Noisy Interactive Graph Search

Qianhao Cong
Department of Industrial Systems Engineering and Management
National University of Singapore
cong_qianhao@u.nus.edu

Jing Tang*
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong Uni. of Sci. and Tech.
jingtang@ust.hk

Kai Han
School of Computer Science and Technology
Soochow University
hankai@suda.edu.cn

Yuming Huang
Department of Industrial Systems Engineering and Management
National University of Singapore
huangyuming@u.nus.edu

Lei Chen
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong Uni. of Sci. and Tech.
leichen@ust.hk

Yeow Meng Chee
Department of Industrial Systems Engineering and Management
National University of Singapore
ymchee@nus.edu.sg

ABSTRACT

The interactive graph search (IGS) problem aims to locate an initially unknown target node leveraging human intelligence. In IGS, we can gradually find the target node by sequentially asking humans some reachability queries like "is the target node reachable from a given node x?". However, human workers may make mistakes when answering these queries. Motivated by this concern, in this paper, we study a noisy version of the IGS problem. Our objective in this problem is to minimize the query complexity while ensuring accuracy. We propose a method to select the query node such that we can push the search process as much as possible and an online method to infer which node is the target after collecting a new answer. By rigorous theoretical analysis, we show that the query complexity of our approach is near-optimal up to a constant factor. The extensive experiments on two real datasets also demonstrate the superiorities of our approach.

CCS CONCEPTS

• **Information systems** → *Crowdsourcing*; • **Theory of computation** → **Graph algorithms analysis**.

KEYWORDS

Crowdsourcing; Interactive Graph Search; Algorithms

ACM Reference Format:
Qianhao Cong, Jing Tang, Kai Han, Yuming Huang, Lei Chen, and Yeow Meng Chee. 2022. Noisy Interactive Graph Search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington DC*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

*Corresponding author: Jing Tang.

KDD '22, August 14–18, 2022, Washington DC

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

Crowdsourcing has been a popular technique to address the problems that are hard to solve by computer alone. The crowdsourcing platforms, such as Amazon's Mechanical Turk and CrowdFlower, allow users to set up human-aided tasks so that the crowd workers can solve them in exchange for rewards. By the crowdsourcing platforms, one can perform some tasks that are hard for computers but easy for humans at a larger scale under lower cost, such as data labeling [23, 26, 45], data collection [16, 33, 48], data mining [1, 12], data filtering and cleaning [30, 32, 44], and many other tasks [9, 13, 18, 34, 37].

Among these tasks, data labeling is perhaps the most popular one, due to the fast development of machine learning techniques and their great need for labeled datasets. However, creating datasets is a burdensome task, especially for the datasets with complex structures and underlying hierarchies, such as ImageNet and Campus3D [10, 21]. Although datasets of this kind inspire many new machine learning methods and algorithms, generating them is still an expensive and challenging task. In the traditional approaches, one has to employ many experts and spend a long time labeling all the data. These experts are required to be familiar with the whole hierarchy, which is very challenging and even impractical when the hierarchy is extremely large (e.g., the hierarchy of ImageNet has more than 20,000 nodes).

Crowdsourcing techniques can help to generate this kind of dataset at a lower cost. The key task of labeling data according to a hierarchy is to find the deepest node that best describes a certain object. Based on this observation, Tao et al. [35] initiated the interactive graph search (IGS) problem recently. They model this task as a problem of locating a (hidden) target node, i.e., the deepest suitable node, in the given hierarchy by interactive queries. Specifically, the task is to locate the target by several rounds of reachability queries. In each round, they select a node x from the graph and ask "is the target node reachable from node x?". By collecting answers from the crowd, the pool of candidate nodes can be gradually narrowed down until the target node is finally identified. The goal of the IGS problem is to locate the target node using a minimum number of queries, since crowdsourcing platforms usually charge a flat fee for each query.

Let us consider an example task of acquiring the label of an image via crowdsourcing given the hierarchy shown in Figure 1. Suppose

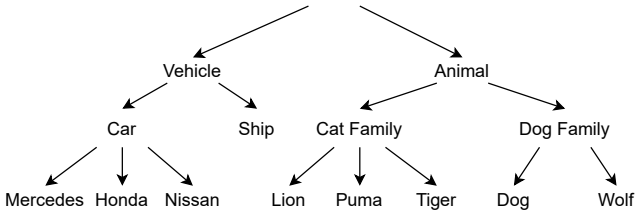


Figure 1: Example Hierarchy

that we are given the image of a *honda* car. We do not know what the picture shows and try to identify the label by showing the picture to the workers and asking the question "is the image a *vehicle*?". Since the edges in the hierarchy represent concept-instance relationships, e.g., *car* and *ship* are instances of *vehicle*, this question is equivalent to asking whether the node corresponding to *vehicle* can reach the hidden target node *honda* by a directed path. For this query, we may get a *yes* answer from the workers, and then continue to ask "is the image a *car*?" and "is the image a *honda*?" sequentially. If the workers answer correctly, we would get two *yes* answers again and finally label the image as *honda*. Note that although the *vehicle* and *car* also correctly describe the image, they are not considered as the target, since *honda* is the deepest node with the most suitable concept. Similarly, if we are given an image showing a *vehicle* but neither a *car* nor a *ship*, e.g., an airplane, we may ask "is the image a *vehicle*?", "is the image a *car*?", and "is the image a *ship*". The workers would answer *yes*, *no*, and *no* respectively, and then we can label the image as a *vehicle*. Note that, in the IGS problem, the tasks are decomposed into a set of queries with binary answers that can be easily addressed by human workers. Therefore, the workers do not have to be familiar with the whole hierarchy.

The key challenge in the IGS problem is to decide which node to query in each round. Tao et al. [35] address this problem utilizing heavy-path decomposition and binary search on the heavy-paths. Following their work, some variants of the IGS problem are studied in the literature. Li et al. [23] investigate how to locate the target node by multiple-choice queries. Zhu et al. [47] consider a budget-constrained IGS problem, where only a limited number of queries are allowed and multiple target nodes may exist in the input hierarchy. However, all the existing proposals assume that the answers to reachability queries are obtained through a reliable oracle without any errors. In practice, the queries are typically answered by human workers on the crowdsourcing platform, and these workers are likely to make mistakes since they are not experts. This observation motivates us to develop an effective algorithm that is robust against noisy answers during the searching process of the IGS problem.

Specifically, in this paper we consider a noisy version of the interactive graph search problem, where the workers may provide a wrong answer to each query with a certain probability. Note that the difficulty of queries on different nodes may vary significantly. For example, "is this an *animal*" is much easier than "is this a *puma*", and "is this a *car*" is much easier than "is this a *honda*'. Therefore, we consider a general case where different queries may have different probabilities to be answered correctly.

The simplest method against noise is to repeat each query multiple times. For example, we may ask three different workers to answer the same question, and regard the answer appearing more than twice as the correct one. This method is also known as majority voting and has been widely adopted in extensive applications [9, 23, 34, 35, 46]. However, this method is not cost-effective, and it is hard to control the accuracy guarantee. In this paper, we propose a new method to improve cost-effectiveness while guaranteeing any accuracy, through a Bayes approach that estimates the probability of each node in the hierarchy being the target node. More specifically, in each round, we select the most informative node that can increase the posterior probability of the target node as much as possible, and returns a node as the target node once we have enough confidence about it, i.e., its posterior probability exceeds the accuracy requirement. By rigorous theoretical analysis, we show that our method can achieve nearly optimal query complexity under the desired accuracy.

Our contributions are listed as follows:

- We formally formulate the noisy interactive graph search problem (Section 2).
- We show how to compute the posterior probabilities based on the Bayes method and how to infer which node is the target. Our theoretical analysis also shows that the expected probability for identifying the correct target is non-decreasing by running a new query (Section 3).
- We propose a method to select the query node such that we can increase the posterior probability of the target node as much as possible. We show that our method achieves a near-optimal (up to a constant factor) query complexity of $2\log_{\alpha}(n)$ in expectation, where n is the number of nodes, and $\alpha > 1$ is a parameter based on the accuracy requirement, the error rate of queries, and the hierarchy structure (Section 4).
- Our extensive experiments demonstrate that our approach can achieve the same accuracy level under much lower cost compared to state-of-the-art algorithms (Section 5).

2 PROBLEM DEFINITION

In this paper, we study the *noisy interactive graph search (OIGS)* problem. In this problem, we are given an unlabeled object and our task is to find out its label by asking reachability questions with (noisy) boolean answers, i.e., *yes* or *no*. Since the cost of crowdsourcing is usually determined by the number of questions asked, our goal is to find a label using the minimum number of questions while guaranteeing accuracy.

The OIGS problem is formally defined as follows. We are given a categorization hierarchy abstracted as a tree $T = (V, E)$ with a set V of n nodes and a set E of $(n-1)$ edges. Let d denote the maximum degree of any node in the hierarchy. Given such a hierarchy T, the OIGS problem aims to find an initially unknown target node z by several rounds of noisy reachability queries. In each round, we interactively pick up a query node q and ask "is the target node z reachable from node q". For any query node $q \in V$, the correct answer of such a reachability query is boolean, denoted as $reach(q)$, which is given as follows.

$$reach(q) := \begin{cases} yes, & \text{if there is a directed path from } q \text{ to target } z, \\ no, & \text{otherwise.} \end{cases}$$

For convenience, for any two nodes u and v in T, we use $u \rightarrow v$ to denote that u can reach v and use $u \not\rightarrow v$ to denote that u cannot reach v, respectively.

The result of these queries will be collected from human workers and hence may introduce some noise. In this paper, we adopt the one-coin noise model which is widely used in the literature [13, 43, 44, 46]. That is, the answer from the oracle can be wrong with a certain probability. Since the difficulty of querying on different nodes may vary significantly, as mentioned in the introduction, we assume that the queries on different nodes have different error rates. Specifically, for the query on node q, there is a parameter e_q characterizing the error probability of such a query by considering workers' average ability in crowdsourcing. Let a be the answer from a human worker for the query q. Given an unknown target z, let $I(q, a; z)$ be an indicator function representing whether the answer a of query q is correct or not, i.e.,

$$I(q,a;z) := \begin{cases} 1, & \text{if } (q \rightarrow z \wedge a = yes) \text{ or } (q \not\rightarrow z \wedge a = no), \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Thus, the conditional probability of observing the result of (q, a) given that z is the target is given by

$$\mathbb{P}((q,a) \mid z) = \begin{cases} 1 - e_q, & \text{if } I(q,a;z) = 1, \\ e_q, & \text{otherwise.} \end{cases} \tag{2}$$

In this paper, we consider that the queries are independent so that the answer to a query only depends on the reachability and the error probability, not influenced by other queries [13, 32, 37, 44]. We make the assumption of rationality [13, 37] such that there does not exist malicious workers who deliberately make mistakes, i.e., $e_q < 0.5$ for every $q \in V$. Following previous work [8, 23, 47], we consider that the target nodes follow an a-priori known distribution where each node v in the hierarchy is associated with a probability $\mathbb{P}(v)$ measuring the likelihood of v being the target. In practice, the a-priori distribution can be estimated by the online learning method [8, 23], i.e., estimating the a-priori distribution by using the distribution of the labeled data and updating it after each time we get a new labeled object, or we can simply use the discrete uniform distribution instead [47]. Due to query noise, we aim to identify the target node with a high accuracy of at least $1 - \varepsilon$. The formal definition of OIGS is given in the following:

Definition 1 (Noisy Interactive Graph Search). *Given a tree hierarchy $T = (V, E)$ and an (unknown) target node $z \in V$ following the a-priori known probability distribution $\mathbb{P}(\cdot)$, OIGS asks for a query strategy that can correctly locates z with a probability of at least $1 - \varepsilon$ under the minimum expected cost.*

For the OIGS problem, the main challenges are two-fold: (i) how to infer which node is the target given a set of queries and their answers, and (ii) how to select the node to query based on the information collected previously.

3 TARGET NODE INFERENCE

In this section, we infer the target node given a set of queries and their corresponding answers, leveraging Bayes' theorem. Moreover, we show that the expected probability of identifying the correct target is non-decreasing when running new queries. For ease of reference, Table 1 lists the notations that we frequently use.

Table 1: Frequently Used Notations.

Notations	Description
$T = (V, E)$	a tree with node set V and edge set E
n	the number of nodes in T
d	the maximum degree of nodes in T
e_u	the error probability of query on node u
T_u	a subtree of T rooted at node u
(q, a)	a pair of query and its (boolean) answer
Q	a set of query results $Q = \{(q_i, a_i) \colon i = 1, 2, \dots\}$
ε	threshold of failure probability

3.1 Computing Posterior Probability

We measure the likelihood of a node being the target utilizing Bayes' theorem and choose the one with the highest posterior probability given the information we have collected so far. Moreover, denote by Q a set of observations obtained from k queries, i.e., $Q = \{(q_1, a_1), (q_2, a_2), \dots, (q_k, a_k)\}$. Recall that the queries are independent. Thus, the probability of observing Q conditioned on z being the target can be computed by

$$\mathbb{P}(Q \mid z) = \prod_{(q,a) \in Q} \mathbb{P}((q,a) \mid z). \tag{3}$$

As a result, given the prior probability $\mathbb{P}(v)$ of node v being the target for every node $v \in V$, the total probability $\mathbb{P}(Q)$ of Q occurring is given by

$$\mathbb{P}(Q) = \sum_{v \in V} \mathbb{P}(Q \mid v)\mathbb{P}(v).$$

Finally, according to Bayes' theorem, given the current information Q collected, the posterior probability $\mathbb{P}(u \mid Q)$ of node u being the target can be calculated as follows,

$$\mathbb{P}(u \mid Q) = \frac{\mathbb{P}(Q \mid u)\mathbb{P}(u)}{\mathbb{P}(Q)} = \frac{\mathbb{P}(Q \mid u)\mathbb{P}(u)}{\sum_{v \in V} \mathbb{P}(Q \mid v)\mathbb{P}(v)}. \tag{4}$$

Based on the posterior probability obtained via the above equation, we return the node with the maximum likelihood (i.e., posterior probability) as the guessed target.

After collecting the answer to the k-th query, we can directly apply (4) to compute the posterior probability for every node being the target from scratch in $O(nk)$ time. Alternatively, we can incrementally update the posterior probability for every node based on the posterior probability obtained just before asking the k-th query. Specifically, consider a newly observed query result (q, a) on top of previously collected information Q. According to (4), for any node $u \in V$, we have

$$\mathbb{P}(u \mid Q \cup \{(q,a)\}) = \frac{\mathbb{P}(Q \cup \{(q,a)\} \mid u)\mathbb{P}(u)}{\sum_{v \in V} \mathbb{P}(Q \cup \{(q,a)\} \mid v)\mathbb{P}(v)}.$$

In addition, by (3), we know that

$$\mathbb{P}(Q \cup \{(q,a)\} \mid v) = \mathbb{P}((q,a) \mid v)\mathbb{P}(Q \mid v).$$

Putting it together yields

$$\begin{aligned}\mathbb{P}(u \mid Q \cup \{(q,a)\}) &= \frac{\mathbb{P}((q,a) \mid u)\mathbb{P}(Q \mid u)\mathbb{P}(u)}{\sum_{v\in V}\mathbb{P}((q,a) \mid v)\mathbb{P}(Q \mid v)\mathbb{P}(v)} \\ &= \frac{\mathbb{P}((q,a) \mid u)\mathbb{P}(Q \mid u)\mathbb{P}(u)/\mathbb{P}(Q)}{\sum_{v\in V}\mathbb{P}((q,a) \mid v)\mathbb{P}(Q \mid v)\mathbb{P}(v)/\mathbb{P}(Q)} \\ &= \frac{\mathbb{P}((q,a) \mid u)\mathbb{P}(u \mid Q)}{\sum_{v\in V}\mathbb{P}((q,a) \mid v)\mathbb{P}(v \mid Q)}. \end{aligned} \tag{5}$$

It can be seen that, for each new query result (q,a) on top of previously collected information Q, using (5) to update the posterior probability of every node being the target just takes $O(n)$ time, which reduces a multiplicative factor of k compared with that by applying the naive calculation.

3.2 Monotonicity of Posterior Probability

By (1) and (2), for any node $u \in V$ and any query outcome (q,a), we have

$$\mathbb{P}((q,a) \mid u) = \begin{cases} 1-e_q, & \text{if } q \to u \wedge a = yes, \\ e_q, & \text{if } q \to u \wedge a = no, \\ e_q, & \text{if } q \not\to u \wedge a = yes, \\ 1-e_q, & \text{if } q \not\to u \wedge a = no. \end{cases} \tag{6}$$

In addition, denote by T_q the subtree of T rooted at q (i.e., consisting of q and all of its descendants in T) and by $T_q^{\complement} = T \setminus T_q$ the complement tree of T_q. Let $\lambda(q,a;u,Q)$ be the multiplier in (5), i.e.,

$$\lambda(q,a;u,Q) := \begin{cases} \frac{1-e_q}{(1-e_q)\mathbb{P}(T_q|Q)+e_q\mathbb{P}(T_q^{\complement}|Q)}, & \text{if } q \to u \wedge a = yes, \\ \frac{e_q}{e_q\mathbb{P}(T_q|Q)+(1-e_q)\mathbb{P}(T_q^{\complement}|Q)}, & \text{if } q \to u \wedge a = no, \\ \frac{e_q}{(1-e_q)\mathbb{P}(T_q|Q)+e_q\mathbb{P}(T_q^{\complement}|Q)}, & \text{if } q \not\to u \wedge a = yes, \\ \frac{1-e_q}{e_q\mathbb{P}(T_q|Q)+(1-e_q)\mathbb{P}(T_q^{\complement}|Q)}, & \text{if } q \not\to u \wedge a = no, \end{cases} \tag{7}$$

where $\mathbb{P}(T_q \mid Q) = \sum_{v\in T_q}\mathbb{P}(v \mid Q)$ and $\mathbb{P}(T_q^{\complement} \mid Q) = 1 - \mathbb{P}(T_q \mid Q)$ are the total posterior probabilities of the nodes in T_q and $T_q^{\complement}$ with respect to Q, respectively. Then, (5) can be rewritten as

$$\mathbb{P}(u \mid Q \cup \{(q,a)\}) = \lambda(q,a;u,Q) \cdot \mathbb{P}(u \mid Q). \tag{8}$$

Leveraging (8), we show that asking new queries never hurts with respect to the expected posterior probability of the (unknown) target node z.

Theorem 1. *Given that the target node is z, for any new query q on top of Q, we have*

$$\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] \geq \mathbb{P}(z \mid Q), \tag{9}$$

where the expectation is over the randomness of query outcome X.

Proof. We consider two scenarios with respect to the reachability of q to z, i.e., (i) $z \in T_q$ (i.e., $q \to z$) and (ii) $z \in T_q^{\complement}$ (i.e., $q \not\to z$).

We first analyze case (i) that $z \in T_q$. According to (8), we have

$$\begin{aligned}&\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] \\ &= \sum_{a\in\{yes,no\}} \Big(\mathbb{P}((q,a) \mid z) \cdot \lambda(q,a;z,Q) \cdot \mathbb{P}(z \mid Q)\Big).\end{aligned}$$

Using $\mathbb{P}((q,a) \mid z)$ and $\lambda(q,a;z,Q)$ in (6) and (7), we can get that

$$\begin{aligned}&\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] \\ &= \frac{(1-e_q)^2 \cdot \mathbb{P}(z \mid Q)}{(1-e_q)\mathbb{P}(T_q \mid Q) + e_q\mathbb{P}(T_q^{\complement} \mid Q)} \\ &\quad + \frac{e_q^2 \cdot \mathbb{P}(z \mid Q)}{e_q\mathbb{P}(T_q \mid Q) + (1-e_q)\mathbb{P}(T_q^{\complement} \mid Q)}. \end{aligned} \tag{10}$$

In fact, for any $x, y \in [0,1]$, it holds that

$$\begin{aligned}&\frac{(1-x)^2}{(1-x)y + x(1-y)} + \frac{x^2}{xy + (1-x)(1-y)} - 1 \\ &= \frac{(1-x)^2 - (1-x)(x+y-2xy)}{x+y-2xy} + \frac{x^2 - x(1-x-y+2xy)}{1-x-y+2xy} \\ &= \frac{(1-x)(1-2x)(1-y)}{x+y-2xy} - \frac{x(1-2x)(1-y)}{1-x-y+2xy} \\ &= \frac{(1-2x)^2(1-y)^2}{(x+y-2xy)(1-x-y+2xy)} \\ &= \frac{(1-2x)^2}{x(1-x)(\frac{1-x}{x}+\frac{y}{1-y})(\frac{x}{1-x}+\frac{y}{1-y})} \\ &\geq 0. \end{aligned} \tag{11}$$

Combining (10) and (11) by setting $x = e_q$ and $y = \mathbb{P}(T_q \mid Q)$ in (11) gives rise to

$$\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] - \mathbb{P}(z \mid Q) \geq 0.$$

The analysis for case (ii) that $z \in T_q^{\complement}$ is similar. That is,

$$\begin{aligned}&\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] \\ &= \frac{e_q^2 \cdot \mathbb{P}(z \mid Q)}{(1-e_q)\mathbb{P}(T_q \mid Q) + e_q\mathbb{P}(T_q^{\complement} \mid Q)} \\ &\quad + \frac{(1-e_q)^2 \cdot \mathbb{P}(z \mid Q)}{e_q\mathbb{P}(T_q \mid Q) + (1-e_q)\mathbb{P}(T_q^{\complement} \mid Q)}. \end{aligned} \tag{12}$$

Combing (11) and (12) by setting $x = e_q$ and $y = \mathbb{P}(T_q^{\complement} \mid Q)$ in (11) immediately yields

$$\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] - \mathbb{P}(z \mid Q) \geq 0.$$

This completes the proof. □

Theorem 1 implies that asking more questions never hurts in expectation. In particular, it is trivial to verify that when $e_q < 0.5$, unless $\mathbb{P}(T_q \mid Q) = 1$ (resp. $\mathbb{P}(T_q^{\complement} \mid Q) = 1$) such that z is ensured to be in T_q (resp. $T_q^{\complement}$) based on Q, the query on q strictly increases the posterior probability of z in expectation.

4 QUERY SELECTION

In this section, we study the strategy of query selection in the OIGS problem. Intuitively, we attempt to select the node that can increase the posterior probability of the target node as much as possible. With rigorous theoretical analysis, we show that our algorithm achieves a query complexity of $2\log_{\alpha}(n)$ in expectation, where $\alpha > 1$ is a constant determined by the hierarchy structure, the query noise, and the allowed failure probability, as elaborated later.

Algorithm 1: OIGS

Input: an input tree T, prior propability $\mathbb{P}(v)$ and error rate e_v for each node $v \in V$, failture threshold ε;
Output: the correct target node with probability $1-\varepsilon$;

```
1  Q ← ∅;
2  ℙ(v | Q) ← ℙ(v) for every node v ∈ V;
3  p* ← max_{v∈V} ℙ(v | Q);
4  while p* < 1 − ε do
5      q ← arg max_{v∈V} {min{f(v | Q), g(v | Q)}};
6      get the answer a of query q from a worker;
7      compute ℙ((q, a) | v) for every node v ∈ V based on (2);
8      compute p_sum ← Σ_{v∈V} ℙ((q, a) | v)ℙ(v | Q);
9      update ℙ(v | Q ∪ {(q, a)}) ← ℙ((q,a)|v)ℙ(v|Q) / p_sum for v ∈ V;
10     update Q ← Q ∪ {(q, a)};
11     update p* ← max_{v∈V} ℙ(v | Q);
12 return arg max_{v∈V} ℙ(v | Q);
```

Our query complexity is just within a factor of $2/\log_2(\alpha)$ compared to the super lower bound of $\log_2(n)$.

4.1 Greedy Strategy

According to the analysis of $\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q]$ in Theorem 1, we define two functions $f(\cdot)$ and $g(\cdot)$ for the query q on top of Q as follows, i.e.,

$$f(q \mid Q) := \frac{(1-2e_q)^2}{e_q(1-e_q)\left(\frac{1-e_q}{e_q} + \frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)}\right)\left(\frac{e_q}{1-e_q} + \frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)}\right)} + 1, \tag{13}$$

$$g(q \mid Q) := \frac{(1-2e_q)^2}{e_q(1-e_q)\left(\frac{1-e_q}{e_q} + \frac{\mathbb{P}(T_q^{\complement}|Q)}{\mathbb{P}(T_q|Q)}\right)\left(\frac{e_q}{1-e_q} + \frac{\mathbb{P}(T_q^{\complement}|Q)}{\mathbb{P}(T_q|Q)}\right)} + 1. \tag{14}$$

Then, we have

$$\mathbb{E}[\mathbb{P}(z \mid Q \cup \{(q,X)\}) \mid Q] = \begin{cases} f(q \mid Q) \cdot \mathbb{P}(z \mid Q), & \text{if } z \in T_q, \\ g(q \mid Q) \cdot \mathbb{P}(z \mid Q), & \text{otherwise.} \end{cases}$$

Since we do not know whether the target node z is reachable from the selected node for query, we greedily choose q that maximizes the smaller of $f(q \mid Q)$ and $g(q \mid Q)$. Formally,

$$q = \arg\max_{v \in V} \{\min\{f(v \mid Q), g(v \mid Q)\}\}.$$

Algorithm 1 gives the pseudocode of our greedy strategy based on the above analysis. Specifically, in each round, it selects the node that maximizes the multiplier $\min\{f(q \mid Q), g(q \mid Q)\}$ to increase the expected posterior probability of the target z (line 5). Upon receiving the query outcome (line 6), it incrementally updates the posterior probability (lines 7–9) and the maximum likelihood (line 11) according to the method explained in Section 3.1. This search process repeats until the maximum likelihood exceeds the predefined threshold $1-\varepsilon$, and then the node with the maximum likelihood is returned as the guessed target (line 12).

4.2 Theoretical Guarantees

In what follows, we show that the expected number of queries invoked by Algorithm 1 is nearly optimal. Specifically, it is evident that $f(q \mid Q)$ decreases with the increasing of $\frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)}$, while $g(q \mid Q)$ increases with $\frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)}$. Reformulating (13) and (14), we can get

$$f(q \mid Q) := (1-2e_q)\left(\frac{1}{\frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)} + \frac{e_q}{1-e_q}} - \frac{1}{\frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)} + \frac{1-e_q}{e_q}}\right) + 1,$$

$$g(q \mid Q) := (1-2e_q)\left(\frac{1}{\frac{\mathbb{P}(T_q^{\complement}|Q)}{\mathbb{P}(T_q|Q)} + \frac{e_q}{1-e_q}} - \frac{1}{\frac{\mathbb{P}(T_q^{\complement}|Q)}{\mathbb{P}(T_q|Q)} + \frac{1-e_q}{e_q}}\right) + 1.$$

It is also evident to see that both of $f(q \mid Q)$ and $g(q \mid Q)$ decrease with the increasing of e_q. Let $e^* := \max_{v \in V} e_v$ be the maximum error rate and

$$\rho(q \mid Q) := \max\left\{\frac{\mathbb{P}(T_q|Q)}{\mathbb{P}(T_q^{\complement}|Q)}, \frac{\mathbb{P}(T_q^{\complement}|Q)}{\mathbb{P}(T_q|Q)}\right\}.$$

Then, we can get that

$$\begin{aligned} &\min\{f(q \mid Q), g(q \mid Q)\} \\ &\geq \frac{(1-2e^*)^2}{e^*(1-e^*)\left(\frac{1-e^*}{e^*} + \rho(q \mid Q)\right)\left(\frac{e^*}{1-e^*} + \rho(q \mid Q)\right)} + 1 \triangleq h(q \mid Q). \end{aligned}$$

As a result, we have

$$\max_{q \in V}\{\min\{f(q \mid Q), g(q \mid Q)\}\} \geq \max_{q \in V}\{h(q \mid Q)\}.$$

Next, we show a lower bound on the right hand side of the above inequality. In particular, it is trivial to see that the solution to $\min_{q \in V}\{\rho(q \mid Q)\}$ is actually the solution to $\max_{q \in V}\{h(q \mid Q)\}$. We give an upper bound on the optimum of the former.

Lemma 1. *Given a tree $T = (V, E)$ where each node v is associated with a weight $p(v)$ such that $\sum_{v \in V} p(v) = 1$. For any subset S of V, let $p(S) = \sum_{v \in S} p(v)$ denote the total weight of the nodes in S. Then,*

$$\min_{v \in V}\left\{\max\left\{\frac{p(T_v)}{1-p(T_v)}, \frac{1-p(T_v)}{p(T_v)}\right\}\right\} \leq \frac{d+p^*}{1-p^*},$$

where T_v is the subtree of T rooted at v, d is the maximum degree of the nodes in T and $p^* = \max_v p(v)$ is the maximum weight in T.

Proof. It is trivial to see that the node u with the minimum value of $\max\left\{\frac{p(T_u)}{1-p(T_u)}, \frac{1-p(T_u)}{p(T_u)}\right\}$ also minimizes $\max\{p(T_u), 1-p(T_u)\}$ among all nodes in V, i.e., for any $v \in V$,

$$\max\{p(T_u), 1-p(T_u)\} \leq \max\{p(T_v), 1-p(T_v)\}, \tag{15}$$

or equivalently

$$\min\{p(T_u), 1-p(T_u)\} \geq \min\{p(T_v), 1-p(T_v)\}. \tag{16}$$

We first show that $\frac{p(T_u)}{1-p(T_u)} \leq \frac{d+p^*}{1-p^*}$. Without loss of generality, for every child v of u, we assume that $p(T_v) < p(T_u)$ (otherwise we can replace u by v as $p(T_v) = p(T_u)$). We note that $p(T_v) < 0.5$ (otherwise $0.5 \leq p(T_v) < p(T_u)$, contradicting (15)). Then, for every child v of u, we have

$$p(T_v) < \max\{p(T_u), 1-p(T_u)\} \leq 1-p(T_v),$$

where the second inequality is by (15). Hence, by (16), we have

$$p(T_v) \le \min\{p(T_u), 1 - p(T_u)\}.$$

As a result,

$$p(T_u) = p(u) + \sum_{v \in child(u)} p(T_v) \le p(u) + d(1 - p(T_u)).$$

Thus,

$$p(T_u) \le \frac{p(u) + d}{d+1}, \text{ and } 1 - p(T_u) \ge \frac{1 - p(u)}{d+1}.$$

Putting it together gives rise to

$$\frac{p(T_u)}{1 - p(T_u)} \le \frac{d + p(u)}{1 - p(u)} \le \frac{d + p^*}{1 - p^*}.$$

Next, we show that $\frac{1-p(T_u)}{p(T_u)} \le \frac{d+p^*}{1-p^*}$. Denote by w the parent of u. Without loss of generality, we also assume that $p(T_w) > p(T_u)$ (otherwise we can replace u by w as $p(T_w) = p(T_u)$). Similarly, by (15) and (16), we have

$$\begin{aligned} &1 - p(T_w) < \max\{p(T_u), 1 - p(T_u)\} \le p(T_w), \\ &1 - p(T_w) \le \min\{p(T_u), 1 - p(T_u)\}. \end{aligned} \tag{17}$$

Moreover, for any child v of w satisfying $v \neq u$, we show that $p(T_v) \le p(T_u)$ unless $p(T_v)+p(T_u) = 1$. Note that if $p(T_v)+p(T_u) = 1$ and $p(T_v) > p(T_u)$, then we can simply replace u by v. Suppose by contradiction that there exists a child v of w such that $p(T_v) > p(T_u)$ and $p(T_v) + p(T_u) < 1$. We note that $p(T_v) > 0.5$ (otherwise $0.5 \ge p(T_v) > p(T_u)$, contradicting (15)). Then, by (15), we must have $1 - p(T_u) \le p(T_v)$, which contradicts $p(T_u) + p(T_v) < 1$. Consequently, combining (17),

$$1-p(T_u) = 1-p(T_w)+p(w)+\sum_{v \in child(w)} p(T_v)-p(T_u) \le p(w)+dp(T_u).$$

Hence,

$$p(T_u) \ge \frac{1 - p(w)}{d+1}, \text{ and } 1 - p(T_u) \le \frac{d + p(w)}{d+1}.$$

Putting it together gives rise to

$$\frac{1 - p(T_u)}{p(T_u)} \le \frac{d + p(w)}{1 - p(w)} \le \frac{d + p^*}{1 - p^*}.$$

This completes the proof. □

Now, we are ready to show that our greedy strategy selects a query that can increase the expected posterior probability of the correct target z by a multiplicative factor of at least α, where

$$\alpha := \frac{(1 - 2e^*)^2}{e^*(1 - e^*)\left(\frac{1-2e^*}{e^*} + \frac{d+1}{\varepsilon}\right)\left(\frac{2e^*-1}{1-e^*} + \frac{d+1}{\varepsilon}\right)} + 1. \tag{18}$$

Theorem 2. *The greedy strategy in Algorithm 1 selects a query q on top of $\mathcal{Q}$ ensuring that*

$$\mathbb{E}[\mathbb{P}(z \mid \mathcal{Q} \cup \{(q, X)\}) \mid \mathcal{Q}] \ge \alpha \cdot \mathbb{P}(z \mid \mathcal{Q}), \tag{19}$$

where α is given in (18).

Proof. Recall that the greedy strategy selects a query q on top of $\mathcal{Q}$ ensuring that

$$\mathbb{E}[\lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}] \ge \max_{q' \in V}\{\min\{f(q' \mid \mathcal{Q}), g(q' \mid \mathcal{Q})\}\},$$

According to the stopping rule of Algorithm 1, we know that $p^* < 1 - \varepsilon$ before it terminates. Using the result of Lemma 1, we have

$$\min_{q' \in V}\{\rho(q' \mid \mathcal{Q})\} \le \frac{d + 1 - \varepsilon}{\varepsilon}.$$

Therefore, we can get that

$$\begin{aligned} &\max_{q' \in V}\{\min\{f(q' \mid \mathcal{Q}), g(q' \mid \mathcal{Q})\}\} \ge \max_{q' \in V}\{h(q' \mid \mathcal{Q})\} \\ &\ge \frac{(1 - 2e^*)^2}{e^*(1 - e^*)\left(\frac{1-e^*}{e^*} + \frac{d+1-\varepsilon}{\varepsilon}\right)\left(\frac{e^*}{1-e^*} + \frac{d+1-\varepsilon}{\varepsilon}\right)} + 1 = \alpha. \end{aligned}$$

This completes the proof. □

Theorem 2 states that the query q selected by Algorithm 1 satisfies $\mathbb{E}[\lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}] \ge \alpha$. Note that the stopping time of Algorithm 1 (i.e., the number of queries) is uncertain. In the following, we build a key relation for characterizing $\lambda(q, X; z, \mathcal{Q})$ based on Theorem 2 that is useful for deriving the query complexity of our algorithm.

Lemma 2. *The query q selected on top of $\mathcal{Q}$ by the greedy strategy of Algorithm 1 satisfies that*

$$2\mathbb{E}[\ln \lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}] \ge \ln \alpha. \tag{20}$$

Proof. In what follows, we will show that for any query q on top of $\mathcal{Q}$, it holds that

$$2\mathbb{E}[\ln \lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}] \ge \ln \mathbb{E}[\lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}].$$

Combining it with Theorem 2 concludes the lemma.

Indeed, by definition we have

$$\begin{aligned} &2\mathbb{E}[\ln \lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}] \\ &= \ln\left(\left(\frac{1 - x}{(1-x)y + x(1-y)}\right)^{2(1-x)}\left(\frac{x}{xy + (1-x)(1-y)}\right)^{2x}\right), \\ &\ln \mathbb{E}[\lambda(q, X; z, \mathcal{Q}) \mid \mathcal{Q}] \\ &= \ln\left(\frac{(1-x)^2}{(1-x)y + x(1-y)} + \frac{x^2}{xy + (1-x)(1-y)}\right), \end{aligned}$$

where $x = e_q$ and $y = \mathbb{P}(T_q \mid \mathcal{Q})$ (resp. $y = \mathbb{P}(T_q^{\complement} \mid \mathcal{Q})$) if $z \in T_q$ (resp. $z \in T_q^{\complement}$). Now, let

$$\begin{aligned} F(x, y) &:= \frac{\left(\frac{1-x}{(1-x)y+x(1-y)}\right)^{2(1-x)}\left(\frac{x}{xy+(1-x)(1-y)}\right)^{2x}}{\frac{(1-x)^2}{(1-x)y+x(1-y)} + \frac{x^2}{xy+(1-x)(1-y)}} \\ &= \frac{(1-x)^2\left(\frac{x}{1-x}\right)^{2x}\left(\frac{s}{1-s}\right)^{2x-1}}{(1-x)^2 - (1-2x)s}, \end{aligned}$$

where $s = x + (1 - 2x)y \in [0, 1]$. It thus suffices to prove that $F(x, y) \ge 1$ for any $x, y \in [0, 1]$. Taking the partial derivative of $F(x, y)$ with respect to y yields

$$\begin{aligned} \frac{\partial F(x, y)}{\partial y} &= \frac{(1-x)^2\left(\frac{x}{1-x}\right)^{2x}}{\left((1-x)^2 - (1-2x)s\right)^2} \cdot \frac{(1-2x)^2\left(\frac{s}{1-s}\right)^{2x-2}}{(1-s)^2} \\ &\quad \cdot\Big(-(1-x)^2 + (1-2x)s + (1-s)s\Big) \\ &= -\frac{(1-x)^2(1-2x)^2(1-x-s)^2\left(\frac{xs}{(1-x)(1-s)}\right)^{2x}}{\left((1-x)^2 - (1-2x)s\right)^2 s^2} \le 0. \end{aligned}$$

Therefore, $F(x, y) \ge F(x, 1) = 1$, which completes the proof. □

Inspired by Wald's equation [39] that simplifies the calculation of the expected value of the cumulative sum of a random number of random quantities, we establish upper bounds on the stopping time of Algorithm 1 on the basis of Lemma 2.

Theorem 3. *For the OIGS problem, given that the target node is z, Algorithm 1 asks at most $2\log_\alpha(\frac{1}{\mathbb{P}(z)})$ queries in expectation. Moreover, the expected number of queries involved in Algorithm 1 is at most $2\log_\alpha(n)$, where the expectation is taken over all randomness, including the randomness of z following the prior probability distribution $\mathbb{P}(z)$.*

Proof. Let Q_i be the information collected after i rounds and $Y_i := \lambda(q, X; z, Q_{i-1})$ be a random variable indicating the multiplier in the i-th round. Let τ_z be the stopping time of Algorithm 1 given that the target node is z. Then,

$$\mathbb{P}(z \mid Q_{\tau_z}) = \mathbb{P}(z) \cdot \prod_{i=1}^{\tau_z} Y_i.$$

Thus,

$$\sum_{i=1}^{\tau_z} \ln Y_i = \ln\left(\frac{\mathbb{P}(z|Q_{\tau_z})}{\mathbb{P}(z)}\right) \le \ln\left(\frac{1}{\mathbb{P}(z)}\right). \tag{21}$$

Meanwhile, denote by $\mathbb{1}_{\{\tau_z \ge i\}}$ the indicator random variable for the event $\{\tau_z \ge i\}$, i.e., $\mathbb{1}_{\{\tau_z \ge i\}} = 1$ if $\tau_z \ge i$ and otherwise $\mathbb{1}_{\{\tau_z \ge i\}} = 0$. Then, we have

$$\sum_{i=1}^{\tau_z} \ln Y_i = \sum_{i=1}^{\infty} \left(\ln Y_i \cdot \mathbb{1}_{\{\tau_z \ge i\}}\right).$$

Taking the expectation for both sides with respect to Y_i's and τ_z yields

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{\tau_z} \ln Y_i\right] &= \sum_{i=1}^{\infty} \mathbb{E}\left[\ln Y_i \cdot \mathbb{1}_{\{\tau_z \ge i\}}\right] \\
&= \sum_{i=1}^{\infty} \mathbb{E}\left[\mathbb{E}\left[\ln Y_i \cdot \mathbb{1}_{\{\tau_z \ge i\}} \mid Q_{i-1}\right]\right] \\
&= \sum_{i=1}^{\infty} \mathbb{E}\left[\mathbb{E}\left[\ln Y_i \mid Q_{i-1}, \mathbb{1}_{\{\tau_z \ge i\}}\right] \cdot \mathbb{1}_{\{\tau_z \ge i\}}\right] \\
&\ge \sum_{i=1}^{\infty} \mathbb{E}\left[\tfrac{\ln \alpha}{2} \cdot \mathbb{1}_{\{\tau_z \ge i\}}\right] \\
&= \tfrac{\ln \alpha}{2} \cdot \sum_{i=1}^{\infty} \mathbb{P}(\tau_z \ge i) \\
&= \tfrac{\ln \alpha}{2} \cdot \mathbb{E}\left[\tau_z\right],
\end{aligned}$$

where the inequality is by Lemma 2. Combining it with (21) immediately yields the first part that

$$\mathbb{E}\left[\tau_z\right] \le 2\log_\alpha(\tfrac{1}{\mathbb{P}(z)}).$$

In addition, taking the expectation over the above inequality with respect to the randomness of z yields

$$\mathbb{E}\left[\mathbb{E}[\tau_z]\right] \le \mathbb{E}\left[2\log_\alpha(\tfrac{1}{\mathbb{P}(z)})\right] \le 2\log_\alpha\left(\mathbb{E}\left[\tfrac{1}{\mathbb{P}(z)}\right]\right) = 2\log_\alpha(n),$$

where the second inequality is by Jensen's inequality [17] as $\log_\alpha(\cdot)$ is a concave function. This completes the proof. □

We can infer from Theorem 3 that it is likely to require more queries to locate the target when (i) the failure threshold, i.e., ε, is stringent, (ii) the noise level, i.e., e^*, is large, (iii) the hierarchy has some bad structure, e.g., the maximum degree d is large, and/or (iv) a rare object occurs, i.e., the prior probability $\mathbb{P}(z)$ of the target z is low.

Interestingly, the expected number of queries (over the randomness of the target) is near-optimal up to a multiplicative factor of $2/\log_2(\alpha)$, as the lower bound is $\log_2(n)$. This lower bound can be seen by considering a special case of the OIGS problem in which the tree is just an ordered list and every error probability is zero [19].

5 EXPERIMENT

In this section, we conduct experiments to evaluate the performance of our method. The key metrics for evaluating an algorithm are its *cost* and *accuracy*. All the experiments are carried out on a machine with an Intel i7-7700 CPU and 32GB RAM. All the algorithms are implemented in Python.

5.1 Experiment Setting

Datasets. Following the previous work [8, 23, 35], we use the same two real-world datasets considered by them in our experiments:

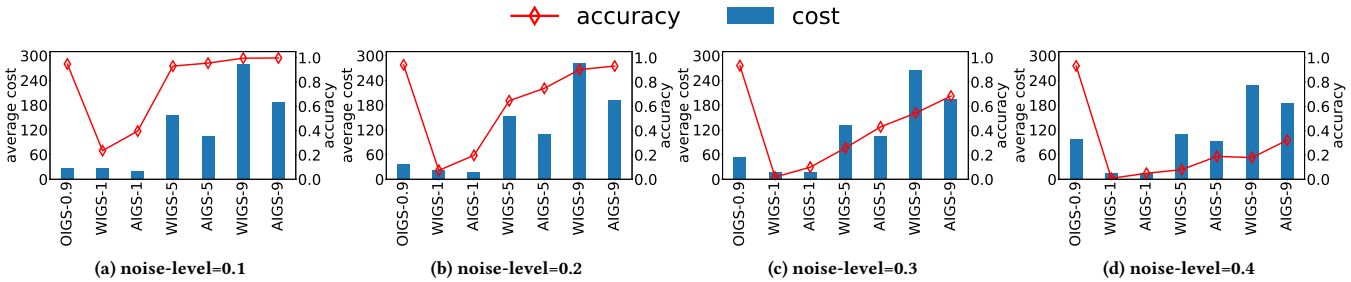
- Amazon[1] [14]. This dataset includes 13,886,889 products sold at Amazon. The hierarchy of the products has a tree structure with 29,240 nodes.
- ImageNet[2] [10]. This is a large-scale image dataset using the structure of WordNet [25], consisting of 12,656,970 images. The hierarchy of this dataset is a directed acyclic graph with 27,714 nodes. Following previous work [47], we extract a spanning tree from the original input hierarchy.

We count the number of products/images for each node in the hierarchy (i.e., label) and normalize it by the total amount as its prior probability. We randomly select 10,000 products/images as the test data to be labeled.

Metrics. The main metrics here are *cost*, i.e., the expected number of queries, and *accuracy*, i.e., the proportion of objects that are correctly labeled.

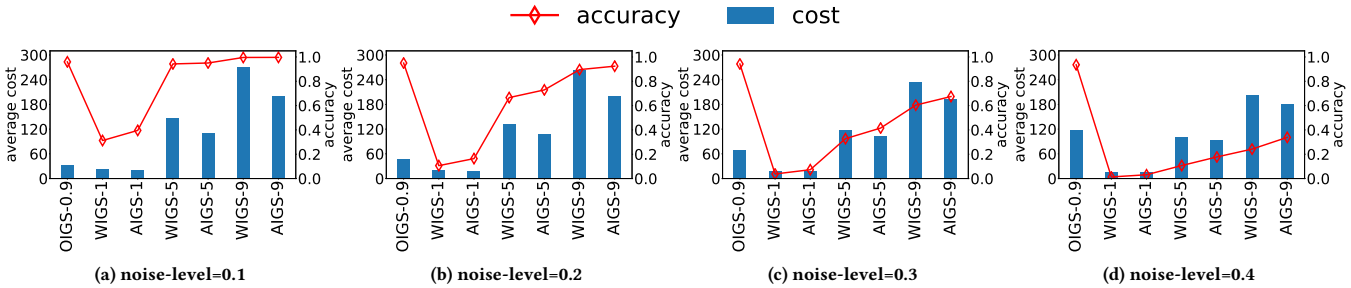
Competing Algorithms. We consider two baselines for the variants of IGS, including the heavy-path-based binary search method proposed by Tao et al. [35], referred to as WIGS, and the greedy-based method proposed by Cong et al. [8], referred to as AIGS. We note that both WIGS and AIGS do not consider query noise, so they will return a wrong result if any query receives a wrong answer during the search progress. To adapt them to the OIGS problem, we incorporate the majority voting technique widely adopted in crowdsouring platforms [35, 46] into these algorithms. That is, we ask each query multiple times and take the most frequent answer as the answer elected for this query. We refer to the resultant algorithms as WIGS-x and AIGS-x, where x is the number of repeated times of each query. For example, the WIGS-5 algorithm selects the same query nodes as WIGS, and sends each query to 5 workers and accepts the majority answer (i.e., from at least 3 workers).

[1] https://jmcauley.ucsd.edu/data/amazon/links.html
[2] https://www.image-net.org/api/xml/structure_released.xml



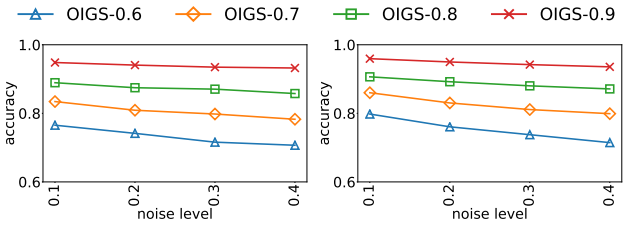
(a) noise-level=0.1 (b) noise-level=0.2 (c) noise-level=0.3 (d) noise-level=0.4

Figure 2: Performance on the Amazon Dataset



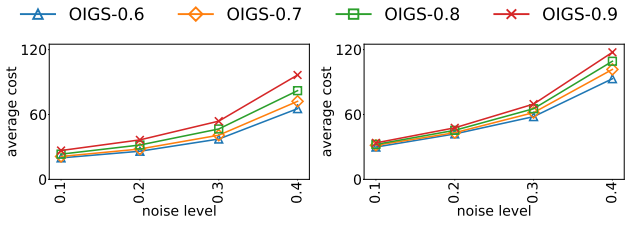
(a) noise-level=0.1 (b) noise-level=0.2 (c) noise-level=0.3 (d) noise-level=0.4

Figure 3: Performance on the ImageNet Dataset



(a) Amazon (b) ImageNet

Figure 4: Accuracy of OIGS with Different Threshold



(a) Amazon (b) ImageNet

Figure 5: Cost of OIGS with Different Threshold

For our OIGS algorithm, the threshold of failure probability ε introduces a tradeoff between cost and accuracy. That is, decreasing ε can improve accuracy but incur a higher cost. For ease of reference, we denote OIGS-y as OIGS with a success probability of at least $y = 1 - \varepsilon$. For example, OIGS-0.9 will stop as long as the maximum posterior probability is no less than 0.9.

Parameter Setting. For the noise setting, we consider four different noise levels, i.e., $e_{max} = 0.1, 0.2, 0.3, 0.4$, where the error rate of each node is generated by a random number uniformly distributed in the range of $[0, e_{max}]$. By default, we set the algorithmic parameter $\varepsilon = 0.1$ for our algorithm, i.e., OIGS-0.9. We shall carry out an experiment to test the effect of ε. For the baseline WIGS-x and AIGS-x algorithms, we set three different values of x, i.e., $x = 1, 5, 9$.

5.2 Experiment Result

Figure 2 shows the average cost and accuracy on the Amazon dataset. As can be seen from Figure 2(a), when the noise level e_{max} is as low as 0.1, our OIGS-0.9 algorithm achieves an accuracy of 0.948 using a cost of 26.68, whereas both WIGS and AIGS either return very poor results in terms of accuracy (i.e., less than 0.25 and 0.4 by WIGS-1 and AIGS-1, respectively) or incur significant overhead in terms of cost with comparable accuracy (e.g., 5.86 and 3.96 times higher costs by WIGS-5 and AIGS-5, respectively). Figures 2(b)–2(d) show that when the query responses become more noisy, to ensure the accuracy, our OIGS-0.9 needs to ask more questions but still notably fewer than WIGS-5/9 and AIGS-5/9. Meanwhile, we observe that the accuracy of WIGS-1/5/9 and AIGS-1/5/9 drops quickly when the noise level increases. In particular, when the noise level is 0.4 as shown in Figure 2(d), the accuracy of WIGS-9 and AIGS-9 is less than 0.2 and 0.4, respectively, not to mention WIGS-1/5 and AIGS-1/5. The results on the ImageNet dataset, as shown in Figure 3, are quite similar. These experiment results demonstrate the superiority of our OIGS method, since OIGS only asks a small number of questions to identify the label for each object while consistently guaranteeing high accuracy, no matter how noisy the crowdsourcing platform is (which is unachievable by the baselines).

Figure 4 show the accuracy of our OIGS method under different threshold settings of $1 - \varepsilon = 0.6, 0.7, 0.8, 0.9$. The accuracy of our algorithm, although exceeding the corresponding threshold setting of $1 - \varepsilon$, deteriorates when higher failure probability ε is allowed. We also observe that when the threshold $1 - \varepsilon$ is low, the accuracy

of our algorithm, e.g., OIGS-0.6, decreases more quickly. To explain, the label returned by OIGS-0.6 is more uncertain than that by OIGS-0.9, and thus noise is more harmful to OIGS-0.6 than to OIGS-0.9. Figure 5 shows the average cost of our OIGS algorithm when the threshold $1-\varepsilon$ varies from 0.6 to 0.9. It can be seen that more queries are needed with the increasing of (i) threshold $1-\varepsilon$ and/or (ii) noise level e_{max}.

6 RELATED WORK

Interactive Graph Search. Interactive graph search (IGS) was firstly proposed by Tao et al. [35]. Prior to it, Parameswaran et al. [31] studied an offline version of IGS that proposes all queries in one round without any interactions. Recently, several variants of IGS have been studied in the literature [8, 23, 47]. Li et al. [23] aimed to locate the target node using fewest rounds of multiple-choice queries. Zhu et al. [47] attempted to find multiple targets subject to a budget constraint on the number of queries allowed. Cong et al. [8] studied an average-case IGS problem that minimizes the expected cost. To our knowledge, all these existing proposals assumed that the queries are answered by a perfect oracle without making errors, which is unrealistic as the answers are usually noisy in a practical crowdsourcing platform. In this paper, we investigate the noisy variant, namely OIGS, that minimizes the query complexity while ensuring accuracy.

Poset and Decision Tree. From the theoretical perspective, our work is highly related to the search in partially-ordered set (poset) problem [6] and the decision tree problem [4]. Cong et al. [8] showed that IGS can be mapped to the aforementioned two problems. In particular, for the worst-case IGS problem [35], the optimal policy can be constructed in linear time if the given hierarchy has a tree structure [3, 27, 29], while there exists an $\frac{O(\log(n))}{O(\log\log(n))}$-approximate algorithm if the input hierarchy is a directed acyclic graph (DAG) [11]. For the average-case IGS problem [8], the simple greedy policy has an approximate ratio of $\frac{1+\sqrt{5}}{2}$ and there exists a fully polynomial time approximation scheme based on dynamic programming on a tree hierarchy [7]; while on a DAG hierarchy, the rounded greedy algorithm can achieve an approximation ratio of $O(\log(n))$ [4] and no ratios better than $o(\log(n))$ can be achieved unless P = NP [6]. Again, noise is not considered in these results. Nowak [28] studied the noisy decision tree problem in the domain of machine learning. However, this work did not provide a solution to achieve a predefined accuracy and their theoretical guarantee requires that the input has a special structure (called as neighborhood) which is not satisfied in our problem. Therefore, it is unclear whether their algorithm can be applied to our OIGS problem with strong theoretical guarantees.

Human-Based Computation. Human-based computation has been studied for decades to address the problems that are hard to solve totally algorithmically, such as building hierarchy [5, 34], entity resolution [37, 38, 40, 42], object categorization [23, 47], data filtering [30, 32], data labeling [10], SQL-like query processing [9, 18, 36], and data cleaning [44]. Problems in human-based computation have attracted considerable attention in the database and data mining areas [2, 15, 20, 22, 24, 41, 48]. Among the above problems, the crowd-based filtering problem [30, 32] is closest to our work, which aims to filter objects based on a set of properties with the minimum cost while ensuring accuracy. The key difference is that they categorize all the objects into two classes only (i.e., a total of two labels) and thus their developments cannot be applied to our general scenario where the object categories form a tree hierarchy.

7 CONCLUSION

In this paper, we have studied the noisy interactive graph search problem and proposed a cost-effective algorithm with provable theoretical guarantees. Our algorithm is expected to stop in $2\log_{\alpha}(n)$ steps to identify a target node with high accuracy, where n is the number of nodes in the categorization hierarchy and α is a parameter related to the accuracy requirement, the noise level, and the structure of the hierarchy. Using extensive experiments on two real datasets, we have shown that our approach can outperform the state-of-the-art algorithms.

ACKNOWLEDGMENTS

Jing Tang's work is partially supported by HKUST(GZ) under a Startup Grant. Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants. Kai Han's work is partially supported by National Natural Science Foundation of China (NSFC) under Grant No. 62172384 and by Alibaba Group through Alibaba Innovative Research Program.

REFERENCES

[1] Yael Amsterdamer, Yael Grossman, Tova Milo, and Pierre Senellart. 2013. Crowd Mining. In *Proc. ACM SIGMOD*. 241–252.
[2] Yukino Baba and Hisashi Kashima. 2013. Statistical Quality Estimation for General Crowdsourcing Tasks. In *Proc. ACM SIGKDD*. 554–562.
[3] Yosi Ben-Asher, Eitan Farchi, and Ilan Newman. 1999. Optimal Search in Trees. *SIAM J. Comput.* 28, 6 (1999), 2090–2102.
[4] Venkatesan T Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh Mohania. 2007. Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results. In *Proc. PODS*. 53–62.
[5] Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. 2013. Cascade: Crowdsourcing Taxonomy Creation. In *Proc. ACM SIGCHI*. 1999–2008.
[6] Ferdinando Cicalese, Tobias Jacobs, Eduardo Laber, and Marco Molinaro. 2011. On the Complexity of Searching in Trees and Partially Ordered Structures. *Theoretical Computer Science* 412, 50 (2011), 6879–6896.
[7] Ferdinando Cicalese, Tobias Jacobs, Eduardo Laber, and Marco Molinaro. 2014. Improved Approximation Algorithms for the Average-Case Tree Searching Problem. *Algorithmica* 68, 4 (2014), 1045–1074.
[8] Qianhao Cong, Jing Tang, Yuming Huang, Lei Chen, and Yeow Meng Chee. 2022. Cost-Effective Algorithms for Average-Case Interactive Graph Search. arXiv:2201.07944 [cs.DB]
[9] Susan B Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. 2013. Using the Crowd for Top-k and Group-by Queries. In *Proc. ICDT*. 225–236.
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A Large-scale Hierarchical Image Database. In *Proc. IEEE CVPR*. 248–255.
[11] Dariusz Dereniowski. 2008. Edge Ranking and Searching in Partial Orders. *Discrete Applied Mathematics* 156, 13 (2008), 2493–2500.
[12] Jing Gao, Qi Li, Bo Zhao, Wei Fan, and Jiawei Han. 2016. Mining Reliable Information from Passively and Actively Crowdsourced Data. In *Proc. ACM SIGKDD*. 2121–2122.

[13] Stephen Guo, Aditya Parameswaran, and Hector Garcia-Molina. 2012. So Who Won? Dynamic Max Discovery With the Crowd. In *Proc. ACM SIGMOD*. 385–396.

[14] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-class Collaborative Filtering. In *Proc. WWW*. 507–517.

[15] Mengdi Huai, Di Wang, Chenglin Miao, Jinhui Xu, and Aidong Zhang. 2019. Privacy-aware Synthesizing for Crowdsourced Data. In *Proc IJCAI*. 2542–2548.

[16] Chao Huang, Xian Wu, and Dong Wang. 2016. Crowdsourcing-based Urban Anomaly Prediction System for Smart Cities. In *Proc ACM CIKM*. 1969–1972.

[17] Johan Jensen. 1906. Sur les Fonctions Convexes et les Inégalités entre les Valeurs Moyennes. *Acta Mathematica* 30, 1 (1906), 175–193.

[18] Adam Marcus Eugene Wu David Karger and Samuel Madden Robert Miller. 2011. Human-powered Sorts and Joins. *Proc. VLDB Endowment* 5, 1 (2011), 13–24.

[19] Donald E. Knuth. 1998. *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*.

[20] Shao-Yuan Li, Yuan Jiang, Nitesh V Chawla, and Zhi-Hua Zhou. 2018. Multi-label Learning from Crowds. *IEEE Transactions on Knowledge and Data Engineering* 31, 7 (2018), 1369–1382.

[21] Xinke Li, Chongshou Li, Zekun Tong, Andrew Lim, Junsong Yuan, Yuwei Wu, Jing Tang, and Raymond Huang. 2020. Campus3d: A Photogrammetry Point Cloud Benchmark for Hierarchical Understanding of Outdoor Scene. In *Proc. ACM MM*. 238–246.

[22] Yanying Li, Haipei Sun, and Wendy Hui Wang. 2020. Towards Fair Truth Discovery from Biased Crowdsourced Answers. In *Proc. ACM SIGKDD*. 599–607.

[23] Yuanbing Li, Xian Wu, Yifei Jin, Jian Li, and Guoliang Li. 2020. Efficient Algorithms for Crowd-Aided Categorization. *Proc. VLDB Endowment* 13, 8 (2020), 1221–1233.

[24] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. 2015. Faitcrowd: Fine Grained Truth Discovery for Crowdsourced Data Aggregation. In *Proc. ACM SIGKDD*. 745–754.

[25] George A Miller. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

[26] Kaixiang Mo, Erheng Zhong, and Qiang Yang. 2013. Cross-task Crowdsourcing. In *Proc. ACM SIGKDD*. 677–685.

[27] Shay Mozes, Krzysztof Onak, and Oren Weimann. 2008. Finding an Optimal Tree Searching Strategy in Linear Time. In *Proc. SODA*. 1096–1105.

[28] Robert Nowak. 2009. Noisy Generalized Binary Search. In *Proc. NeurIPS*. 1366–1374.

[29] Krzysztof Onak and Pawel Parys. 2006. Generalization of Binary Search: Searching in Trees and Forest-like Partial Orders. In *Proc. IEEE FOCS*. 379–388.

[30] Aditya Parameswaran, Stephen Boyd, Hector Garcia-Molina, Ashish Gupta, Neoklis Polyzotis, and Jennifer Widom. 2014. Optimal Crowd-Powered Rating and Filtering Algorithms. *Proc. VLDB Endowment* 7, 9 (2014), 685–696.

[31] Aditya Parameswaran, Anish Das Sarma, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. 2011. Human-Assisted Graph Search: It's Okay to Ask Questions. *Proc. VLDB Endowment* 4, 5 (2011), 267–278.

[32] Aditya G Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. 2012. Crowdscreen: Algorithms for Filtering Data with Humans. In *Proc. ACM SIGMOD*. 361–372.

[33] Hesam Salehian, Patrick Howell, and Chul Lee. 2017. Matching Restaurant Menus to Crowdsourced Food Data: A Scalable Machine Learning Approach. In *Proc. ACM SIGKDD*. 2001–2009.

[34] Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. 2015. Building hierarchies of concepts via crowdsourcing. In *Proc. IJCAI*. 844–853.

[35] Yufei Tao, Yuanbing Li, and Guoliang Li. 2019. Interactive Graph Search. In *Proc. ACM SIGMOD*. 1393–1410.

[36] Petros Venetis, Hector Garcia-Molina, Kerui Huang, and Neoklis Polyzotis. 2012. Max Algorithms in Crowdsourcing Environments. In *Proc. WWW*. 989–998.

[37] Vasilis Verroios and Hector Garcia-Molina. 2015. Entity Resolution with Crowd Rrrors. In *Proc. IEEE ICDE*. 219–230.

[38] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. 2014. Crowdsourcing Algorithms for Entity Resolution. *Proc. VLDB Endowment* 7, 12 (2014), 1071–1082.

[39] Abraham Wald. 1947. *Sequential Analysis*. Wiley.

[40] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *Proc. VLDB Endowment* 5, 11 (2012), 1483–1494.

[41] Yue Wang, Ke Wang, and Chunyan Miao. 2020. Truth Discovery Against Strategic Sybil Attack in Crowdsourcing. In *Proc. ACM SIGKDD*. 95–104.

[42] Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. 2013. Question Selection for Crowd Entity Resolution. *Proc. VLDB Endowment* 6, 6 (2013), 349–360.

[43] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Advances in NeurIPS*, Vol. 22. 2035–2043.

[44] Chen Jason Zhang, Lei Chen, Yongxin Tong, and Zheng Liu. 2015. Cleaning Uncertain Data with a Noisy Crowd. In *Proc. IEEE ICDE*. 6–17.

[45] Jing Zhang and Xindong Wu. 2018. Multi-label Inference for Crowdsourcing. In *Proc. ACM SIGKDD*. 2738–2747.

[46] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth Inference in Crowdsourcing: Is the Problem Solved? *Proc. VLDB Endowment* 10, 5 (2017), 541–552.

[47] Xuliang Zhu, Xin Huang, Byron Choi, Jiaxin Jiang, Zhaonian Zou, and Jianliang Xu. 2021. Budget Constrained Interactive Search for Multiple Targets. *Proc. VLDB Endowment* 14, 6 (2021), 890–902.

[48] Honglei Zhuang, Aditya Parameswaran, Dan Roth, and Jiawei Han. 2015. Debiasing Crowdsourced Batches. In *Proc. ACM SIGKDD*. 1593–1602.