presence of Errors

Dariusz Dereniowski¹

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology
Narutowicza 11/12, 80-233 Gdańsk, Poland
deren@eti.pg.edu.pl


 <https://orcid.org/0000-0003-4000-4818>

Stefan Tiegel

Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland
tiegels@student.ethz.ch


Przemysław Uznański

Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland
przemyslaw.uznanski@inf.ethz.ch

 <https://orcid.org/0000-0002-8652-0490>

Daniel Wolleb-Graf

Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland
daniel.graf@inf.ethz.ch

 <https://orcid.org/0000-0002-6137-5725>

Abstract

We consider a problem of searching for an unknown target vertex t in a (possibly edge-weighted) graph. Each *vertex-query* points to a vertex v and the response either admits that v is the target or provides any neighbor s of v that lies on a shortest path from v to t . This model has been introduced for trees by Onak and Parys [FOCS 2006] and for general graphs by Emamjomeh-Zadeh et al. [STOC 2016]. In the latter, the authors provide algorithms for the error-less case and for the independent noise model (where each query independently receives an erroneous answer with known probability $p < 1/2$ and a correct one with probability $1 - p$).

We study this problem both with adversarial errors and independent noise models. First, we show an algorithm that needs at most $\frac{\log_2 n}{1-H(r)}$ queries in case of *adversarial* errors, where the adversary is bounded with its rate of errors by a known constant $r < 1/2$. Our algorithm is in fact a simplification of previous work, and our refinement lies in invoking an amortization argument. We then show that our algorithm coupled with a Chernoff bound argument leads to a simpler algorithm for the independent noise model and has a query complexity that is both simpler and asymptotically better than the one of Emamjomeh-Zadeh et al. [STOC 2016].

Our approach has a wide range of applications. First, it improves and simplifies the Robust Interactive Learning framework proposed by Emamjomeh-Zadeh and Kempe [NIPS 2017]. Secondly, performing analogous analysis for *edge-queries* (where a query to an edge e returns its endpoint that is closer to the target) we actually recover (as a special case) a noisy binary search algorithm that is asymptotically optimal, matching the complexity of Feige et al. [SIAM J. Comput. 1994]. Thirdly, we improve and simplify upon an algorithm for searching of *unbounded* domains due to Aslam and Dhagat [STOC 1991].

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases graph algorithms, noisy binary search, query complexity, reliability

¹ Partially supported by National Science Centre (Poland) grant number 2015/17/B/ST6/01887.



Digital Object Identifier 10.4230/OASICS.SOSA.2019.4

Related Version <https://arxiv.org/abs/1804.02075>

1 Introduction

Consider the following game played on a simple connected graph $G = (V, E)$:

Initially, the *Responder* selects a *target* $v^* \in V$. In each *round*, the *Questioner* asks a *vertex-query* by pointing to a vertex v of G , and the *Responder* provides a *reply*. The reply either states that v is the target, i.e., $v = v^*$, or provides an edge incident to v that lies on a shortest path to the target, breaking ties arbitrarily. A specific number of replies can be erroneous (we call them *lies*). The goal is to design a strategy for the *Questioner* that identifies v^* using as few queries as possible.

We remark that this problem is known, among several other names, as Rényi-Ulam games [38, 41], *noisy binary search* or *noisy decision trees* [20, 24, 5]. One needs to put some restriction as how often the *Responder* is allowed to lie. Following earlier works, we focus on the most natural probabilistic model, in which each reply is independently correct with a certain fixed probability.

This problem has interesting applications in noisy interactive learning [1, 18, 25, 29, 40]. In general terms, the learning process occurs as a version of the following scheme. A user is presented with some *information* – this information reflects the current state of knowledge of the system and should take into account earlier interactions with the user (thus, the process is interactive). Then, the user responds, which provides a new piece of data to the system. In order to model such dynamics as our problem, one needs to place some rules: what the information should look like and what is allowed as a valid user’s response. A crucial element in those applications is the fact that the learning process (reflected by queries and responses) does not require an explicit construction of the underlying graph on which the process takes place. Instead, it is enough to argue that there *exists* a graph whose vertices reflect possible states. Moreover, this graph needs to have the property that a valid user’s response reveals an edge lying on a shortest path to the state that needs to be determined by the system. Specific applications pointed out in [18] are the following. In *learning a ranking* the system aims at learning user’s preference list [36, 30]. An information presented to the user is some list, and as a response the user swaps two consecutive elements on this list which are in the wrong order with respect to the user’s target preference list. Or, the response may reveal which element on a presented list has the highest rank. Both versions of the response turn out to be consistent with our graph-theoretic game over a properly defined graph, whose vertex set is the set of all possible preference lists. Another application is *learning a clustering*, where the user’s reply tells the system that in the current clustering some cluster needs to be split (the reply does not need to reveal how) or two clusters should be merged [3, 4]. Yet another application includes learning a binary classifier. The strength that comes from a graph-theoretic modeling of those applications as our game is that, although the underlying graph structure has usually exponential number of vertices (for learning a ranking it is $l!$, where l is the maximum length of the preference list), the number of required queries is asymptotically logarithmic in this size [19, 18]. Thus, the learning strategies derived from the algorithms in [19] and [18] turn out to be quite efficient. We stress out that the lies in the query game reflect the fact that the user may sometimes provide incorrect replies. We

also note that any improvement of those algorithms, at which we aim in this work, leads to immediate improvements in the above-mentioned applications.

In [19], the authors provide an algorithm with the following *query complexity*, i.e., the worst-case number of vertex-queries:

$$\frac{1}{1-H(p)} \left(\log_2 n + \mathcal{O}\left(\frac{1}{C} \log n + C^2 \log \delta^{-1}\right) \right), \text{ where } C = \max\left(\frac{1}{2} - p, \sqrt{\log \log n}, 1\right) \quad (1)$$

that identifies the target with probability at least $1 - \delta$, where n is the number of vertices of an input graph and $H(p) = -p \log p - (1-p) \log(1-p)$ is the entropy and p is the success probability of a query. It is further observed that when $p < 1/2$ is constant (w.r.t. to n), then (1) reduces to $\frac{\log_2 n}{1-H(p)} + o(\log n) + \mathcal{O}(\log^2 \delta^{-1})$. However, this complexity deteriorates when $1/2 - p = \mathcal{O}(1/\sqrt{\log \log n})$, and then (1) becomes $\mathcal{O}\left(\frac{1}{1-H(p)}(\log n + \log \delta^{-1})\right)$.

1.1 Our Contribution – Improved Query Complexity

In our analysis, we first focus on an *adversarial* model called *linearly bounded*, in which a rate of lies $r < 1/2$ is given at the beginning of the game and the Responder is restricted so that at most rt lies occur in a game of length t . It turns out that this model is easier to analyze and leads to the following theorem whose proof is postponed to Section 3.3.

► **Theorem 1.** *In the linearly bounded error model, with known error rate $r < 1/2$, the target can be found in at most $\frac{\log_2 n}{1-H(r)}$ vertex queries.*

This bound is strong enough to make an improvement in the probabilistic model. By a simple application of Chernoff bound, we get the following query complexity.

► **Theorem 2.** *In the probabilistic error model with error probability $p < 1/2$, the target can be found using at most*

$$\frac{1}{1-H(p)} \left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}) \right)$$

vertex queries, correctly with probability at least $1 - \delta$.

By an application of Young's inequality² and assuming that $p < 1/2$ is constant, we derive a query complexity of

$$\frac{\log_2 n}{1-H(p)} + o(\log n) + \mathcal{O}(\log \delta^{-1} \log \log \delta^{-1}).$$

Query complexity comparison

We compare, in the independent noise model, the precise query complexities of [19], i.e. (1) with Theorem 2. Observe that $\log n \cdot \frac{1}{C} + \log \delta^{-1} \cdot C^2 \geq 2\sqrt{\log n \log \delta^{-1}} \cdot \sqrt{C} \geq 2\sqrt{\log n \log \delta^{-1}}$ and that $\log \delta^{-1} \cdot C^2 \geq \log \delta^{-1}$, both holding since $C \geq 1$. Thus, our bound from Theorem 2 for all ranges of parameters asymptotically improves the one in (1).

Note that the compared bounds are with respect to worst-case strategy lengths. Our bounds can be made *in expectation* smaller by a factor of roughly $(1 - \delta)$ using the same techniques as in [5] and [19].

² $ab \leq \frac{a^p}{p} + \frac{b^q}{q}$ for $1/p + 1/q = 1$ and $a, b \geq 0$, from which follows that for $0 < A \leq B$ it holds $\sqrt{AB} = \mathcal{O}(A/\log A) + \mathcal{O}(B \log B)$. Thus, if $\log n \leq \log \delta^{-1}$, then we bound the term $\mathcal{O}(\sqrt{\log n \log \delta^{-1}})$ by $\mathcal{O}(\log n / \log \log n) + \mathcal{O}(\log \delta^{-1} \log \log \delta^{-1})$, and otherwise by the term $\mathcal{O}(\log \delta^{-1})$.

1.2 Our Contribution – Simplified Algorithmic Techniques

The crucial underlying idea behind the algorithm from [19] that reaches the query complexity in (1) is as follows. The algorithm maintains a weight function μ for the vertex set of the input graph $G = (V, E)$ so that, at any given time, $\mu(v)$ represents the likelihood that v is the target. Initially, all vertices have the same weight. For a given μ , define a *potential* of a vertex v to be $\Phi_\mu(v) = \sum_{u \in V} \mu(u)d(v, u)$, where $d(u, v)$ is the distance between the vertices u and v in G . A vertex q that minimizes this potential function is called a *weighted median*, or a *median* for short, $q = \arg \min_{v \in V} \Phi_\mu(v)$. The vertex to be queried in each iteration of the algorithm is a median (ties are broken arbitrarily). After each query, the weights are updated: the weight of each vertex that is compatible with the reply is multiplied by p , and the weights of the remaining vertices are multiplied by $1 - p$.

The above scheme for querying subsequent vertices is the main building block of the algorithm that reaches the query complexity in (1). However, the analysis of the algorithm reveals a problematic case, namely the vertices that account for at least half of the total weight, call them *heavy*. On one side, such vertices are good candidates to include the target, so they are ‘removed’ from the graph to be investigated later. However, the need to investigate them in this separate way leads to an algorithm that has three phases, where the first two end by trimming the graph by leaving only the heavy vertices for the next phase. The first two phases are sequences of vertex queries performed on a median. The last phase uses yet a different majority technique. The duration of each of the first two phases are dictated by complicated formulas, which makes the algorithm difficult to analyze and understand.

We propose a simpler algorithm than the one in [19]. In each step, we simply query a median until just one candidate target vertex remains. Our improvement lies in a refined analysis in how such a query technique updates the weights, which has several advantages. It not only leads to a better query complexity but also provides a much simpler proof. Moreover, it results in a better understanding as how querying a median works in general graphs. We point out that this technique is quite general: it can be successfully applied to other query models – the details can be found in the appendix.

1.3 Related Work

Regarding the problem of searching in graphs without errors, many papers have been devoted to trees, mainly because it is a structure that naturally generalizes paths, which represents the classical binary search (see e.g. [27] for search in a path with non-uniform query times). This query model in case of trees is equivalent to several other problems, including vertex ranking [15] or tree-depth [33]. There exist linear-time algorithms for finding optimal query strategies [34, 39]. A lot of effort has been done to understand the complexity for trees with non-uniform query times. It turns out that the problem becomes hard for trees [17, 16]. Also refer the reader to works on a closely related query game with edge queries [10, 11, 14, 28, 31]. For general graphs, a strategy that always queries a 1-median (the minimizer of the sum of distances over all vertices) has length at most $\log_2 n$ [19].

To shift our attention to searching in graphs with errors, two works have been recently published on probabilistic models [19, 18]. These models are further generalized in [12] by considering the case of identifying two targets t_1 and t_2 , where each answer to a query gives an edge on a shortest path to t_1 with probability p_1 or to t_2 with probability $p_2 = 1 - p_1$, respectively. Furthermore, there exists a closely related model in which the search is restricted in such a way, that each query performed to a vertex v must be followed by a vertex query to one of its neighbors – see [7, 21, 23, 22, 26] – in this context errors are usually referred to as unreliable advice.

An extensive amount of work has been devoted to searching problems in the presence of lies in a non-graph-theoretic context. The main tool of analysis is the concept of *volume* introduced by Berlekamp [6] – see also [9, 13] for a more detailed descriptions. We skip references to very numerous works that deal with fixed number of lies, pointing to surveys in [9, 13, 35]. For general queries, it is known [37] that a strategy of length $\log n + L \log_2 \log_2 n + \mathcal{O}(L \log L)$ exists, where n is the size of the search space and L is an upper bound on the number of lies. An almost optimal approximation strategy can be found in [32], which is actually given for a more general model of q -ary queries. For the most relevant model in our context, the probabilistic model, we remark on the early works, which bound strategy lengths to $\mathcal{O}(\frac{1}{\text{poly}(\varepsilon)} \log n \log \delta^{-1})$, where $p < \frac{1}{2}$ and $\varepsilon = \frac{1}{2} - p$, with confidence probability $1 - \delta$ [2, 8]. A strategy of length $\mathcal{O}(\varepsilon^{-2}(\log n + \log \delta^{-1}))$ is given in [20]. Finally, [5] gives the best known bound of $\frac{1}{1-H(p)}(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$. We note that we arrive at a strategy matching asymptotically the complexity of [20] as a by-product from our graph-theoretic analysis (presented in the appendix).

2 Preliminaries

We now introduce the notation regarding the dynamics of the game. We assume an input graph with non-uniform edge lengths, and we denote said lengths by $\omega(e)$. We denote by $d(u, v)$ the *distance* between two vertices u and v , which is the length of a shortest path in G between u and v . We first focus on a simplified error model where the Responder is allowed a fixed number of lies, with the upper bound denoted as L . During the game, the Questioner keeps track of a *lie counter* ℓ_v for each vertex v of G . The value of ℓ_v equals the number of lies that must have already occurred assuming that v is actually the target v^* . The Questioner will utilize a constant $\Gamma > 1$ that will be fixed later. The goal of having this parameter is that we can tune it in order to obtain the right asymptotics. We define a *weight* $\mu_t(v)$ of a vertex v at the end of a round $t > 0$:

$$\mu_t(v) = \mu_0(v) \cdot \Gamma^{-\ell_v},$$

where $\mu_0(v)$ is the initial weight of v . For subsets $U \subseteq V$, let $\mu(U) = \sum_{v \in U} \mu(v)$. For brevity we write μ_t in place of $\mu_t(V)$. For a queried vertex q and an answer v , a vertex u is *compatible* with the answer if $u = v$ when $q = v$, or v lies on a shortest path from q to u .

As soon as there is only one vertex v left with $\ell_v \leq L$, the Questioner can successfully detect the target, $v^* = v$. We will set the initial weight of each vertex v to be $\mu_0(v) = 1$. Thus, $\mu_0 = n$ and $\mu_T \geq \Gamma^{-L}$ if the strategy length is T .

Based on the weight function μ , we define a *potential* of a vertex v :

$$\Phi(v) = \sum_{u \in V} \mu(u) \cdot d(v, u).$$

We write $\Phi_t(v)$ to refer to the value of a potential at the end of round t . Any vertex $x \in V$ minimizing $\Phi(x)$ is called *1-median*.

Denote for an edge $\{v, u\}$, $N(v, u) = \{x \mid d(u, x) + \omega(\{v, u\}) = d(v, x)\}$ to be the set of all vertices to which some shortest path from v leads through u . Thus, $N(v, u)$ consists of the compatible vertices for the answer u when v has been queried. For any $S \subseteq V$, we write for brevity $\bar{S} = V \setminus S$, and for singletons $\{\bar{v}\}$ we further shorten to \bar{v} . We say that a vertex v is α -*heavy*, for some $0 \leq \alpha \leq 1$, if $\mu(v) > \alpha \cdot \mu(V)$. For a queried vertex q , if the answer is q , then such a reply is called a *yes-answer*; otherwise it is called a *no-answer*.

Algorithm VERTEX: Vertex queries for a fixed number of L lies.

```

1 for  $v \in V$  do
2    $\mu(v) = 1$ 
3    $\ell_v = 0$ 
4 while more than one vertex  $x \in V$  has  $\ell_x \leq L$  do
5    $q = \arg \min_{x \in V} \Phi(x)$ 
6   query the vertex  $q$ 
7   for all nodes  $u$  not compatible with the answer do
8      $\ell_u = \ell_u + 1$ 
9      $\mu(u) = \mu(u)/\Gamma$ 
10 return the only  $x$  such that  $\ell_x \leq L$ 

```

3 Vertex Searching

We now formally state the search strategy for a fixed number of lies – see Algorithm VERTEX. We combine our weight together with the idea of querying a 1-median [19]. As announced earlier, it turns out that our bound together with an appropriately selected weight function are strong enough so that we do not need the additional stages enhanced with a majority selection used in [19] in order to gain asymptotic improvements. We also note that we can easily introduce technical modifications to this strategy by changing the initial weight, the value of Γ or the stopping condition. We will do this to conclude several results for various error models (see the appendix).

3.1 Analysis of the Strategy

In this subsection we prove the following main technical contribution.

► **Theorem 3.** *Algorithm VERTEX finds the target in at most*

$$\frac{1}{\log_2(2\Gamma/(\Gamma+1))} \log_2 n + \frac{\log_2 \Gamma}{\log_2(2\Gamma/(\Gamma+1))} \cdot L$$

vertex queries.

Note that, due to the values of the initial and the final weight, it is enough to argue that the weight decreases on average, i.e., in an amortized way, by a factor of $(\Gamma+1)/(2\Gamma)$ per round. We first handle two cases (see Lemmas 4 and 5) when the weight decreases appropriately after a single query. These cases are a no-answer, and a yes-answer but only when the queried vertex is not 1/2-heavy. In the remaining case, i.e., when the queried vertex q is 1/2 heavy, it is not necessarily true that the weight decreases by the desired factor – this particularly happens in case of a yes-answer to such a query. This case is handled by the amortized analysis: we pair such yes-answers with no-answers to the query on q and show that in each such pair the weight decreases appropriately.

► **Lemma 4.** *If Algorithm VERTEX receives a no-answer in a round $t+1$, then $\mu_{t+1} \leq \frac{\Gamma+1}{2\Gamma} \mu_t$.*

Proof. Let q be the vertex queried in round $t+1$. Assume that the reply is some neighbor v of q . By [19], Lemma 4, we get that $\frac{\mu_t(N(q,v))}{|N(q,v)|} \leq \mu_t/2$. Moreover, because the lie counter increases by one for all vertices in $\overline{N(q,v)}$ and does not change for all vertices in $N(q,v)$ in round $t+1$, it follows that $\mu_{t+1} = \mu_t(N(q,v)) + \frac{1}{\Gamma} \mu_t(\overline{N(q,v)}) \leq \frac{\Gamma+1}{2\Gamma} \mu_t$. ◀

► **Lemma 5.** *Suppose that Algorithm VERTEX queries in round $t + 1$ a vertex q that is not 1/2-heavy. If a yes-answer is received, then $\mu_{t+1} \leq \frac{\Gamma+1}{2\Gamma} \mu_t$.*

Proof. The lie counter increments for each vertex of G except for q and remains the same for q in round $t + 1$: $\mu_{t+1}(q) = \mu_t(q)$ and $\mu_{t+1}(\bar{q}) = \frac{1}{\Gamma} \mu_t(\bar{q})$. Since q is not 1/2-heavy at the beginning of round $t + 1$, $\mu_t(q) \leq \mu_t/2$. Thus, we get $\mu_{t+1} = \mu_t(q) + \frac{1}{\Gamma} \mu_t(\bar{q}) \leq \frac{\Gamma+1}{2\Gamma} \mu_t$. ◀

Now we turn to the proof of Theorem 3. Consider a maximal interval $[t_1, t_2]$, where $t_1 \leq t_2$ are integers, such that there exists a vertex q that is 1/2-heavy in each round t_1, \dots, t_2 , and q is not 1/2-heavy in round $t_2 + 1$. Call it a q -interval. Note that $t_1 > 0$ and q is not 1/2-heavy in round $t_1 - 1$. We permute the replies given by the Responder in the q -interval to obtain a new sequence of replies as follows. The replies in rounds $1, \dots, t_1 - 1$ and $t_2 + 1$ onwards are the same in both sequences. Note that in the interval $[t_1, t_2]$ the number of yes-answers, denote it by p , is smaller than or equal to the number of no-answers. Reorder the replies in the q -interval so that the yes-answers occur in rounds $t_1 + 2i$ for each $i \in \{0, \dots, p - 1\}$. In other words, we pair the yes-answers with no-answers so that a yes-answer in round $t_1 + 2i$ is paired with a no-answer in round $t_1 + 2i + 1$; we call such two rounds a *pair*. Following the pairs, some remaining, if any, no-answers follow in rounds $t_1 + 2p, \dots, t_2$. Perform this transformation as long as a q -interval exists for some $q \in V$. Denote by μ' the weight of the new sequence.

Denote by t' , if it exists, the minimum integer such that for some vertex v and for each $t > t'$, v is 1/2-heavy at the end of the round t . If no such t' exists, then let t' be defined to be the number of rounds of the strategy.

We first analyze what happens, in the new sequence, in rounds i and $i + 1$ that are a pair in an arbitrary q -interval for some vertex q . After such two rounds the lie counter for q increases by one, and the lie counter of any other vertex increases by at least one. This in particular implies that q is a 1-median throughout the entire q -interval in the new sequence. Moreover, the two replies in these rounds result in weight decrease by a factor of at least Γ , $\mu'_{i+1} \leq \mu'_{i-1}/\Gamma$. Since $\frac{1}{\Gamma} < (\frac{1+\Gamma}{2\Gamma})^2$, the overall progress after the pair is as required.

We now prove that for each $t \in \{0, \dots, t' - 1\}$ that does not belong to any pair it holds

$$\mu'_{t+1} \leq \frac{\Gamma + 1}{2\Gamma} \mu'_t. \quad (2)$$

Recall that for each $t \leq t'$ that does not belong to any q -interval, $\mu'_t(v) = \mu_t(v)$ for each $v \in V$. If the answer to this query is a no-answer, then (2) follows from Lemma 4. Lemma 4 also applies to no-answers of a q -interval that do not belong to any pair since, as argued above, q is a 1-median throughout the q -interval. If the answer is a yes-answer, then since the queried vertex q is not 1/2-heavy due to the choice of q -intervals, Inequality (2) follows from Lemma 5.

If t' is the last round in the original search strategy, then the proof is completed. Otherwise, consider the suffix of the original sequence of replies, consisting of rounds t for $t > t'$. In all these rounds, by definition, some vertex q is 1/2-heavy. Also by definition, both sequences μ and μ' are identical in this suffix. One can check that if a vertex is heavy at the end of some round, then in the subsequent round Algorithm VERTEX does query this vertex. Thus, the vertex q is queried in all rounds of the suffix, and hence q is the target. Thus, it is enough to observe how the weight decreases on \bar{q} in case of a yes-answer in a round $t > t'$: $\mu'_t(\bar{q}) = \mu'_{t-1}(\bar{q})/\Gamma \leq \frac{\Gamma+1}{2\Gamma} \mu'_{t-1}(\bar{q})$. This completes the proof of Theorem 3.

3.2 Proof of Theorem 1

Proof. We turn our attention to the model with a rate of lies bounded by a fraction $r < 1/2$ (linearly bounded error model). Our result, Theorem 1, is obtained on the basis of Algorithm VERTEX and the precise bound from Theorem 3. In particular, we run Algorithm VERTEX with $\Gamma = \frac{1-r}{r}$ and with a fixed bound on number of lies $L = \frac{\log_2 n}{1-H(r)}r$. By Theorem 3, Algorithm VERTEX asks then at most $\frac{\log_2 n}{\log_2(2 \cdot (1-r))} + \frac{\log_2 \frac{1-r}{r}}{\log_2(2 \cdot (1-r))} \cdot L = \frac{\log_2 n}{1-H(r)} \cdot \frac{1-H(r)+r \log_2 \frac{1-r}{r}}{1+\log_2(1-r)} = \frac{\log_2 n}{1-H(r)} = L/r$ queries. This bound concludes the proof, since the number of lies is within r fraction of strategy length. ◀

3.3 Proof of Theorem 2

Proof. Let $\varepsilon > 0$ be such that $p = \frac{1}{2}(1 - \varepsilon)$. We run the strategy from Theorem 1 with an error rate $r = \frac{1}{2}(1 - \varepsilon_0)$, where $\varepsilon_0 = \varepsilon / \left(1 + \sqrt{8 \ln \delta^{-1} / \ln n}\right)$. By Theorem 1 the strategy length is $Q = \frac{\log_2 n}{1-H(r)}$ which is (up to lower-order terms) $2\varepsilon_0^{-2} \ln n$, thus at least $\varepsilon_0^{-2} \ln n$ for n large enough. The expected number of lies is $\mathbb{E}[L] = p \cdot Q$ and by the Chernoff bound,

$$\begin{aligned} \Pr[Q - L \leq (1-r) \cdot Q] &\leq \exp\left(-\frac{1}{2} \left(1 - \frac{1-r}{1-p}\right)^2 \cdot (1-p) \cdot \frac{\ln n}{\varepsilon_0^2}\right) \\ &\leq \exp\left(-\frac{1}{8} \left(\frac{\varepsilon - \varepsilon_0}{\varepsilon_0}\right)^2 \ln n\right) = \delta. \end{aligned}$$

The bound $Q = \frac{\log_2 n}{1-H(p)}(1 + \mathcal{O}(\sqrt{\ln \delta^{-1} / \ln n}) + \mathcal{O}(\ln \delta^{-1} / \ln n))$ follows then from $1 - H(x) \sim (1/2 - x)^2$. ◀

4 Conclusions

We note that also other query models have been studied in the graph-theoretic context, including edge queries. In an *edge query*, the Questioner points to an edge and the Responder tells which endpoint of that edge is closer to the target, breaking ties arbitrarily. It turns out that edge queries are more challenging to analyze, i.e., our technique for vertex queries does not transfer without changes. This is mostly due to a possible lack of edges that subdivide the search space equally enough. This issue can be patched by treating heavy vertices in a separate way. We provide a strategy of query complexity $\mathcal{O}(\frac{1}{\varepsilon^2} \Delta \log \Delta (\log n + \log \delta^{-1}))$. This generalizes the noisy binary search of [20] to general graphs, and has the advantage of being a weight-based strategy.

We additionally show the generalizations of our strategies to searching in unbounded domains, where one is concerned in searching e.g., the space of all positive integers with comparison queries. The goal is to minimize the number of queries as a function of N , the (unknown) position of the target. By adjusting the initial distribution of the weight to decay at polynomial rate with respect to the distance from the point 0, we almost automatically get desired solutions, e.g., a strategy of query complexity $\mathcal{O}(\frac{1}{\varepsilon^2} (\log N + \log \delta^{-1}))$ for searching in the probabilistic error model, improving upon $\mathcal{O}(\text{poly}(\varepsilon^{-1}) \log N \log \delta^{-1})$ of [2].

References

- 1 Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1987. doi:10.1007/BF00116828.
- 2 Javed A Aslam. *Noise tolerant algorithms for learning and searching*. PhD thesis, Massachusetts Institute of Technology, 1995.
- 3 Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 18:3:1–3:35, 2017. URL: <http://jmlr.org/papers/v18/15-085.html>.
- 4 Maria-Florina Balcan and Avrim Blum. Clustering with Interactive Feedback. In *Algorithmic Learning Theory, 19th International Conference, ALT 2008, Budapest, Hungary, October 13-16, 2008. Proceedings*, pages 316–328, 2008. doi:10.1007/978-3-540-87987-9_27.
- 5 Michael Ben-Or and Avinatan Hassidim. The Bayesian Learner is Optimal for Noisy Binary Search (and Pretty Good for Quantum as Well). In *FOCS*, pages 221–230, 2008. doi:10.1109/FOCS.2008.58.
- 6 Elvyn R. Berlekamp. *Block Coding For The Binary Symmetric Channel With Noiseless, Delayless Feedback*, pages 61–88. Wiley & Sons, New York, 1968.
- 7 Lucas Boczkowski, Amos Korman, and Yoav Rodeh. Searching a Tree with Permanently Noisy Advice. *CoRR*, abs/1611.01403, 2016. arXiv:1611.01403.
- 8 Ryan S. Borgstrom and S. Rao Kosaraju. Comparison-based search in the presence of errors. In *STOC*, pages 130–136, 1993. doi:10.1145/167088.167129.
- 9 Ferdinando Cicalese. *Fault-Tolerant Search Algorithms: Reliable Computation with Unreliable Information*. Springer Publishing Company, Incorporated, 2013.
- 10 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. doi:10.1016/j.tcs.2012.06.023.
- 11 Ferdinando Cicalese, Balázs Keszegh, Bernard Lidický, Dömötör Pálvölgyi, and Tomás Valla. On the tree search problem with non-uniform costs. *Theor. Comput. Sci.*, 647:22–32, 2016. doi:10.1016/j.tcs.2016.07.019.
- 12 Argyrios Deligkas, George B. Mertzios, and Paul G. Spirakis. Binary Search in Graphs Revisited. In *MFCS*, pages 20:1–20:14, 2017. doi:10.4230/LIPIcs.MFCS.2017.20.
- 13 Christian Deppe. *Coding with Feedback and Searching with Lies*, pages 27–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-32777-6_2.
- 14 Dariusz Dereniowski. Edge ranking of weighted trees. *Discrete Applied Mathematics*, 154(8):1198–1209, 2006. doi:10.1016/j.dam.2005.11.005.
- 15 Dariusz Dereniowski. Edge ranking and searching in partial orders. *Discrete Applied Mathematics*, 156(13):2493–2500, 2008. doi:10.1016/j.dam.2008.03.007.
- 16 Dariusz Dereniowski, Adrian Kosowski, Przemyslaw Uznański, and Mengchuan Zou. Approximation Strategies for Generalized Binary Search in Weighted Trees. In *ICALP*, pages 84:1–84:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.84.
- 17 Dariusz Dereniowski and Adam Nadolski. Vertex rankings of chordal graphs and weighted trees. *Inf. Process. Lett.*, 98(3):96–100, 2006. doi:10.1016/j.ipl.2005.12.006.
- 18 Ehsan Emamjomeh-Zadeh and David Kempe. A General Framework for Robust Interactive Learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 7085–7094, 2017. URL: <http://papers.nips.cc/paper/7283-a-general-framework-for-robust-interactive-learning>.
- 19 Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *STOC*, pages 519–532, 2016. doi:10.1145/2897518.2897656.

- 20 Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with Noisy Information. *SIAM J. Comput.*, 23(5):1001–1018, 1994. doi:10.1137/S0097539791195877.
- 21 Nicolas Hanusse, David Ilcinkas, Adrian Kosowski, and Nicolas Nisse. Locating a target with an agent guided by unreliable local advice: how to beat the random walk when you have a clock? In *PODC*, pages 355–364, 2010. doi:10.1145/1835698.1835781.
- 22 Nicolas Hanusse, Dimitris J. Kavadias, Evangelos Kranakis, and Danny Krizanc. Memoryless search algorithms in a network with faulty advice. *Theor. Comput. Sci.*, 402(2-3):190–198, 2008. doi:10.1016/j.tcs.2008.04.034.
- 23 Nicolas Hanusse, Evangelos Kranakis, and Danny Krizanc. Searching with mobile agents in networks with liars. *Discrete Applied Mathematics*, 137(1):69–85, 2004. doi:10.1016/S0166-218X(03)00189-6.
- 24 Richard M. Karp and Robert Kleinberg. Noisy binary search and its applications. In *SODA*, pages 881–890, 2007. URL: <http://dl.acm.org/citation.cfm?id=1283383.1283478>.
- 25 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. URL: <https://mitpress.mit.edu/books/introduction-computational-learning-theory>.
- 26 Evangelos Kranakis and Danny Krizanc. Searching with Uncertainty. In *SIROCCO'99, 6th International Colloquium on Structural Information & Communication Complexity, Lacanau-Ocean, France, 1-3 July, 1999*, pages 194–203, 1999.
- 27 Eduardo Sany Laber, Ruy Luiz Milidiú, and Artur Alves Pessoa. On binary searching with non-uniform costs. In *SODA*, pages 855–864, 2001. URL: <http://dl.acm.org/citation.cfm?id=365411.365796>.
- 28 Tak Wah Lam and Fung Ling Yue. Optimal Edge Ranking of Trees in Linear Time. *Algorithmica*, 30(1):12–33, 2001. doi:10.1007/s004530010076.
- 29 Nick Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning*, 2(4):285–318, 1987. doi:10.1007/BF00116827.
- 30 Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer, 2011. doi:10.1007/978-3-642-14267-3.
- 31 Shay Mozes, Krzysztof Onak, and Oren Weimann. Finding an optimal tree searching strategy in linear time. In *SODA*, pages 1096–1105, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347202>.
- 32 S. Muthukrishnan. On Optimal Strategies for Searching in Presence of Errors. In *SODA*, pages 680–689, 1994. URL: <http://dl.acm.org/citation.cfm?id=314464.314672>.
- 33 Jaroslav Nesetril and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/j.ejc.2005.01.010.
- 34 Krzysztof Onak and Pawel Parys. Generalization of Binary Search: Searching in Trees and Forest-Like Partial Orders. In *F0CS*, pages 379–388, 2006. doi:10.1109/F0CS.2006.32.
- 35 Andrzej Pelc. Searching games with errors—fifty years of coping with liars. *Theoretical Computer Science*, 270(1):71–109, 2002. doi:10.1016/S0304-3975(01)00303-6.
- 36 Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 239–248, 2005. doi:10.1145/1081870.1081899.
- 37 Ronald L. Rivest, Albert R. Meyer, Daniel J. Kleitman, Karl Winklmann, and Joel Spencer. Coping with errors in binary search procedures. *Journal of Computer and System Sciences*, 20(3):396–404, 1980.
- 38 Alfréd Rényi. On a problem of information theory. *MTA Mat. Kut. Int. Kozl.*, 6B:505–516, 1961.

- 39 Alejandro A. Schäffer. Optimal Node Ranking of Trees in Linear Time. *Inf. Process. Lett.*, 33(2):91–96, 1989. doi:10.1016/0020-0190(89)90161-0.
- 40 Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012. doi:10.2200/S00429ED1V01Y201207AIM018.
- 41 Stanislaw M. Ulam. *Adventures of a Mathematician*. Scribner, New York, 1976.

A Analysis of the Generic Strategies for Edge Queries

We recall a different format of queries called *edge queries*, where in each round the Questioner selects an edge $\{u, v\}$ of an input graph and the Responder replies with the endpoint of $\{u, v\}$ that is closer to the target. Again, ties are broken arbitrarily. The edge-query model naturally generalizes comparison queries in linearly or partially ordered data. In case of edge-queries we consider graphs with unit edge lengths.

We start by giving the notation regarding edge queries. The *degree* of a vertex v , denoted by $\deg(v)$, is the number of its neighbors in G . We denote by $\Delta = \max_{v \in V} \deg(v)$ the maximum degree of G . We define an edge-vertex distance $d(e, v) = \min(d(x, v), d(y, v))$ for an edge $e = \{x, y\}$. Similarly as for vertex queries, based on a weight function μ and distance d , we define a potential of an edge e :

$$\Phi(e) = \sum_{u \in V} \mu(u) \cdot d(e, u).$$

Again, we write $\Phi_t(e)$ to refer to this value at the end of round t . Any edge e minimizing $\Phi(e)$ is called *1-edge-median*. For an edge $e = \{u, v\}$ and one of its endpoints,

$$N(e, v) = \{w \mid d(v, w) \leq d(u, w)\}, \quad N_{<}(e, v) = \{w \mid d(v, w) < d(u, w)\}.$$

For edge-queries we give a strategy that is a bit more complicated – see Algorithm EDGE. Intuitively, as opposed to the vertex-query case, there may be no edges in the graph that ‘subdivide’ the search space evenly enough. This already happens as soon as one of the vertices is $\frac{1}{\Delta+1}$ -heavy. If this is the case, and say vertex v is $\frac{1}{\Delta+1}$ -heavy, we cyclically query edges incident to v in an appropriate greedy order. We continue to do so until all other vertices have been eliminated, and hence v must be the target, or v is no longer $\frac{1}{\Delta+1}$ -heavy. If none of the vertices is $\frac{1}{\Delta+1}$ -heavy, we simply query a 1-edge-median. The absence of such heavy vertices essentially ensures, that this decreases the weight sufficiently.

This results in a more involved proof given in Section 3.1. Similarly as for vertex queries, we also first provide an analysis for a fixed number of lies (see Theorem 6) and then from this bound we derive appropriate bounds for other models (Theorems 7 and 8).

► **Theorem 6.** *Let $\Gamma > 1$. Algorithm EDGE finds the target in at most $\frac{\log n + L \log \Gamma}{\log(1 + \frac{\Gamma-1}{\Gamma\Delta+1})}$ edge queries.*

► **Theorem 7.** *In the linearly bounded error model with error rate $r = \frac{1}{\Delta+1}(1 - \varepsilon)$ for some $0 < \varepsilon \leq 1$, the target can be found in at most $2\varepsilon^{-2}\Delta \ln n$ edge queries.*

► **Theorem 8.** *In the probabilistic error model with error rate $p = \frac{1}{2}(1 - \varepsilon)$ for some $0 < \varepsilon \leq 1$ there exists a strategy that finds the target using at most $\mathcal{O}(\varepsilon^{-2}\Delta \log \Delta \cdot (\log n + \log \delta^{-1}))$ edge queries, correctly with probability at least $1 - \delta$.*

Algorithm EDGE: Edge queries for fixed number of L lies.

```

1 for  $v \in V$  do
2    $\mu(v) = 1$ 
3    $\ell_v = 0$ 
4 while more than one vertex  $x$  satisfies  $\ell_x \leq L$  do
5   if there exists  $v$  such that  $\mu(v) > \mu/(\Delta + 1)$  then ▷  $v$  is  $\frac{1}{\Delta+1}$ -heavy
6     for  $i = 1$  to  $\deg(v)$  do ▷ greedy ordering of neighbors
7       select an edge  $e_i$  incident to  $v$  to maximize  $\mu(\bigcup_{j \leq i} N_{<}(e_j, v))$ 
8        $i = 1$ 
9       do ▷ cyclically query edges incident to  $v$ 
10        query  $e_i$ 
11        for all nodes  $u$  not compatible with the answer do
12           $\ell_u = \ell_u + 1$ 
13           $\mu(u) = \mu(u)/\Gamma$ 
14        if the answer to the last query is  $v$  then
15           $i = (i + 1) \bmod \deg(v)$ 
16        while  $\mu(v) > \mu/(\Delta + 1)$  and there exists more than one  $x$  with  $\ell_x \leq L$ 
17      else
18         $e = \arg \min_{x \in E} \Phi(x)$ 
19        query  $e$ 
20        for all nodes  $u$  not compatible with the answer do
21           $\ell_u = \ell_u + 1$ 
22           $\mu(u) = \mu(u)/\Gamma$ 
23 return  $v$  such that  $\ell_v \leq L$ 

```

Proof of Theorem 6

We first prove two technical lemmas and then we give the proof of the theorem.

► **Lemma 9.** *Let $\Gamma > 1$. Suppose that Algorithm EDGE queries in round $t + 1$ an edge e_q incident to a vertex q such that $e_q = \arg \min_{x \in E} \Phi_t(x)$. If $\deg(q) > 1$, then*

$$\mu_t(\overline{N(e_q, q)}) \geq \frac{1}{\deg(q)} (\mu_t - \mu_t(q)). \quad (3)$$

Proof. Denote $e_q = \{q, v\}$. For each neighbor w of q define

$$N_w^\cap = N(e_q, q) \cap N_{<}(\{q, w\}, w).$$

Consider an edge $e' = \{q, w\}$ that maximizes $\mu_t(N_w^\cap)$. If X is the set of neighbors of q , then by definition and by the fact that e_q lies on no shortest path from q to any vertex in $N_{<}(e_q, v)$, i.e., $N_v^\cap = \emptyset$, it holds

$$N(e_q, q) \setminus \{q\} \subseteq \bigcup_{w' \in X} N_{w'}^\cap = \bigcup_{w' \in X \setminus \{v\}} N_{w'}^\cap.$$

Hence (since e' maximizes $\mu_t(N_w^\cap)$) we obtain that

$$\mu_t(N_w^\cap) \geq \frac{1}{\deg(q) - 1} (\mu_t(N(e_q, q)) - \mu_t(q)). \quad (4)$$

For brevity, we extend our notation in the following way: for an edge e and a subset S of vertices, $\Phi_t(e, S) = \sum_{z \in S} \mu_t(z) \cdot d(e, z)$. Note that for any $S \subseteq V$ and any edge e , $\Phi_t(e) = \Phi_t(e, S) + \Phi_t(e, \overline{S})$. We obtain

$$\begin{aligned}
\Phi_t(e', N(e_q, q)) &= \Phi_t(e', N_w^\cap) + \Phi_t(e', N(e_q, q) \setminus N_w^\cap) \\
&= \sum_{u \in N_w^\cap} \mu_t(u) \cdot (d(q, u) - 1) + \sum_{u \in N(e_q, q) \setminus N_w^\cap} \mu_t(u) \cdot d(q, u) \\
&= \sum_{u \in N(e_q, q)} \mu_t(u) \cdot d(q, u) - \mu_t(N_w^\cap) \\
&\leq \Phi_t(e_q, N(e_q, q)) - \frac{1}{\deg(q) - 1} (\mu_t(N(e_q, q)) - \mu_t(q)), \tag{5}
\end{aligned}$$

where the last inequality is due to (4). For any vertex u , $d(e', u) \leq d(e_q, u) + 1$ because e_q and e' are adjacent. Using this fact we obtain:

$$\begin{aligned}
\Phi_t(e', \overline{N(e_q, q)}) &= \sum_{u \notin N(e_q, q)} \mu_t(u) \cdot d(e', u) \\
&\leq \sum_{u \notin N(e_q, q)} \mu_t(u) \cdot d(e_q, u) + \sum_{u \notin N(e_q, q)} \mu_t(u) \\
&= \Phi_t(e_q, \overline{N(e_q, q)}) + \mu_t(\overline{N(e_q, q)}). \tag{6}
\end{aligned}$$

Finally, by (5) and (6) we get:

$$\begin{aligned}
\Phi_t(e') &= \Phi_t(e', N(e_q, q)) + \Phi_t(e', \overline{N(e_q, q)}) \\
&\leq \Phi_t(e_q, N(e_q, q)) - \frac{\mu_t(N(e_q, q)) - \mu_t(q)}{\deg(q) - 1} + \Phi_t(e_q, \overline{N(e_q, q)}) + \mu_t(\overline{N(e_q, q)}) \\
&= \Phi_t(e_q) + \mu_t(\overline{N(e_q, q)}) - \frac{1}{\deg(q) - 1} (\mu_t(N(e_q, q)) - \mu_t(q)).
\end{aligned}$$

By assumption, $\Phi_t(e_q) \leq \Phi_t(e')$. Therefore,

$$\frac{1}{\deg(q) - 1} (\mu_t(N(e_q, q)) - \mu_t(q)) \leq \mu_t(\overline{N(e_q, q)}),$$

which can be rewritten as in (3). ◀

► **Lemma 10.** *Let $\Gamma > 1$. Suppose that Algorithm EDGE queries in round $t + 1$ an edge incident to a vertex q that is not $\frac{1}{\Delta+1}$ -heavy in this round, and the answer is q . Then, $\mu_{t+1} \leq (1 - \frac{\Gamma-1}{\Gamma(\Delta+1)})\mu_t$.*

Proof. Let $e_q = \{q, v\}$ be the edge queried in round $t + 1$. Suppose first that $\deg(q) > 1$. By Lemma 9,

$$\mu_t(\overline{N(e_q, q)}) \geq \frac{1}{\deg(q)} (\mu_t - \mu_t(q)) \geq \frac{1}{\Delta} (\mu_t - \mu_t(q)). \tag{7}$$

Because e_q is the queried edge in round $t + 1$ and the reply is q , the lie counter remains unchanged for the vertices in $N(e_q, q)$ and decreases by one in the complement $\overline{N(e_q, q)}$. Hence,

$$\mu_{t+1} = \mu_t(N(e_q, q)) + \frac{1}{\Gamma} \mu_t(\overline{N(e_q, q)}) = \mu_t - \frac{\Gamma - 1}{\Gamma} \mu_t(\overline{N(e_q, q)}).$$

4:14 A Framework for Searching in Graphs in the Presence of Errors

Thus, by (7) and by the fact that $\mu_t(q) \leq \frac{1}{\Delta+1}\mu_t$ for q that is not $\frac{1}{\Delta+1}$ -heavy in round t ,

$$\mu_{t+1} \leq \left(1 - \frac{\Gamma-1}{\Gamma\Delta} \cdot \frac{\Delta}{\Delta+1}\right) \mu_t,$$

which completes the proof in the case when $\deg(q) > 1$.

If $\deg(q) = 1$, then in round t the lie counter increases by one for each vertex in \bar{q} . Thus, again by the fact that q is not $\frac{1}{\Delta+1}$ -heavy,

$$\mu_{t+1} = \mu_t(q) + \frac{1}{\Gamma}\mu_t(\bar{q}) \leq \left(\frac{1}{\Delta+1} + \frac{1}{\Gamma}\right) \mu_t \leq \left(1 - \frac{\Gamma-1}{\Gamma(\Delta+1)}\right) \mu_t. \quad \blacktriangleleft$$

Proof of Theorem 6. Having proved the technical lemmas, we now turn to the proof of Theorem 6. It is enough to argue that every query, amortized, multiplies the weight by a factor of $1 - \frac{\Gamma-1}{\Gamma(\Delta+1)} = 1/(1 + \frac{\Gamma-1}{\Gamma\Delta+1})$. If there is no $\frac{1}{\Delta+1}$ -heavy vertex, then the theorem follows from Lemma 10. Hence suppose in the rest of the proof that there exists a $\frac{1}{\Delta+1}$ -heavy vertex and denote this vertex by q .

For the amortized analysis, consider a sequence of t consecutive queries to edges e_1, \dots, e_t , $t \leq \deg(q)$, performed while q is $\frac{1}{\Delta+1}$ -heavy; call such a sequence a *segment*. Suppose this sequence starts in round t' . Denote $e_i = \{q, v_i\}$, $i \in \{1, \dots, t\}$, and let

$$Q_1 = \bigcup_{i=1}^t N_{<}(e_i, v_i), \quad Q_2 = V \setminus (Q_1 \cup \{q\}).$$

First we assume that the query in round $t' + t$ (i.e., the query that follows the sequence) does not return v as a reply, or v stops being $\frac{1}{\Delta+1}$ -heavy. We argue, informally speaking, that this query in round $t' + t$ amortizes the t queries prior to it thanks to the assumption $t \leq \deg(q)$. Because the lie counter of q increments in round $t' + t$,

$$\mu_{t'+t}(q) \leq \frac{1}{\Gamma}\mu_{t'}(q). \quad (8)$$

We have $\mu_{t'+t}(Q_1) \leq \frac{1}{\Gamma}\mu_{t'}(Q_1)$ by the formulation of Algorithm EDGE, and $\mu_{t'+t}(Q_2) \leq \mu_{t'}(Q_2)$. Then, $Q_1 \cup Q_2 = \bar{q}$ and $Q_1 \cap Q_2 = \emptyset$ imply $\mu_{t'}(Q_1) \leq \mu_{t'}(\bar{q}) - \mu_{t'}(Q_2)$ and hence

$$\mu_{t'+t}(Q_1) + \mu_{t'+t}(Q_2) \leq \frac{1}{\Gamma}\mu_{t'}(\bar{q}) + \frac{\Gamma-1}{\Gamma}\mu_{t'}(Q_2). \quad (9)$$

Due to the order according to which the edges $\{q, v_i\}$ are queried, we have

$$\mu_{t'}(Q_2) \leq \left(1 - \frac{t}{\deg(q)}\right) \mu_{t'}(\bar{q}) \leq \left(1 - \frac{t}{\Delta}\right) \mu_{t'}(\bar{q}). \quad (10)$$

Note that $\mu_{t'}(\bar{q}) \leq \frac{\Delta}{\Delta+1}\mu_{t'}$ since by assumption q is $\frac{1}{\Delta+1}$ -heavy in round t' . Since $\mu_{t'+t} = \mu_{t'+t}(q) + \mu_{t'+t}(Q_1) + \mu_{t'+t}(Q_2)$, we get by (8), (9) and (10):

$$\mu_{t'+t} \leq \left(\frac{1}{\Gamma} + \frac{\Gamma-1}{\Gamma} \frac{\Delta-t}{\Delta+1}\right) \mu_{t'} = \left(1 - \frac{\Gamma-1}{\Gamma} \frac{t+1}{\Delta+1}\right) \mu_{t'} \leq \left(1 - \frac{\Gamma-1}{\Gamma(\Delta+1)}\right)^{t+1} \mu_{t'},$$

where the last inequality comes from $(1-x)^k \geq 1-xk$, for $k \geq 1$ and $x < 1$.

Consider now a maximal sequence S of rounds in which q is $\frac{1}{\Delta+1}$ -heavy and is not $\frac{1}{\Delta+1}$ -heavy in the round that follows the sequence. Note that Algorithm EDGE cyclically queries the edges incident to q in S . Let $r'_1 \leq \dots \leq r'_b$ be all rounds in S having q as an

answer. Denote $X = S \setminus \{r'_1, \dots, r'_{b'}\}$, the set of rounds in S in which q is not an answer. Let $a = \lceil b' / \deg(q) \rceil$. The lie counter of each vertex in \bar{q} increases by at least $a - 1$ and by at most a times by executing S – we point out that this crucial property follows from the fact that the queries in the segment are applied to the edges incident to q consecutively modulo $\deg(v)$. Since q is $\frac{1}{\Delta+1}$ -heavy at the beginning of S and is not $\frac{1}{\Delta+1}$ -heavy right after S , the lie counter of q increases by at least a as a result of S . Hence, $|X| \geq a$. Partition $r'_1, \dots, r'_{b'}$ into a minimum number of segments of length at most $\deg(q)$ each, which leads to at most a segments. Thus, we can pair these segments with rounds in X . For each such pair of at most $\deg(q) + 1$ rounds we apply the amortized analysis performed above. Note that this approach is valid since the amortized analysis is insensitive of the order of appearance of the queries in X and the queries in $S \setminus X$.

Finally, suppose that there is a series of queries at the end of the strategy (a suffix) performed to edges incident to a $\frac{1}{\Delta+1}$ -heavy vertex q such that all replies point to q and q remains $\frac{1}{\Delta+1}$ -heavy till the end of the strategy. Note that in such a case q is the target. The vertex q had the uniquely smallest lie counter just before those queries. This in particular implies that the lie counter is strictly smaller than L . We artificially add a sequence of *pseudo-queries*, each of which increments the lie counter of q until it reaches L . This implies that the suffix of the search strategy now consists of a reply (which comes from a regular query or a pseudo-query) which does not point to q . Thus, we use again the arguments from our amortized analysis: we can find a segment and pair with it the above mentioned query pointing away from q . ◀

Proof of Theorem 7

Proof. Similarly as in the case of vertex queries, the generic strategy in Algorithm EDGE for edge queries and its corresponding bound for a fixed number of lies can be used to provide strong bounds for linearly bounded and probabilistic error models.

Let $\Gamma = 1 + \frac{\Delta+1}{\Delta} \frac{\varepsilon}{1-\varepsilon} = \frac{1-r}{r} \cdot \frac{1}{\Delta}$. Denote $Q_{\min} = \frac{\ln n}{\ln(1+\frac{\Gamma-1}{\Gamma\Delta+1})-r \ln \Gamma}$. We run Algorithm EDGE with bound $L = Q_{\min} r$ and parameter Γ set as just mentioned above. Then, by Theorem 6, the length of the strategy is at most $\frac{1}{\ln(1+\frac{\Gamma-1}{\Gamma\Delta+1})} \cdot \ln n + \frac{\ln \Gamma}{\ln(1+\frac{\Gamma-1}{\Gamma\Delta+1})} \cdot Q_{\min} r = Q_{\min} = L/r$. To conclude the proof, we bound

$$Q_{\min} = \frac{\ln n}{F(\varepsilon)/(\Delta+1) + F(-\varepsilon/\Delta) \cdot \Delta/(\Delta+1)} =$$

(where $F(x) \stackrel{\text{def}}{=} x + (1-x) \ln(1-x) = \sum_{i=2}^{\infty} \frac{x^i}{i(i-1)}$)

$$= \frac{\ln n}{\sum_{i=2}^{\infty} \frac{\varepsilon^i}{i(i-1)} \frac{\Delta^{i-1} + (-1)^i}{(\Delta+1)\Delta^{i-1}}} \leq \frac{\ln n}{\varepsilon^2/(2\Delta)} = 2\varepsilon^{-2} \Delta \ln n. \quad \blacktriangleleft$$

Proof of Theorem 8

Proof. For edge queries, we use a two step approach: first, we repeatedly ask queries to boost their error rate from $\sim 1/2$ to below $1/(\Delta+1)$, and then use the linearly bounded error strategy.

As a first step, we show that for $p_0 = \frac{1}{\Delta+1}(1-\varepsilon_0)$, there exists a strategy that locates the target with high probability using $\mathcal{O}(\Delta \log n / \varepsilon_0^2)$ edge queries. Indeed, assume without loss of generality that $\varepsilon_0 < 1/2$. We fix $\varepsilon_1 = \varepsilon_0 / (1 + \sqrt{\frac{3}{2} \frac{\Delta+1}{\Delta} \ln \delta^{-1} / \ln n})$, and use Theorem 7

with error rate $r_0 = \frac{1}{\Delta+1}(1 - \varepsilon_1)$. By Theorem 7, we obtain that the strategy length is $Q = 2\varepsilon_1^{-2}\Delta \ln n = \mathcal{O}(\Delta\varepsilon_0^{-2}(\log n + \log \delta^{-1}))$. The expected number of lies is $\mathbb{E}[L] = p_0 \cdot Q$ and by the Chernoff bound,

$$\Pr[L \geq r_0 \cdot Q] \leq \exp\left(-\frac{1}{3}\left(\frac{r_0}{p_0} - 1\right)^2 \cdot p_0 \cdot Q\right) \leq \exp\left(-\frac{2}{3}\left(\frac{\varepsilon_0 - \varepsilon_1}{\varepsilon_1}\right)^2 \cdot \frac{\Delta}{\Delta+1} \ln n\right) = \delta.$$

We now observe that to achieve the error rate of $\frac{1}{2}(1 - \varepsilon)$, we can boost the query error rate to be smaller by repeating the same query multiple times and taking the majority answer. By repeating each query $P = \mathcal{O}(\log(2\Delta + 2) \cdot \varepsilon^{-2})$ times, we get the correct answer with probability $1 - p' = 1 - \frac{1}{2} \cdot \frac{1}{\Delta+1}$, and as shown already, we only need $\mathcal{O}(\Delta(\log n + \log \delta^{-1}))$ queries with the error rate p' to locate the target with probability at least $1 - \delta$. Thus the claimed bound follows. \blacktriangleleft

As an immediate corollary we obtain a very simple strategy for noisy binary search in an integer range of complexity $\mathcal{O}(\varepsilon^{-2}(\log n + \log \delta^{-1}))$ matching [20].

B Application: Searching Unbounded Integer Ranges

Building on our generic strategies, we now obtain a general technique for searching an unbounded domain $\mathbb{N} = \{1, 2, \dots\}$ with comparison queries. Here the measure of complexity is the dependency on the error rate (number of lies) and on N , the (initially unknown) position of the target. The main idea is to use Algorithms VERTEX and EDGE, tweaking the initial weight distribution. We fix the *initial* weight of an integer n to be $\mu_0(n) = n^{-2}$. The total initial weight then equals $\pi^2/6 = \Theta(1)$. We provide the following bounds.³

► **Corollary 11.** *There exists a strategy that finds an integer in an unbounded integer range (\mathbb{N}) using at most*

- $\frac{\log \frac{\pi^2}{6} + 2 \log N + L \log \Gamma}{\log \frac{2\Gamma}{\Gamma+1}}$ ternary queries, or
- $\frac{\log \frac{\pi^2}{6} + 2 \log N + L \log \Gamma}{\log \frac{3\Gamma}{2\Gamma+1}}$ binary (comparison) queries,

where N is the target, L is an upper bound on the number of (adversarial) lies and $\Gamma > 1$ is an arbitrarily selected coefficient.

Proof. We use Algorithm VERTEX for ternary queries; let the strategy length be Q . By the proof of Theorem 3, $\mu_Q \leq (\frac{2\Gamma}{\Gamma+1})^Q \cdot \frac{\pi^2}{6}$. The final weight is at least $\mu_Q \geq N^{-2} \cdot \Gamma^{-L}$, and the bound for ternary queries follows since the number of queries is at most $\log(\frac{\pi^2/6}{N^{-2}\Gamma^{-L}})/\log \frac{2\Gamma}{\Gamma+1}$.

The bound for binary queries is obtained analogously from Theorem 6 (note that $\Delta = 2$) since we apply Algorithm EDGE for binary queries. \blacktriangleleft

Simply setting $\Gamma = 2$ yields an $\mathcal{O}(\log N + L)$ length strategy with comparison queries on unbounded integer domains with a fixed number of L lies.

We need to restate the linearly bounded error model in the case of unbounded domains since the Responder does not know a priori the length of the strategy. We define this error model as follows: whenever the Questioner finds the target and thus declares the search to be completed after t rounds, it is guaranteed that at most rt lies have occurred throughout the search.

³ We note that the term *ternary* refers to a model in which each query selects an integer i and as a response receives information whether the target is smaller than i , equals i , or is greater than i .

► **Corollary 12.** *For the linearly bounded error model with an error rate r and an unbounded integer domain, there exists a strategy that finds the target integer N in:*

- $\mathcal{O}(\varepsilon^{-2} \log N)$ ternary queries when $r = \frac{1}{2}(1 - \varepsilon)$, or
- $\mathcal{O}(\varepsilon^{-2} \log N)$ binary queries when $r = \frac{1}{3}(1 - \varepsilon)$.

Proof. Consider ternary queries. We proceed analogously as in the proof of Theorem 1. We have that the initial weight is $\pi^2/6$. Run Algorithm VERTEX until there is a single n such that $\ell_n \leq r \cdot Q$. Any Q such that $Q \geq \ln(\pi^2/6)/\ln \frac{2\Gamma}{\Gamma+1} + 2 \ln N/\ln \frac{2\Gamma}{\Gamma+1} + L \ln \Gamma/\ln \frac{2\Gamma}{\Gamma+1}$ is an upper bound on the length of the strategy. We thus get an upper bound

$$Q \leq 2\varepsilon^{-2}(2 \ln N + \mathcal{O}(1)).$$

The binary case follows in an analogous manner. ◀

► **Corollary 13.** *In the probabilistic error model, the target integer N can be found in an unbounded integer range using $\mathcal{O}(\varepsilon^{-2}(\log N + \log \delta^{-1}))$ binary queries for $p = \frac{1}{2}(1 - \varepsilon)$, correctly with probability at least $1 - \delta$.*

Proof. Same proof strategy as for Theorem 8, with $\Delta = 2$, applies. ◀