Binary Search in Graphs as a Framework for Interactive Learning

Nicola Amadio

AY 2018/2019

Abstract

We consider the problem of searching for an unknown target vertex t in a positively weighted graph. Each vertex-query points to a vertex v and the response either admits that v is the target or provides any neighbors of v that lies on a shortest path from v to t. This model has been introduced for trees by Onak and Parys [1] and for general graphs by Emamjomeh-Zadeh et al. [2], where 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 $1 - p$). The latter was recently further improved by Dereniowski et el. [3], whose work, covering both the adversarial errors and independent noise models, uses a simpler and more efficient algorithm than the previous. As pointed out by Emamjomeh-Zadeh and Kempe [4], the above-described problem can be used as a framework for interactively learning models, such as (binary or non-binary) classifiers, orderings/rankings of items, or clusterings of data points. In this report, we provide a summarized exposition of the above topics, focusing on the first work of Emamjomeh-Zadeh et al. [2], and limiting ourself to the vertex-query case. We present the algorithm they used in the noise-free version, and the proofs of its query-complexity both in the case of undirected graphs and in the generalized case of directed graphs, for which they also provided a lower bound. Finally, we present the algorithm used by Dereniowski et al. [3], and its query-complexity results in both adversarial error and independent noise models, omitting the proofs.

1 Introduction and Related Works

With the pervasive reliance on machine learning systems across myriad application domains in the real world, these systems frequently need to be deployed before they are fully trained. This is particularly true when the systems are supposed to learn a specific user's (or a small group of users') personal and idiosyncratic preferences. As a result, we are seeing an increased practical interest in online and interactive learning across a variety of domains. This type of interaction can be seen as a game and, in particular, as a game on a graph.

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 [7], noisy binary search or noisy decision trees [8], [9], [10]. One needs to put some restriction on how often the Responder is allowed to lie. Following earlier works, [2] focused 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 [5], [4], [6], [11]. 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 [4] are the following. In learning a ranking the system aims at learning user's preference list [12], [13]. 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 the above 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 [14], [15]. 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 [2], [4]. Thus, the learning strategies derived from the algorithms in [2], [4] and [3] turn out to be quite efficient. We point out that the lies in the query game reflect the fact that the user may sometimes provide incorrect replies.

2 Deterministic Model

We now focus on the noise-free case, i.e., where $p = 1$. After having explained the notation we will use, we present the algorithm presented in [2], along with its analysis for both undirected graphs and (for a special kind of) directed graphs, for which [2] also provided a query-complexity lower bound.

2.1 Notation and Preliminaries

The algorithm is given a positively weighted connected undirected (or directed and strongly connected) graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. Edge weights are denoted by $\omega_e > 0$. The distance $d(u, v)$ between vertices u, v is the length of a shortest $u - v$ path with respect to the ω_e. For undirected graphs, $d(u, v) = d(v, u)$. For a vertex u and edge $e = (u, v)$ incident on u (out of u for directed graphs), we denote by $N(u, e) = \{w \in V \mid d(u, w) = \omega_e + d(v, w)\}$ the set of all vertices w for which e lies on a shortest $u - w$ path. Let t be the (unknown) target vertex. The guarantee made to the algorithm is that when it queries a vertex q which is not the target, with probability p, it will be given an edge e out of q such that $t \in N(q, e)$. Since G is assumed (strongly) connected, such an answer will always exist; if multiple edges e exist such that $t \in N(q, e)$, an arbitrary one (possibly chosen adversarially) will be returned. If q is the target, then with probability p, this fact is revealed to the algorithm. In both cases, with the remaining probability $1 - p$, the algorithm is given an arbitrary (adversarial) answer, which includes the possibility of the correct answer.

2.2 Searching Strategy and Analysis

Theorem 2.1. *There exists an adaptive querying strategy with the following property: given any undirected, connected and positively weighted graph G with an unknown target vertex t, the strategy will find t using at most $\log n$ queries. (This bound is tight even when the graph is the line.)*

Proof. In each iteration, based on the responses the algorithm has received so far, there will be a set $S \subseteq V$ of candidate nodes remaining one of which is the target. The strategy is to always query a vertex q minimizing the sum of distances to vertices in S, i.e., a 1-median of S (with ties broken arbitrarily). Notice that q itself may not lie in S. More formally, for any set $S \subseteq V$ and vertex $u \in V$, we define $\Phi_S(q) = \sum_{v \in S} d(u, v)$. The algorithm is given as Algorithm 1. First, note that $\Phi_S(q)$ and $N(q, e)$ can be computed using Dijkstra's algorithm, and q can be found by exhaustive search in linear time, so the entire algorithm takes polynomial time. We claim that it uses at most $\log n$ queries to find the target t. To see this, consider an iteration in which t was not found; let $e = (q, v)$ be the response to the query. We write $S^+ = S \cap N(q, e)$, and $S^- = S \backslash S^+$. By definition, the edge e lies on a shortest $q - u$ path for all $u \in S^+$, so that $d(v, u) = d(q, u) - \omega_e$ for all $u \in S^+$. Furthermore, for all $u \in S^-$, the shortest path from v to u can be no longer than those going through q, so that

$d(v,u) \leq d(q,u)+\omega_e$ for all $u \in S^-$. Thus, $\Phi_S(v) \leq \Phi_S(q)-\omega_e\cdot(|S^+|-|S^-|)$. By minimality of $\Phi_S(q)$, it follows that $|S^+| \leq |S^-|$, so $|S^+| \leq \frac{|S|}{2}$. Consequently, the algorithm takes at most log$n$ queries. □

Algorithm 1 Target Search for Undirected Graph

```
S ← V
while |S| > 1 do
    q ← a vertex minimizing Φ_S(q).
    if q is the target then
        return q.
    else
        e = (q, v) ← response to the query.
        S ← S ∩ N(q, e).
    end if
end while
return the only vertex in S.
```

A natural question is how much the logarithmic bound of Theorem 2.1 depends on the assumptions. Let's consider directed graphs: the example of a directed cycle shows that one can, in general, not hope to find a target using a sublinear number of queries. Thus, in order to achieve positive results, additional assumptions must be placed on the graph structure. We show that if the graph is "almost undirected," then the positive result of Theorem 2.1 degrades gracefully. Specifically, we assume that each edge e with weight ω_e is part of a cycle of total weight at most $c \cdot \omega_e$. Notice that for unweighted graphs, this means that each edge e is part of a cycle of at most c edges, and specifically for $c = 2$, the graph is undirected.

Theorem 2.2. *Algorithm 1 has the following property: if G is a strongly connected, positively weighted graph, in which each edge e belongs to a cycle of total weight at most $c \cdot \omega_e$, then the algorithm finds the target using at most*

$$\frac{1}{\ln c - \ln (c-1)} \cdot \ln n \leq c \ln 2 \cdot \log n$$

queries in the worst case.

Proof. In Algorithm 1, the potential is now defined with respect to directed distances. To analyze its performance, assume that the algorithm, in some iteration, queried the node q, and received as a response an edge $e = (q, v)$. We define the set S^+ and S^- in the proof of Theorem 3.1. As before, $d(v,u) = d(q,u) - \omega_e$ for all vertices $u \in S^+$. On the other hand, there exists a path of total weight at most $(c-1)\cdot\omega_e$ from v to q (since e appears in a cycle of length at most $c \cdot \omega_e$). Thus, for any vertex $u \in S^-, d(v,u) \leq d(q,u) + (c-1)\cdot\omega_e$. Therefore,

$$\Phi_S(v) \leq \Phi_S(q) - \omega_e \cdot (|S^+| - (c-1)\cdot|S^-|).$$

Since $\Phi_S(q)$ is minimal, $|S^+| \leq (c-1) \cdot |S^-|$, so $|S^+| \leq \frac{c-1}{c} \cdot |S|$. Thus, after at most $\log_{c/(c-1)}(n) = \frac{\ln n}{\ln c - \ln(c-1)}$ queries, the target must be identified. Finally

$$\ln c - \ln(c-1) = \int_c^{c-1} dx/x \geq 1/x,$$

implying that $\frac{\ln n}{\ln c - \ln(c-1)} \leq c \cdot \ln n = c \ln 2 \cdot \log n$. □

The upper bound of Theorem 2.2 is nearly matched, up to a factor of $\mathcal{O}(\log c)$, by the following lower bound.

Proposition 1. *For any integers N and $c \geq 2$, there exists an unweighted and strongly connected graph G of $n \geq N$ vertices, such that each edge in G belongs to a cycle of length c, and at least $\frac{c-1}{\log c} \cdot \log n$ queries are required to identify a target in G.*

Proof. We construct a family of unweighted strongly connected directed graphs $H_{c,k}$, $k = 0, 1, ...$ inductively. $H_{c,0}$ is a single vertex. For any $k \geq 1$, $H_{c,k}$ is obtained from c disjoint copies of $H_{c,k-1}$. For each of the c copies $i = 1, ..., c$, let v_i be an arbitrary vertex in that copy. Now add a directed cycle of length c through the vertices v_i. By construction, each edge is part of a cycle of length c; and by induction, $H_{c,k}$ is strongly connected for all k. We will prove by induction that any strategy requires at least $(c-1) \cdot k$ queries in the worst case to identify the target in $H_{c,k}$. Because $n = c^k$, this proves a lower bound of $\frac{c-1}{\log c} \cdot \log n$ on the number of queries in an n-vertex graph. The base case $k = 0$ of the induction is trivial. For the induction step, let $G_1, G_2, ..., G_c$ be the c copies of $H_{c,k-1}$ that were combined to form $H_{c,k}$. For $i < k$, define v'_i to be v_{i+1} and let v'_k be v_1. Consider a query of a node $q \in G_i$. By construction, v'_i lies on any path from q to any vertex not in G_i. In other words, if an algorithm receives an edge (q, v) lying on a shortest path from q to v'_i, it might learn that the target is not in G_i, but cannot infer which G_j, $j \neq i$ the target is in. The adversary's strategy is now simple: for the first $c-1$ queries, to each query $q \in G_i$, the adversary will give an edge toward v'_i, i.e., the first edge of a shortest path from q to v'_i. At this point, there is at least one G_i such that no vertex in G_i has been queried. The adversary now picks one such G_i (arbitrarily, if there are multiple), and commits the target to G_i. Then, he continues to answer queries to vertices in G_j, $j \neq i$ in the same way as before. Queries to vertices in G_i are answered with the adversarial strategy for $H_{c,k-1}$. As a result, after $c-1$ queries, there will always remain at least one entirely unqueried copy of $H_{c,k-1}$; the algorithm has no information about the location of the target vertex in this copy, and by induction hypothesis, it will take at least $(c-1) \cdot (k-1)$ queries to find the target. Adding the $c-1$ queries to reach this point gives us a total of at least $(c-1) \cdot k$ queries to find the target in $H_{c,k}$. □

3 Noisy Model

We now cover the strategy used by [3] in the noisy scenario, i.e., when $p < 1$. Under this assumption, in fact, the solution in [3] is both simpler and more efficient than the one in [2].

3.1 Notation and Preliminaries

We will need to add some extra notation in order to do that. 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 $t > 0$:

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

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 $l_v \leq L$, 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, $\mu_0(v) = n$ 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.

3.2 Searching Strategy and Analysis

The overall strategy is based on the same idea of the one in Section 2.2 by [2]: querying a *median* and reducing the set of the searchable items according to the user feedback. Now the problem is more complex since the user is allowed to lie from time to time. In [3] this is handled by using a clever and simple method to update the weights. In their analysis, [3] 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.

Theorem 3.1. *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 applying a Chernoff bound, we get the following query complexity.

Theorem 3.2. *In the probabilistic error model with error probability $p < 1/2$, the target can be found using at most*

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

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

The following is the algorithm used by [3]:

Algorithm 2 Vertex

for $v \in V$ **do**
  $\mu(v) = 1$
  $l_v = 0$
end for
while more than one vertex $x \in V$ has $l_x \leq L$ **do**
  $q = \text{argmin}_{x \in V}\ \Phi(x)$.
  query the vertex q
  for all nodes u not compatible with the answer **do**
    $l_u = l_u + 1$
    $\mu(u) = \mu(u)/\Gamma$
  end for
end while
return the only x such that $l_x \leq L$

4 Conclusion and Open Problems

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., the technique shown for vertex queries does not transfer without changes. We refer the reader to the two works this summary is based on ([3] and [2]) for a broader and more complete analysis of the subject and to [4] for a complete exposition of the application of the presented model to interactive learning, with focus on ranking, clustering and classification. Finally we point out that the query-complexity bounds we discussed in this paper do not consider the actual structure of the graph. For example, for undirected graphs, the proved upper bound is $\log n$, but we know that in the case of a clique the time

required to reach the target vertex is constant. It would be nice, therefore, to include some graph-theoretic quantity in the analysis, and possibly making the bound tight by doing so. For example, we could consider looking at the independence number α or at the domination number δ. This may also require a slightly different approach (maybe one of type *minimax*) and possibly another algorithm. We finally refer the reader to a set of works ([16][17]), related to the problem of online learning with feedback graphs, which provides an example of how such graph-teoretic properties have been considered in a problem somehow close to the one presented in this paper.

References

[1] Krzysztof Onak and Pawel Parys. *Generalization of binary search: Searching in treesand forest-like partial order*. In FOCS, pages 379–388, 2006.

[2] Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. *Deterministic and probabilistic binary search in graphs*. In STOC, pages 519–532, 2016.

[3] Dariusz Dereniowski, Stefan Tiegel, Przemysław Uznański, and Daniel Wolleb-Graf. *A Framework for Searching in Graphs in the Presence of Errors*. In arXiv preprint arXiv:1804.02075, 2019

[4] 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

[5] D. Angluin. *Queries and concept learning*. Machine Learning, 2:319–342, 1988

[6] N. Littlestone. *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*. Machine Learning, 2:285–318, 1988

[7] Alfréd Rényi. *On a problem of information theory*. MTA Mat. Kut. Int. Kozl., 6B:505–516, 1961.

[8] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. *Computing with noisyinformation*. SIAM J. Comput., 23(5):1001–1018, 1994.

[9] Richard M. Karp and Robert Kleinberg. *Noisy binary search and its applications*. In SODA, pages 881–890, 2007

[10] 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.

[11] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool Publishers, 2012.

[12] 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.

[13] Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer, 2011

[14] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. *Local algorithms for interactive clustering*. Journal of Machine Learning Research, 18:3:1–3:35, 2017.

[15] 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.

[16] N. Alon, N. Cesa-Bianchi, O. Dekel, and T. Koren. *Online Learning with Feedback Graphs: Beyond Bandits*. In Proceedings of the 28th Conference on Learning Theory (COLT). JMLR Workshop and Conference Proceedings, 40:23-35, 2015.

[17] S. Mannor and O. Shamir. *From Bandits to Experts: On the Value of Side-Observations*. In Neural Information Processing Systems (NIPS), 2011.