Learning-Based Algorithms for Graph Searching Problems

Adela Frances DePavia
University of Chicago
adepavia@uchicago.edu

Erasmo Tani
University of Chicago
etani@uchicago.edu

Ali Vakilian
TTIC
vakilian@ttic.edu

Abstract

We consider the problem of graph searching with prediction recently introduced by Banerjee et al. (2023). In this problem, an agent, starting at some vertex r has to traverse a (potentially unknown) graph G to find a hidden goal node g while minimizing the total distance travelled. We study a setting in which at any node v, the agent receives a noisy estimate of the distance from v to g. We design algorithms for this search task on unknown graphs. We establish the first formal guarantees on unknown weighted graphs and provide lower bounds showing that the algorithms we propose have optimal or nearly-optimal dependence on the prediction error. Further, we perform numerical experiments demonstrating that in addition to being robust to adversarial error, our algorithms perform well in typical instances in which the error is stochastic. Finally, we provide alternative simpler performance bounds on the algorithms of Banerjee et al. (2023) for the case of searching on a known graph, and establish new lower bounds for this setting.

1 INTRODUCTION

Searching on graphs is a fundamental problem which models many real-world applications in autonomous navigation. In a *graph searching problem* instance, an agent is initialized at some vertex $r \in V$ (referred to as the root) in some (potentially weighted, directed) graph $G = (V, E)$. The agent's task is to find a goal node $g \in V$. The agent searches for g by sequentially visiting adjacent nodes in the graph. The graph searching problem terminates when the agent reaches the goal node, and the cost incurred by the agent is the total amount of distance they travelled.

There are two main settings of interest. In the *exploration* setting, as the agent moves through the graph, it only learns the structure of G by observing the vertices and edges adjacent to the nodes it has visited, a model sometimes referred to as the *fixed graph scenario* (Komm et al., 2015; Kalyanasundaram and Pruhs, 1994). In the strictly-easier *planning* setting, the agent is given the entire graph G ahead of time, but does not know the identity of the goal node g.

Without additional information, in the worst-case the agent must resort to visiting the entire graph, a task which amounts to finding an efficient tour in an unknown graph[1](Berman, 2005; Dobrev et al., 2012; Megow et al., 2012; Eberle et al., 2022). Recently, Banerjee et al. (2023) consider the setting when an algorithm for graph searching also receives some prediction function, $f : V \to \mathbb{R}$, representing some (noisy) estimate of the distance to the goal node g at any given node $v \in V$. This setup models applications in which the searcher receives advice from some machine-learning model designed to predict the distance to the goal; this problem fits into the broader framework of learning-based algorithms, which exploit (potentially noisy) advice from a machine learning model to enhance their performance.

In the case of planning, Banerjee et al. (2023) propose an intuitive strategy that can be deployed in weighted and unweighted graphs and analyze its performance in terms of structural properties of the instance graph, such as its maximum-degree and its doubling-dimension. They establish formal guarantees on the cost incurred by their algorithm under different notions of prediction error. In contrast, their results in the exploration setting are limited: they propose an

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

[1]We note that historically the task of finding an efficient tour in an unknown graph has been referred to as graph exploration, but following the conventions of Banerjee et al. (2023) we reserve this name for the graph search problem on an unknown graph.

algorithm for exploration in unweighted trees. Their algorithm is tailored to this restricted class of graphs and their guarantees are parameterized by the number of incorrect predictions, which does not capture the magnitude of the deviation between predictions and true distances.

This paper seeks to expand the understanding of graph exploration problems with predictions. We design algorithms which can be deployed on a variety of weighted graphs and prove worst-case guarantees on their performance. We also complement this analysis by providing lower-bounds for these problems, showing that the algorithms studied in this work are optimal or nearly optimal. In particular, we focus on two different error models. In the absolute error model, the magnitude of the error at a node is independent of that node's true distance to the goal, and guarantees are given in terms of the total magnitude of the error incurred at every node. In the relative error model, nodes further from the goal may have larger deviation between the prediction and the truth, and algorithmic guarantees are parameterized by the maximum ratio of the error to the true distance at any vertex.

Related Works Online graph searching problems have long been used as basic models for problems in autonomous navigation Berman (2005). Searching with access to predictions, also referred to as advice or heuristics in different communities, is a commonly studied variant (Pelc, 2002; Dobrev et al., 2012; Eberle et al., 2022; Banerjee et al., 2023). The problem and prediction settings considered in this work most closely correspond to those considered by Banerjee et al. (2023). This setup models applications in which predictions are the output of some machine-learning model. Recent years have seen a marked increase in the integration of machine learning techniques to enhance traditional algorithmic challenges (Angelopoulos et al., 2020; Gupta et al., 2022; Mitzenmacher and Vassilvitskii, 2022; Antoniadis et al., 2023). More generic forms of advice and the advice-complexity of exploration tasks are long-standing subjects of study. Komm et al. (2015) study the case when the the searcher receives generic advice, which can take the form of any bit string, and prove results about the advice complexity of this task. For a more detailed survey of related works, we direct the reader to Section A.

Organization In Section 1.1 we formally state the main results of the paper. In Section 2 we present technical preliminaries and define relevant notation. Sections 3 and 4 contain algorithms and analysis for exploration under absolute and relative error models respectively. In Section 5 we derive new bounds in the planning setting via metric embeddings. Finally, in Section 6 we complement these results with numerical experiments. All missing proofs are in Section B of the supplementary material.

1.1 Summary Of Results

We begin by formally describing the exploration and planning settings:

The Exploration Problem In this setting, both the graph G and the predictions f are initially unknown to the agent: the agent is initialized with access to the root node r, the neighbors of r, and the predictions at all of these nodes. As the algorithm proceeds, on each iteration i it has access in memory to a subgraph $G_i \subseteq G$ containing the nodes it has visited, the neighbors of those nodes, and any edges between visited nodes and neighbors. The searcher can only query predictions from nodes in the subgraph G_i. This problem models exploration of an unknown environment.

The Planning Problem In this setting, both the graph G and the predictions f are fully known to the searcher upon initialization.

For any algorithm which visits an ordered sequence of vertices $v_1, \ldots, v_T$, we denote the algorithmic cost $\mathrm{ALG} \stackrel{\text{def}}{=} \sum_{i \in T} d_{G_i}(v_i, v_{i+1})$, where $d_{G_i}(\cdot, \cdot)$ denotes the shortest (weighted) path distance in the graph G_i described above. The guarantees in this paper compare ALG to the optimal cost a posteriori, denoted by $\mathrm{OPT} \stackrel{\text{def}}{=} d_G(r, g)$.

Exploration Under Absolute Error A natural way of measuring the error of some predictions is the magnitude of the difference between the true distance-to-goal and prediction value at each node. In Section 3 we propose an algorithm for the exploration problem on weighted graphs and prove performance guarantees parameterized by these error measures. In particular, we prove the following theorem:

Theorem 1. There is an algorithm (Algorithm 1) for searching arbitrary (potentially directed) graphs which finds the goal g by traveling a distance of at most $\mathrm{OPT} + E_1 + n\,E_1^+$, where $E_1 \stackrel{\text{def}}{=} \sum_{v \in V} \max\{0, d(v, g) - f(v)\}$ and $E_1^+ \stackrel{\text{def}}{=} \max_{v \in V} \max\{0, f(v) - d(v, g)\}$.

This algorithm enjoys several advantages over the most recent results on the exploration problem by Banerjee et al. (2023). Their work introduces an involved combinatorial algorithm for exploration on unweighted trees whose performance is parametrized by the ℓ_0 norm

(the number of non-zero entries) of the vector of errors. The guarantees of their algorithm do not apply when the graph being searched is not an unweighted tree. In contrast, the algorithm proposed in the present work is intuitive and easily implementable, and the guarantees obtained hold in a wide variety of settings, e.g. when the graph is weighted and/or directed. The performance of the algorithm is parameterized by the natural absolute deviation (ℓ_1) error measure.

We also establish that under the above parameterization, the proposed algorithm is in some sense optimal (see Theorem 6). Further our analysis highlights that the same algorithm performs particularly well when (erroneous) predictions only underestimate distance to the true goal. Prediction functions with this property are referred to as admissible. They are key objects of study in path-finding literature and are well-motivated by applications (Dechter and Pearl, 1985; Eden et al., 2022; Ferguson et al., 2005; Pohl, 1969).

Relative Errors One realistic setting for applications is one in which the error is proportionate to the magnitude of the distance to the goal. In order to capture this behavior, we consider a different error model in which the ratio of the error to true distance-to-goal at every vertex is assumed to be bounded by some value $\varepsilon \in (0,1)$:

$$(1-\varepsilon)d(v,g) \le f(v) \le (1+\varepsilon)d(v,g). \qquad (1)$$

We do not place any restriction on the total amount of error in the graph beyond this condition.

We consider two regimes of multiplicative error. In the first setting, ε is assumed to be known to the searcher a priori. We propose an algorithm and show that it achieves the following competitive ratio:

Theorem 2. Consider the exploration problem on a weighted tree where predictions satisfy (1) with respect to $\varepsilon \in (0,1)$, and ε is known. Then there exists an algorithm (Algorithm 2) which succeeds in finding the goal g and incurs competitive ratio at most

$$\frac{ALG}{OPT} \le \frac{1}{1-\varepsilon} + n\varepsilon\,\frac{4}{(1-\varepsilon)^2}.$$

In particular, if the predictions are admissible, then the same algorithm incurs competitive ratio

$$\frac{ALG}{OPT} \le 1 + n\varepsilon\,\frac{2}{1-\varepsilon}.$$

In the second regime ε is assumed to be small ($\varepsilon < 1/3$) but its exact value is not assumed to be known. For this setting, we design a different algorithm which allows us to prove the following result:

Theorem 3. Given G a weighted tree with predictions f satisfying Equation (1) for some unknown $\varepsilon < 1/3$, then Algorithm 3 with $\alpha = 2/3$ incurs competitive ratio at most

$$\frac{ALG}{OPT} \le 2 + O\left(n\varepsilon\,\frac{5+3\varepsilon}{(1-3\varepsilon)^2}\right).$$

In Section 4, we describe our algorithms for these problems, prove the above theorems, and complement these algorithmic guarantees with lower bounds that show these algorithms are nearly optimal.

Planning Problems Banerjee et al. (2023) consider problems in which both the full graph and all predictions are available to the algorithm upon initialization. This setting is referred to as the planning problem. They construct algorithms for this version of the problem under different error models and the guarantees they obtained are outlined in Table 1. In many regimes (e.g. planning on unweighted trees under error parametrized by the ℓ_0-norm of the vector of errors, denoted E_0, and planning on graphs under error parametrized by the ℓ_1-norm of the vector of errors, denoted E_1) matching lower bounds for their algorithms can be established, as shown in the table. A notable exception is the case of planning on unweighted graphs under E_0 parametrization: they establish an upper bound of $OPT + 2^{O(\alpha)}O(E_0^2)$ where α is the doubling dimension of the graph. In particular, the lower bounds they provide fail to match the depedence on the quadratic term E_0^2 in their upper bound.

We provide an alternative analysis of their algorithm based on metric properties of the instance graph which shows that in some classes of graphs one can reduce the asymptotic dependence on E_0 from a quadratic to a linear factor. In particular, we consider the distortion of embedding the instance graph into a weighted path or cycle graph, and establish the following guarantee:

Theorem 4. Consider G an unweighted graph such that G admits an embedding into a weighted path or a weighted cycle of distortion at most τ. Then on G, the E_0 planning algorithm of Banerjee et al. (2023) incurs cost at most $OPT + O(\tau E_0)$.

We note that the results present in Table 1 which were not proved by Banerjee et al. (2023) are established in this paper in Section 5 and the proved in the supplementary material.

2 TECHNICAL PRELIMINARIES AND NOTATION

Graphs are assumed to be weighted and directed, unless otherwise specified. (Weighted) shortest-path dis-

		E_0, unweighted	E_1, positive weights
Trees with maximum degree	upper bound	OPT + O(E_0) („)	OPT + O(E_1^2) (integer distances)
	lower bound	max{ OPT; (E_0)} („)	max OPT; (E_1^2)
Graphs with doubling dimension	upper bound	OPT + $2^{O(\ )}$O(E_0^2) („)	OPT + $2^{O(\ )}$O(E_1) („)
	lower bound	Unknown	max{ OPT; (2 E_1)}
Graphs with path-embedding distortion	upper bound	OPT + O(E_0)	OPT + O(E_1)
	lower bound	Unknown	max{ OPT; (E_1)}

Table 1: Known results for planning problem in different settings. („) denotes results from Banerjee et al. (2023). denotes the maximum degree of any vertex in the graph, denotes the doubling-dimension of the graph, and denotes the distortion of the path-embedding on G. Banerjee et al. (2023) in their work note the absence of a matching lower bound in the graph planning problem. This gap motivates this work's study of planning bounds parameterized by metric embeddings, which yields a reduced asymptotic dependency on E_0. For full discussions of lower bounds for E_1 parametrized by , , and , see Lemmas 11 and 13, and Lemma 20 in Section B of the supplementary material.

tances in a graph G are denoted by $d_G(\ ;\ )$. Given a set $S \subseteq V$, let ∂S be its external vertex boundary:

$$\partial S \stackrel{\text{def}}{=} \{ v \in G \setminus S \mid \exists u \in S : v \sim u \}$$

For a vertex set $S \subseteq V$, we denote by $\text{tour}_G(S)$ the (weighted) length of the shortest walk that visits all nodes in S:

$$\text{tour}_G(S) \stackrel{\text{def}}{=} \max_{v \in S} \min_{W \in \mathcal{W}(v;S)} \text{length}_G(W); \qquad (2)$$

where $\mathcal{W}(v;S)$ is the set of walks in G starting at vertex v and visiting every vertex in S.

Given a metric space $(X; d_X)$ its doubling constant is the minimum number such that, for every $r > 0$, every ball of radius r can be covered by at most balls of radius $r/2$ (Gupta et al., 2003). The doubling constant of a graph G is the doubling constant of $(G; d)$ where d is the shortest path distance on G. The doubling dimension of the space, denoted , is defined as $\stackrel{\text{def}}{=} \log_2$.

Notation For Exploration Algorithms Recall that in exploration problems, the true graph G is initially unknown to the searcher. Thus in the setting of exploration, one needs to distinguish between the shortest known path distance between two vertices and the true shortest path distance. To this end, let V_{i-1} be the set of vertices visited by iteration i and let G_i be the subgraph of G containing: all of the vertices in $V_{i-1} \cup \partial V_{i-1}$, and all of the edges adjacent to V_{i-1}. Throughout this paper, we emphasize the distinction between d_{G_i} and d_G. In general, we have $d_{G_i} \geq d_G$.

When analyzing performance, we denote the progress made on the ith iteration as

$$\Delta_i \stackrel{\text{def}}{=} d_G(v_i; g) - d_G(v_{i+1}; g). \qquad (3)$$

Observe that, under the assumption that $v_0; v_1; \ldots; v_T$ are nodes visited by some algorithm which originates at r and terminates at g (i.e. $v_0 = r$ and $v_T = g$) we have:

$$\sum_{i=0}^{T-1} \Delta_i = d_G(r; g) - 0 = \text{OPT}.$$

Metric Embeddings When analyzing the planning algorithms of Banerjee et al. (2023), we consider metric embeddings on graphs, and parametrize results in terms of the distortion of the relevant embedding:

Definition 1 (Distortion). Given a function $: X \to Y$ between two finite metric spaces $(X; d_X)$ and $(Y; d_Y)$, we define the distortion $\text{dist}(\ )$ of as the minimum value satisfying the following: there exists a constant $c > 0$ such that for all $x_1; x_2 \in X$,

$$c \cdot d_X(x_1; x_2) \leq d_Y(\ (x_1);\ (x_2)) \leq c \cdot \ \cdot d_X(x_1; x_2).$$

3 EXPLORATION UNDER ABSOLUTE ERROR

In this section we study the absolute error regime, in which the ℓ_1 norm of the vector of errors is bounded by some constant E_1, not necessarily known to the searcher. We consider the following natural rule: on the ith iteration, choose the next vertex to visit by

picking the node $v_i \in \partial V_{i-1}$ minimizing the sum of $d_{G_i}(v_{i-1}, v_i)$ and $f(v_i)$ (See Algorithm 1). This iterative step can be interpreted as visiting the vertex that would be on the shortest path to the goal if all the predictions $f(v)$ were correct.

Algorithm 1 ℓ_1-Greedy_Search(G, r)

```
v_0 ← r
i ← 0
V_0 ← {r}
while v_i ≠ g do
    i ← i + 1
    v_i ∈ argmin_{v ∈ ∂V_{i-1}} d_{G_i}(v_{i-1}, v) + f(v)
    V_i ← {v_0, ..., v_i}
end while
```

We remark that because we are working in the exploration setting, Algorithm 1 does not have access to the true distances $d_G(\cdot,\cdot)$ and must instead make use of $d_{G_i}(\cdot,\cdot)$ the distances in the subgraph of observed vertices at iteration i.

Theorem 1 parametrizes the worst-case guarantees for Algorithm 1 in terms of the total negative error E^- and the maximum-occurring positive error E_1^+. This asymmetry corresponds to the intuition that positive errors can obstruct the search task more dramatically than negative errors by obscuring shortest paths to the goal. Indeed, an immediate corollary of Theorem 1 is the following result about the performance of Algorithm 1 in the setting in which the prediction function is admissible (i.e. error is only negative):

Corollary 5. Consider the problem of searching a weighted (possibly directed) graph with predictions f satisfying $f(v) \le d_G(v, g)$ $\forall v \in V$. Then there exists an algorithm which finds the goal g with cost at most $\mathrm{OPT} + E_1$, where E_1 is the total ℓ_1 error in the predictions, i.e. $E_1 \stackrel{\text{def}}{=} \sum_{v \in V} |f(v) - d(v, g)|$.

The dependency on E^- and E_1^+ in Theorem 1 is optimal in the following sense:

Theorem 6. For every $E^- > 0$, there exist graph search instances with total negative error E^- such that any algorithm for the exploration problem on these instances must incur cost at least $\mathrm{OPT} + E^-$ in the worst case. Additionally, for any $n > 3$ and any E_1^+, there exist graph search instances on n nodes with maximum positive error E_1^+ such that any algorithm for the exploration problem on these instances must incur cost at least $\mathrm{OPT} + E_1^+(n-2)$ in the worst case.

We note that the lower bound in Theorem 6 also holds for the expected distance travelled of randomized search strategies, up to constant factors, as per the following result.

Proposition 7. For every $E^- > 0$ there exists a graph search instance with total negative error E^- such that any randomized algorithm incurs expected costs at least $\mathrm{OPT} + \frac{1}{2}E^-$ on this instance. Moreover, for any $n > 3$ and any E_1^+ there exists a graph search instance on n nodes with maximum positive error E_1^+ such that any randomized algorithm must incur expected cost at least $\mathrm{OPT} + (n-2)E_1^+/2$ on this instance.

The full proof of Theorem 1 and the proof of Corollary 5, given in Section B, rely on a charging argument which shows that distance travelled away from the goal can be directly attributed to errors in the predictions of observed nodes. The proof of Theorem 6 and Proposition 7 are constructive and can also be found in Section B.

For completeness we now sketch the proof of Theorem 1. The proof considers the progress Φ_i as in Equation (3), which measures how much closer the agent is to the goal after the i^{th} iteration of the algorithm. The cost of the algorithm is given by $\mathrm{ALG} = \sum_{i \in [T]} d_{G_i}(v_{i-1}, v_i)$. At each step $i \in [T]$, one can show that the distance travelled $d_{G_i}(v_{i-1}, v_i)$ in the observed subgraph G_i is bounded above by the sum of three terms:

$$d_{G_i}(v_{i-1}, v_i) \le \Phi_{i-1} + E^-(v_i) + E^+(w_i); \qquad (4)$$

where $E^-(v_i)$ is the negative error at v_i, and $E^+(w_i)$ is the positive error at some vertex w_i on a shortest path from v_{i-1} to g. In particular, all three terms in this upper bound are independent of the observed subgraph G_i. The statement of the theorem then follows by summing both sides of Equation (4) over all $i \in [T]$.

4 EXPLORATION UNDER RELATIVE ERROR

In this section, we consider the setting when the prediction function satisfies Equation (1) for every $v \in V$. We assume that $\varepsilon \in (0, 1)$: in particular, if $\varepsilon \ge 1$, then $f(v) = 0$ is a valid prediction at every vertex and no exploration algorithm can avoid visiting the entire graph in the worst case.

If ε is known to the searcher a priori then given access to the prediction at a node, the searcher can construct an upper bound on the true distance-to-goal. On trees this allows one to limit exploration to a ball of some radius R (dependent on ε and OPT) around the initial vertex, effectively pruning distant nodes from the vertex set. In particular, one could limit their search to the set:

$$S_{\varepsilon, r} \stackrel{\text{def}}{=} \{v \in V \mid d_G(v, r) \le \frac{1}{1-\varepsilon} f(r)\}. \qquad (5)$$

We couple this observation with the algorithm in the previous section and obtain the following algorithm.

Algorithm 2 "-Known_Search(G, r, ε)

$v_0 \leftarrow r$
$i \leftarrow 0$
$V_0 \leftarrow \{r\}$
while $v_i \neq v$ **do**
  $i \leftarrow i + 1$
  $v_i \in \operatorname{argmin}_{v \in \partial V_{i-1} \setminus S_{\varepsilon,r}} d_{G_i}(v_{i-1}, v) + f(v)$
  $V_i \leftarrow \{v_0, \ldots, v_i\}$
end while

In Section B we leverage favorable properties of the set $S_{\varepsilon,r}$ to show that Algorithm 2 satisfies the guarantees of Theorem 2. We observe that in particular, Theorem 2 implies that for any n the competitive ratio incurred by Algorithm 2 tends to 1 as $\varepsilon \to 0$. The combination of the truncation with the shortest-path rule in Algorithm 2 was necessary to secure this property: for example, a simple scheme such as running breadth-first-search on the truncated set $S_{\varepsilon,r}$ would not enjoy such a guarantee in the worst case.

We note that Algorithm 2 crucially relies on the fact that the searcher knows the value of ε and so it cannot be deployed in the setting where ε is unknown. For the latter regime we propose an alternative algorithm, also based on Algorithm 1, which provably succeeds for unknown values of ε under the assumption that ε is small, e.g. $\varepsilon < 1/3$.

Algorithm 3 "-Unknown_Weighted_Search(G, r, γ)

$v_0 \leftarrow r$
$i \leftarrow 0$
$V_0 \leftarrow \{r\}$
while $v_i \neq v$ **do**
  $i \leftarrow i + 1$
  $v_i \in \operatorname{argmin}_{v \in \partial V_{i-1}} d_{G_i}(v_{i-1}, v) + f(v)$
  $V_i \leftarrow \{v_0, \ldots, v_i\}$
end while

In Section B we show that on trees this reweighting scheme ensures that Algorithm 3 never explores nodes which are far from the goal and use this property to establish the guarantees in Theorem 3 under the setting where $\gamma = 2/3$.

We give a lower bound to establish that Algorithm 2 is almost optimal. Specifically, we show that even when ε is known a-priori, the asymptotic dependence on ε is tight up to factors of $1/(1-\varepsilon)$:

Theorem 8. For all n sufficiently large ($n \geq 6$) and for any $\varepsilon \in (0, 1)$, there exists an instance I of the exploration on weighted trees G with predictions (1), such that any algorithm for the exploration problem on G must incur cost $ALG \geq (1 + n\varepsilon)OPT$ on I.

Note that this lower bound also applies to the regime addressed by Algorithm 3. Moreover, one can prove an analogous lower bound for the case of potentially randomized exploration strategies, as per the following result.

Proposition 9. For all n sufficiently large ($n \geq 6$) and for any $\varepsilon \in (0, 1)$, there exists an instance I of the exploration on weighted trees G with predictions (1), such that any randomized algorithm for the exploration problem on G must incur cost $ALG \geq (1 + \frac{n\varepsilon}{2})OPT$ on I.

4.1 Planning With Relative Error

Under the model of relative error described by Equation (1), planning problems become either trivial or impossible, with no intermediate regimes. In the context of predictions $f(v) = (1 + \varepsilon_v)d_G(v, g)$ for some $\varepsilon_v \in [-\varepsilon, \varepsilon]$, we observe that it must be that $f(g) = 0$, independent of the value of ε. For the planning problem to be nontrivial, there must also occur vertices $v \neq g$ such that $f(v) = 0$. Thus, for nontrivial instances of the planning problem in this regime, $\varepsilon \geq 1$. However, instances with such (large) values of ε are hopeless in the worst case: such a setting allows for the prediction of every node to be set equal to 0, a case which forces the searcher to visit every node in the worst case.

5 PLANNING

5.1 Planning Bounds Via Metric Embeddings

In this section, we analyze the performance of the algorithm by Banerjee et al. (2023) for the planning problem, as a function of how similar the target graph is to some graph G' admitting inexpensive tours. In the planning problem, the full graph G as well as all predictions f are made available to the algorithm upon initialization. Banerjee et al. (2023) study the implied error functions $\eta_0 : V \to \mathbb{R}$ and $\eta_1 : V \to \mathbb{R}$, defined as

$$\eta_0(v) \stackrel{\text{def}}{=} |\{u \in V : f(u) \neq d_G(u, v)\}|$$

and

$$\eta_1(v) \stackrel{\text{def}}{=} \sum_{u \in V} |f(u) - d_G(u, v)|.$$

Banerjee et al. (2023) consider an algorithm that iteratively visits sublevel sets of η_0 or η_1 respectively, for geometrically increasing thresholds (see Algorithm 4). Their algorithm is very simple: for every threshold, it visits every node in the sublevel set before increasing

to the next threshold value. Algorithmically, Banerjee et al. (2023) accomplish this by computing a minimum length Steiner tree of the sublevel set, which is in general not computationally e cient. However, one can replace this computationally expensive procedure with a polynomial-time constant factor approximation (see e.g. (Karlin et al., 2021)), which preserves the asymptotic upper bound on the algorithms' performance.

Algorithm 4 FullInfoX (G; r;) from Banerjee et al. (2023)

B 8 2 R : L ( ) := f v 2 G j (v) g
V_i r
0
while g 62 V_i do
V_i V_i [L (2)
Compute a minimum length Steiner tree of V_i and perform an Euler tour of the tree
+ 1
end while

This de nition of Algorithm 4 motivates our analysis, which focuses on the distortion of embedding the instance graph into some graph G^0 which admits inexpensive tours. Recall the de nition of $\text{tour}_G(S)$ as in Equation (2).

De nition 2 (Easily-tourable). A graph $G^0 = (V^0; E^0)$ is c-easily-tourable for some $c > 0$ if for any S^0 V^0, tour_{G^0} c diam(S^0).

In Section B, we establish that in easily-tourable graphs the algorithm of Banerjee et al. enjoys good performance. We then show that if a graph G can be embedded into an easily-tourable graph G^0 with distortion , then G itself must be easily-tourable with tour costs that scale with . This culminates in the following result:

Lemma 10. Given G an unweighted graph, if G admits an embedding : G ! G^0 of distortion for G^0 some c_{G^0}-easily-tourable graph, then Algorithm 4 with objective = $_0$ from Banerjee et al. (2023) incurs cost at most OPT + O(c_{G^0} E_0): If G has integer-valued distances and admits an embedding of distortion into G^0 some easily-tourable graph, then Algorithm 4 with objective = $_1$ incurs cost at most OPT + O(c_{G^0} E_1):

In particular, (weighted) paths and cycles are easily-tourable with respect to constant c, resulting in the guarantees in Theorem 4. In Section C we give results suggesting that our analysis of planning problems via metric embeddings is a re nement of the analysis of Banerjee et al. (2023).

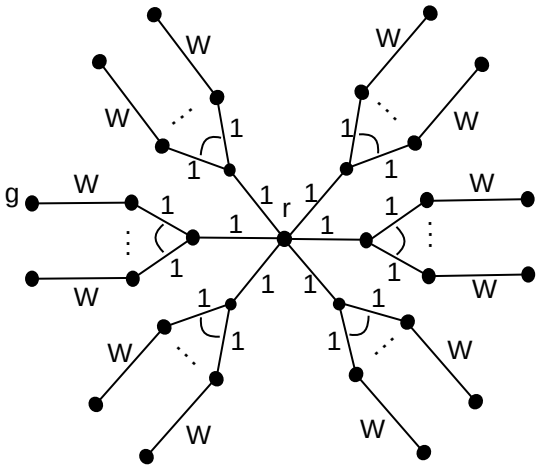

Figure 1: The lower bound construction for the proof of Lemmas 11 and 13.

5.2 Lower bounds For Planning

In this section, we provide lower bounds that extend the results in Banerjee et al. (2023). Consider the planning problem on some weighted graph G with integer weights[2] where prediction error is parameterized by E_1. Banerjee et al. (2023) propose an algorithm which provably incurs cost at most OPT + O(E_1poly ( )); where is the doubling constant of the input graph. We provide complementary lower bounds by establishing the following result:

Lemma 11. Let A be any algorithm for the planning problem which is guaranteed to incur cost: OPT + O(E_1^a ^{b}) for some a; b 2 R when run on a weighted graph G with doubling constant , and with a error vector e such that kek$_1$ = E_1. Then it must be the case that a 1. Moreover, if a = 1, then b 1.

Our lower bounds are constructive, and consider a family of graphs illustrated in Figure 1. A full proof can be found in Section B.

It is known that the doubling constant is always at least as large as the maximum degree but we note that in general it may be much larger even if one restricts themselves to trees. We analyze the algorithm of Banerjee et al. (2023) and prove that in trees with integer weights performance of their algorithm can be bounded in terms of maximum degree, at the cost of paying a higher asymptotic dependence on E_1:

Lemma 12. Given G a tree with integer-valued distances and maximum degree , consider the planning problem on G with predictions satisfying E_1 1.

[2] We note that the analysis of Banerjee et al. for the above setting also appears to go through for the case non-integer weights, and that the lower bound provided by Lemma 11 would then hold for that setting too.

Figure 2: Performance of Algorithms 1 and 3 against random errors. Experimental procedures are detailed in Section D. The number of nodes is fixed over all graph topologies and error settings. LEFT: Average and standard deviation of ALG − OPT incurred by Algorithm 1 over 2000 independent random trials for varying values of E_1. RIGHT: Average and standard deviation of ALG/OPT incurred by Algorithm 3 over 2000 independent random trials for varying values of ε.

Then on this problem instance Algorithm 4 with objective $= {}_1$ from Banerjee et al. (2023) incurs cost at most OPT + $O(E_1^2)$:

We establish corresponding lower bounds via a similar construction to that in Lemma 11:

Lemma 13. Let A be any algorithm for the planning problem on trees with integer weights which is guaranteed to incur cost: OPT + $O(E_1^a{}^b)$ for some $a, b \in \mathbb{R}$ when run on a tree G with maximum degree and with a prediction vector $\hat{e}$ such that $\|\hat{e}\|_1 = E_1$. Then it must be the case that $a \geq 1$ and $b \geq 1$. Moreover, it must be that $a + b \geq 3$.

6 NUMERICAL EXPERIMENTS: IMPACT OF RANDOM ERRORS

Throughout this paper we have focused on worst-case theoretical guarantees. In this section, we provide numerical results exploring the effectiveness of Algorithms 1 and 3 on exploration problems beyond the worst-case setting, particularly under random errors. The experiments show that in addition to being robust to adversarial error, the algorithms considered perform well in instances with stochastic error. Moreover, we find that although the guarantees for Algorithm 3 were shown only for trees, empirically the algorithm also succeeds on general (cyclic) graphs. These results suggest that Algorithms 1 and 3 could be deployed effectively for exploration problems in practice.

We study the performance of Algorithm 1 in the absolute regime and Algorithm 3 in the relative regime. In the left subfigure of Figure 2 we plot the performance of Algorithm 1 for different graph topologies when the error is sampled at random in an absolute fashion. We plot the performance against the total ℓ_1-norm of the error vector. In the right subfigure of Figure 2 we explore the performance of Algorithm 3 against relative error. Once again, we find that the algorithm enjoys empirical performance superior to that predicted by the worst-case upper bound. In particular, the gap between the worst-case bound and the average empirical performance is consistent over families of graphs with very different topologies.

In Table 2 we report the average ratio of algorithmic cost incurred to value of the upper-bound in Theorem 1 (given as a percentage). We compute this percentage for different classes of graphs over 100 runs of these experiments when the error, initial node and goal node, and graph structure (when applicable) have been sampled at random.

We also empirically compare the performance of Algorithm 1 to another natural heuristic, which we call Smallest Prediction. In Smallest Prediction, the agent always travels to the vertex v in ∂V_1 with the smallest value of the prediction function $f(v)$. While our theoretical results already show that Algorithm 1 achieves optimal performance in the worst-case, we show that our algorithm performs better than Smallest Prediction in the presence of random error across a variety of graph topologies. In Figure 3, we plot the average performance of Algorithm 1 against the performance of Smallest Prediction (measured as the distance travelled by the agent, minus OPT, as a fraction of of OPT) in random trees with 100 vertices for a growing value of the magnitude E_1 of the error vector. More details on this comparison can be found in the supplementary material.

Further details of our experiments, including details

Figure 3: A comparison of the performance of Algorithm 1 with the Smallest Prediction heuristic. We plot the average and the standard deviation of the performance of Algorithm 1 and that of the Smallest Prediction heuristic against the magnitude of the error vector E_1. Experiments in this gure were conducted on random trees; for analogous results on other graph topologies, see Figure 9 in Appendix D.

GRAPH FAMILY	Random Lobster		Erdös Rényi		Random Tree		Circular Ladder	
COST (%)	2:4	2:5	3:1	4:3	1:7	2:3	0:7	0:5

Table 2: Average empirical cost of running Algorithm 1 as a percentage of the upper bound in Theorem 1 for di erent family of graphs. Experiments performed on graphs with 300 nodes. These results demonstrate that when run with randomly-generated errors, the actual cost incurred by the algorithm is a very small fraction of the upper bound.

on the error models and the graph families being used in the experiments can be found in Section D in the supplementary material.

7 CONCLUSIONS AND FUTURE DIRECTIONS

In this work we have introduced new general algorithms for the problem of searching in an unknown graph. Under the absolute error model we design algorithms which succeed in a broad class of graphs and prove that these algorithms are optimal (Section 3). We then move beyond the absolute error regime and consider relative error; to the best of the authors' knowledge, this work is the rst to address the exploration problem under this natural error model. Within this setting we propose algorithms for the exploration problem on weighted trees and show that their performance is nearly-optimal (Section 4).

We complement our advances in the exploration setting by expanding the landscape of results for the planning problem. We extend the work of Banerjee et al. (2023) by providing alternative performance guarantees which establish a linear rather than quadratic dependency on the error parameter E_0 in some graph families, and which suggests that such a lower asymptotic dependency may be attainable in general (Theorem 4). We also complete the results of Banerjee et al. in the planning setting on integer-distance graphs by proving one cannot improve the factor of E_1 in their upper bound and that achieving this linear dependence on the error requires cost linear in the doubling constant of the instance graph (Lemma 11).

The work in this paper directly suggests several avenues for further study. While our lower bounds demonstrate the impossibility of uniformly improving the results in Theorem 1, it is possible the bound may be overly pessimistic in certain classes of graphs; it would be interesting to consider whether making stronger structural assumptions about the instance graph would yield better guarantees. In the setting of relative error, an immediate open problem is whether the guarantees on Algorithms 2 and 3 can be extended to more general graphs. In order to improve understanding of the planning problem, a complete characterization of easily-tourable graphs would better contextualize Lemma 10. Finally, while the numerical results suggest the algorithms proposed in this paper perform well under random errors, formal guarantees studying this setting would be a valuable addition.

Acknowledgements

The authors wish to thank all the anonymous reviewers for their thoughtful feedback.

A. DePavia is supported by NSF DGE 2140001.

While working on this project, E. Tani was supported by the Institute for Data, Econometrics, Algorithms, and Learning (IDEAL) with the NSF Grant ECCS-2216912.

References

A. Aamand, P. Indyk, and A. Vakilian. Frequency estimation algorithms under zip an distribution. arXiv preprint arXiv:1908.05198, 2019.

A. Aamand, J. Y. Chen, and P. Indyk. (Optimal) Online Bipartite Matching with Degree Information. In Advances in Neural Information Processing Systems, volume 35, 2022.

S. Alpern and S. Gal. The theory of search games and

rendezvous, volume 55. Springer Science & Business Media, 2006.

K. Anand, R. Ge, and D. Panigrahi. Customizing ml predictions for online algorithms. In International Conference on Machine Learning, pages 303 313, 2020.

S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, and M. Renault. Online computation with untrusted advice. In 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

A. Antoniadis, C. Coester, M. Eliá², A. Polak, and B. Simon. Online metric algorithms with untrusted predictions. ACM Transactions on Algorithms, 19 (2):1 34, 2023.

E. Bamas, A. Maggiori, and O. Svensson. The primal-dual method for learning augmented algorithms. Advances in Neural Information Processing Systems, 33:20083 20094, 2020.

S. Banerjee, V. Cohen-Addad, A. Gupta, and Z. Li. Graph searching with predictions. In 14th Innovations in Theoretical Computer Science Conference, ITCS, volume 251 of LIPIcs, pages 12:1 12:24, 2023.

P. Berman. On-line searching and navigation. Online Algorithms: The State of the Art, pages 232 241, 2005.

A. Bhattacharya, B. Gorain, and P. S. Mandal. Treasure hunt in graph using pebbles. In International Symposium on Stabilizing, Safety, and Security of Distributed Systems, pages 99 113. Springer, 2022.

S. Bouchard, Y. Dieudonne, A. Labourel, and A. Pelc. Almost-optimal deterministic treasure hunt in arbitrary graphs. In International Colloquium on Automata, Languages and Programming (ICALP) 2021, 2021.

J. Y. Chen, T. Eden, P. Indyk, H. Lin, S. Narayanan, R. Rubinfeld, S. Silwal, T. Wagner, D. P. Woodru , and M. Zhang. Triangle and four cycle counting with predictions in graph streams. In 10th International Conference on Learning Representations, ICLR , 2022.

E. Cohen, O. Geri, and R. Pagh. Composable sketches for functions of frequencies: Beyond the worst case. In Proceedings of the 37th International Conference on Machine Learning, 2020.

R. Dechter and J. Pearl. Generalized best- rst search strategies and the optimality of A*. Journal of the ACM (JACM) , 32(3):505 536, 1985.

I. Diakonikolas, V. Kontonis, C. Tzamos, A. Vakilian, and N. Zari s. Learning online algorithms with distributional advice. In International Conference on Machine Learning, pages 2687 2696, 2021.

S. Dobrev, R. Královi£, and E. Markou. Online graph exploration with advice. In International Colloquium on Structural Information and Communication Complexity, pages 267 278. Springer, 2012.

E. Du, F. Wang, and M. Mitzenmacher. Putting the learning" into learning-augmented algorithms for frequency estimation. In Proceedings of the 38th International Conference on Machine Learning pages 2860 2869, 2021.

F. Eberle, A. Lindermayr, N. Megow, L. Nölke, and J. Schlöter. Robusti cation of online graph exploration methods. In Proceedings of the AAAI Conference on Arti cial Intelligence , volume 36, pages 9732 9740, 2022.

T. Eden, P. Indyk, S. Narayanan, R. Rubinfeld, S. Silwal, and T. Wagner. Learning-based support estimation in sublinear time. In 9th International Conference on Learning Representations, ICLR, 2021.

T. Eden, P. Indyk, and H. Xu. Embeddings and labeling schemes for a. In 13th Innovations in Theoretical Computer Science Conference (ITCS 2022). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

P. Erd®s, A. Rényi, et al. On the evolution of random graphs. Publ. math. inst. hung. acad. sci, 5(1):17 60, 1960.

D. Ferguson, M. Likhachev, and A. Stentz. A guide to heuristic-based path planning. In Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS) , pages 9 18, 2005.

P. Ferragina and G. Vinciguerra. Learned data structures. In Recent Trends in Learning From Data: Tutorials from the INNS Big Data and Deep Learning Conference (INNSBDDL2019), pages 5 41. Springer, 2020.

D. Foead, A. Ghifari, M. B. Kusuma, N. Hana ah, and E. Gunawan. A systematic literature review of A* path nding. Procedia Computer Science, 179: 507 514, 2021.

B. Gorain, K. Mondal, H. Nayak, and S. Pandit. Pebble guided optimal treasure hunt in anonymous graphs. Theoretical Computer Science, 922:61 80, 2022.

A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings. pages 534 543. IEEE, 2003.

A. Gupta, D. Panigrahi, B. Subercaseaux, and K. Sun. Augmenting online algorithms with "-accurate predictions. Advances in Neural Information Processing Systems, 35:2115 2127, 2022.

C. Hsu, P. Indyk, D. Katabi, and A. Vakilian. Learning-based frequency estimation algorithms. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.

P. Indyk, A. Vakilian, and Y. Yuan. Learning-based low-rank approximations. In Advances in Neural Information Processing Systems, pages 7400 7410, 2019.

T. Jiang, Y. Li, H. Lin, Y. Ruan, and D. P. Woodru . Learning-augmented data stream algorithms. In International Conference on Learning Representations, 2020.

B. Kalyanasundaram and K. R. Pruhs. Constructing competitive tours from local information. Theoretical Computer Science, 130(1):125 138, 1994.

A. R. Karlin, N. Klein, and S. O. Gharan. A (slightly) improved approximation algorithm for metric tsp. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pages 32 45, 2021.

D. Komm, R. Královi£, R. Královi£, and J. Smula. Treasure hunt with advice. In Structural Information and Communication Complexity: 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015. Post-Proceedings 22, pages 328 341. Springer, 2015.

T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. In Proceedings of the 2018 International Conference on Management of Data, pages 489 504, 2018.

Y. Li, H. Lin, S. Liu, A. Vakilian, and D. Woodru . Learning the positions in countsketch. In The Eleventh International Conference on Learning Representations, 2023.

H. Lin, T. Luo, and D. Woodru . Learning augmented binary search trees. In International Conference on Machine Learning, pages 13431 13440. PMLR, 2022.

T. Lykouris and S. Vassilvitskii. Competitive caching with machine learned advice. In International Conference on Machine Learning pages 3302 3311, 2018.

N. Megow, K. Mehlhorn, and P. Schweitzer. Online graph exploration: New results on old and new algorithms. Theoretical Computer Science, 463:62 72, 2012.

M. Mitzenmacher. A model for learned bloom lters and optimizing by sandwiching. In Advances in Neural Information Processing Systems, pages 464 473, 2018.

M. Mitzenmacher and S. Vassilvitskii. Algorithms with predictions. Communications of the ACM, 65(7):33 35, 2022.

P. Paliwal. A survey of a-star algorithm family for motion planning of autonomous vehicles. In 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pages 1 6. IEEE, 2023.

A. Pelc. Searching games with errors fty years of coping with liars. Theoretical Computer Science, 270 (1-2):71 109, 2002.

I. Pohl. Bi-directional and heuristic search in path problems. Technical report, Stanford Linear Accelerator Center, Calif., 1969.

M. Purohit, Z. Svitkina, and R. Kumar. Improving online algorithms via ml predictions. In Advances in Neural Information Processing Systems, pages 9661 9670, 2018.

L. H. O. Rios and L. Chaimowicz. A survey and classi cation of A* based best- rst heuristic search algorithms. In Brazilian Symposium on Arti cial Intelligence, pages 253 262. Springer, 2010.

A. Wei and F. Zhang. Optimal robustness-consistency trade-o s for learning-augmented online algorithms. Advances in Neural Information Processing Systems, 33:8042 8053, 2020.

A. C.-C. Yao. Probabilistic computations: Toward a uni ed measure of complexity. In 18th Annual Symposium on Foundations of Computer Science (sfcs 1977), pages 222 227. IEEE Computer Society, 1977.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes : These are included in the main body of the paper when discussing the relevant results.]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes : The summary results of the analysis are in the body of the paper, the formal proofs can be found in the supplementary material.]
   (c) (Optional) Anonymized source code, with speci cation of all dependencies, including external libraries. [Yes: see uploaded code, along with README.md le. README.md outlines all dependencies.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes : These are included in the main body of the paper.]
   (b) Complete proofs of all theoretical results. [Yes : These can be found in Section B of the supplementary material.]
   (c) Clear explanations of any assumptions. [Yes : These are included in the main body of the paper.]
3. For all gures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes: we provide both the code used to generate the gures, and data les for exactly recreating gures.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable : No model was trained for this paper.]
   (c) A clear de nition of the speci c measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes: this is discussed in Section 6 and further detailed in Section D of the supplementary material.]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes: See Section D in the supplementary material.]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identi able information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material

A SURVEY OF RELATED WORKS

Graph search with distance-to-goal predictions The problem and prediction settings considered in this work most closely correspond to those considered by Banerjee et al. (2023). Banerjee et al. (2023) study exploration and planning under absolute error models. They consider two parametrizations of prediction error: the first in terms of the number of nodes at which predictions are not equal to true distance-to-goal, denoted E_0, and the second in terms of the ℓ_1 norm of the vector of errors, denoted E_1 as in this work. They develop an algorithm for exploration on unweighted trees, and prove guarantees on its performance in terms of E_0. They also develop algorithms for planning on graphs: on unweighted graphs, they establish guarantees parameterized by E_0, while in graphs with integer-valued distances their performance bounds are parameterized by E_1.

Treasure Hunt In the treasure hunt problem, a mobile agent must traverse some unknown environment, continuous or discrete, to locate a stationary hidden goal (Alpern and Gal, 2006). When the search environment is a graph, this problem shares many features with the exploration problem considered in this paper. Bouchard et al. (2021) study the graph treasure hunt problem when the searcher receives no additional information. They establish lower bounds on the total cost incurred by any algorithm in terms of the number of edges in the ball of radius OPT around the root node, and give algorithms with performance guarantees which asymptotically exceed these lower bounds by at most a factor of $\log(\mathrm{OPT})$. Graph treasure hunt problems have also been considered when the agent receives help with the task. Komm et al. (2015) study the case when the the searcher receives generic advice, which can take the form of any bit string. They consider the advice complexity of the treasure hunt task; they prove that there is an algorithm which achieves competitive ratio r by receiving $O(n/r)$ bits of advice along the search and moreover they establish that any algorithm achieving a competitive ratio of r must receive (n/r) bits of advice (see Theorems 4 and 5 in Komm et al. (2015)). In the setting where graph vertices are anonymized, i.e. the searcher has no way to recognize whether a vertex has or has not previously been visited, recent work has studied the task of graph treasure hunt with access to advice from an omniscient oracle which marks vertices with binary labels (Bhattacharya et al., 2022; Gorain et al., 2022).

The exploration model studied in this paper can be viewed as a graph treasure hunt problem with specific kinds of advice (predictions of distance-to-goal). One main contrast with this work is that the advice considered can contain adversarial errors, and indeed the impact that different error models have on the graph exploration task is a core topic investigated in this work.

Path-Planning and A* Search Distance-to-goal predictions have been the subject of study in many path-planning and graph-traversal settings. Initialized with a root node and some known target node, path-planning problems seek to learn a shortest path between the root and goal and common problem models assume access to a set of distance-to-goal predictions, referred to as "heuristics" within this literature Pohl (1969); Ferguson et al. (2005). A* search is a celebrated algorithm for the problem of path-finding and graph-traversal, designed for cases when the entire graph G and all predictions f are accessible in memory, and has spawned many algorithmic variants (Rios and Chaimowicz, 2010; Foead et al., 2021; Paliwal, 2023).

Much of the theory of A* search focuses on cases when predictions have particular structural properties: a prediction function $f : V \to \mathbb{R}$ is called *admissible* if the prediction at every node v is never greater than the actual distance to the goal from v. A prediction function $f : V \to \mathbb{R}$ is *consistent* (or monotone) if for every node v and every neighbor u of v, $f(v) \le f(u) + d(v,u)$ and $f(g) = 0$. Consistency is studied so heavily in part because it implies admissibility. Admissible heuristics are well-motivated and occur in various other problems. For example, Eden et al. (2022) consider access to an oracle that provides an underestimate for the probabilities of any element in some discrete probability distribution. In addition to being well-motivated by applications, admissibility is a focus of literature because A* search with admissible heuristics enjoys optimality properties (Dechter and Pearl, 1985).

Figure 4: Comparing Algorithm 1 versus A search on a random tree with randomly generated errors. The same set of predictions is provided to both algorithms. While the set of nodes visited by the two algorithms is comparable, the computational model in A places no penalty on traversal distance so that algorithm has a tendency to double-back on itself, leading to a more expensive tour. Nodes are colored by prediction value and labeled with the order in which they are rst visited by the relevant algorithm. Tour cost is taken to be $\sum_i d(v_i, v_{i+1})$ for indices i ordered according to when a node is rst visited: the tour cost for A omits costs incurred by re-expanding a node within the execution of the algorithm.

While many of the problems and algorithms within path-planning may appear closely related to this work at rst-glance, we emphasize that the goals and cost models di erentiate the graph searching problems considered in this work from those in path-planning. In particular, the design and algorithmic guarantees on path-planning algorithms like A search implicitly assume that the full graph and predictions are available in memory upon initialization of the algorithm. Performance guarantees and notions of optimality are proven in terms of computational procedures, rather than traversal distance. For example, when predictions are admissible Ais optimal in the sense that the set of nodes expanded (an operation analogous to visiting a node) is minimal (Dechter and Pearl, 1985). However, the sequence in which these nodes are expanded can incur high traversal distance, as illustrated in Figure 4. More generally, the goals of algorithms for path-planning di er substantially from that in the graph search problem: the path- nding problem seeks to learn and return a shortest path even if the process required to nd such a path is expensive in the sense of graph traversal, whereas the aim of a graph searching problem is nding the goal node in an inexpensive manner and makes no demand that a shortest path from the root node to the goal be in the set of visited nodes upon termination of the algorithm. These di erences in goals and algorithmic guarantees mean that in cases when the environment is unknown, i.e. when the graph and predictions are not available in memory but must instead by accessed by traversal, path-planning algorithms may incur much higher traversal cost than the graph search algorithms proposed in this paper.

Learning-Based Algorithms. Recent years have seen a marked increase in the integration of machine learning techniques to enhance traditional algorithmic challenges. Such algorithms have been developed for various topics including online algorithms Lykouris and Vassilvitskii (2018); Purohit et al. (2018); Angelopoulos et al. (2020); Wei and Zhang (2020); Bamas et al. (2020); Aamand et al. (2022); Antoniadis et al. (2023); Anand et al. (2020); Diakonikolas et al. (2021); Gupta et al. (2022), data structures Kraska et al. (2018); Mitzenmacher (2018); Ferragina and Vinciguerra (2020); Lin et al. (2022), and streaming models Hsu et al. (2019); Indyk et al. (2019); Aamand et al. (2019); Jiang et al. (2020); Cohen et al. (2020); Du et al. (2021); Eden et al. (2021); Chen et al. (2022); Li et al. (2023). For an extensive collection of learning-based algorithms, refer to the repository at https://algorithms-with-predictions.github.io/.

B MISSING PROOFS

In this section, we present detailed proofs of the results in the main body of the paper.

B.1 Proofs For Section 3

Proof of Theorem 1 and Corollary 5. Algorithm 1 follows a shortest path in G_i from v_{i-1} to v_i. A simple consequence of this is that:

$$ALG = \sum_{i \in [T]} d_{G_i}(v_{i-1}, v_i).$$

Let $\Delta_i \stackrel{def}{=} d_G(v_{i-1}, g) - d_G(v_i, g)$, and let T be the number of iterations of the while loop executed, so that $v_T = g$. We then have:

$$\sum_{i \in [T]} \Delta_i = d_G(v_0, g) - d_G(v_T, g) = d_G(v_0, g) - d_G(g, g) = OPT.$$

Consider any iteration $i \in [T]$. Let w_i be the first vertex outside of V_{i-1} encountered when traversing a shortest path from v_{i-1} to g. Note that $w_i \in \partial V_{i-1}$, and hence, by the update rule in Algorithm 1 we have:

$$f(v_i) + d_{G_i}(v_{i-1}, v_i) \leq f(w_i) + d_{G_i}(v_{i-1}, w_i). \tag{6}$$

Furthermore, by the definition of w_i, we have:

$$d_{G_i}(v_{i-1}, w_i) = d_G(v_{i-1}, w_i). \tag{7}$$

The above follows from a simple contradiction argument, for the existence of a shorter path from v_{i-1} to w_i in G which is not in G_i, would contradict the definition of w_i. We also have:

$$d_G(v_{i-1}, g) = d_G(v_{i-1}, w_i) + d_G(w_i, g). \tag{8}$$

We then have, for any $i \in [T]$:

$$\begin{aligned} d_G(v_i, g) - f(v_i) &\overset{(6)}{\geq} d_G(v_i, g) + d_{G_i}(v_{i-1}, v_i) - d_{G_i}(v_{i-1}, w_i) - f(w_i) \\ &\overset{(7)}{=} d_G(v_i, g) + d_{G_i}(v_{i-1}, v_i) - d_G(v_{i-1}, w_i) - f(w_i) \\ &= d_G(v_{i-1}, g) - \Delta_i + d_{G_i}(v_{i-1}, v_i) - d_G(v_{i-1}, w_i) - f(w_i) \\ &\overset{(8)}{=} d_{G_i}(v_{i-1}, v_i) - \Delta_i + d_G(w_i, g) - f(w_i). \end{aligned}$$

Note that the vertices in the sequence $\{v_i\}_{i \in [T] \cup \{0\}}$ are always distinct, while the vertices in the sequence $\{w_i\}_{i \in [T]}$ might not be. This also implies that $T \leq n$. The above then implies:

$$\begin{aligned} ALG &= \sum_{i \in [T]} d_{G_i}(v_{i-1}, v_i) \\ &\leq \sum_{i \in [T]} \Delta_i + d_G(v_i, g) - f(v_i) + f(w_i) - d_G(w_i, g) \\ &= \sum_{i \in [T]} \Delta_i + \sum_{i \in [T]} d_G(v_i, g) - f(v_i) + \sum_{i \in [T]} f(w_i) - d_G(w_i, g) \\ &\leq OPT + E_1^- + T \cdot E_1^+ \\ &\leq OPT + E_1^- + n \cdot E_1^+. \end{aligned}$$

This completes the proof of Theorem 1. When predictions are admissible $E_1^- = E_1$ and $E_1^+ = 0$ so Corollary 5 follows. □

Proof of Theorem 6. We begin by proving the first half of the theorem. Given $E_1 > 0$ one considers the three-vertex graph path P_3 where the two edges are weighted with weight $w = E_1/2$, the start vertex / root is chosen to be the middle vertex and the goal is one of the other two vertices (see left side of Figure 5). Note that the value of OPT is $d_G(r, g) = w$. When the predictions on the vertices are given by: $f(v_1) = 0$, $f(v_2) = w$ and

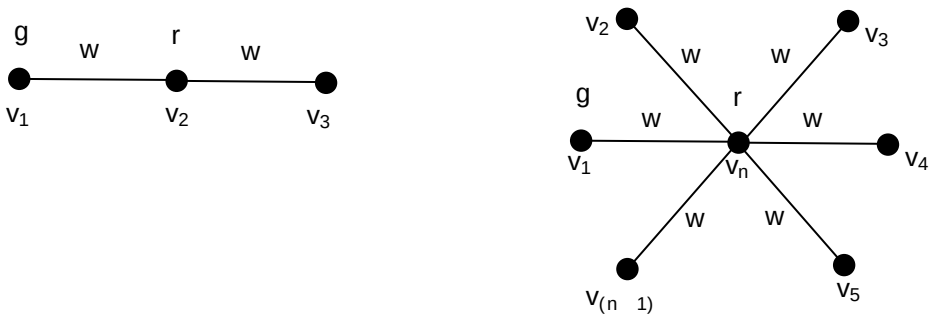

Figure 5: The construction in the proof of Theorem 6.

$f(v_3) = 0$, the error is equal to E and the graph looks completely symmetric to the searcher, and hence in the worst case to find the goal, the searcher has to incur a cost of $3w = w + 2w = OPT + E$ as needed.

For the second part of the theorem, we construct a star on n vertices, where each edge has weight $w = E_1^+/2$, the starting vertex is at the center of the star, and the goal is chosen arbitrarily among the other vertices. The prediction at the goal is then picked to equal E_1^+ so that it equals the prediction in all other vertices, i.e. we set the predictions to $f(r) = w$ and $f(v) = 2w$ for all $w \neq r$. Every algorithm will then have to visit the entire star in the worst case, incurring a cost of $E_1^+/2 + E_1^+(n-2) = OPT + E_1^+(n-2)$ as needed (See the right side of Figure 5). □

Proof of Proposition 7. We apply Yao's minimax principle (Yao, 1977) to the same constructions used in the proof of Theorem 6. For both constructions, we consider the distribution over instances produced by choosing the goal node uniformly at random among the leaf nodes.

In particular, for the first statement, we consider the performance of any deterministic algorithm on the distribution of instances given by taking the three-node path graph on the left-hand side of Figure 5, and placing the goal g at either v_1 or v_3 with equal probability. We fix predictions $f(v_1) = f(v_3) = 0$ and $f(v_2) = w$ as in the proof of Theorem 6. The expected cost incurred by any deterministic algorithm over this distribution of instances is $2w = OPT + w = OPT + \frac{1}{2}E$. Yao's minimax principle then implies the stated lower bound for all randomized algorithms.

The proof of the second result follows analogously by considering the second construction in the proof of Theorem 6. □

B.2 Proofs For Section 4

B.2.1 Algorithmic Guarantees Under Relative Error

We begin the analysis of Algorithm 2 by establishing the following properties of the set $S_{\varepsilon,r}$ defined in Equation (5). For G a tree, let $P_G(u,v)$ denote the (unique) shortest path between nodes u and v in G.

Lemma 14. For $S_{\varepsilon,r}$ as defined in (5) and G a weighted tree, then the following hold:

(i) $\forall v \in S_{\varepsilon,r}$, $d_G(v,r) > OPT$,

(ii) $\forall v \in S_{\varepsilon,r}$, $d_G(v,r) \leq \frac{1+\varepsilon}{1-\varepsilon} OPT$,

(iii) $\forall v \in S_{\varepsilon,r}$, $d_G(v,g) \leq \frac{2}{1-\varepsilon} OPT$,

(iv) $P_G(r,g) \subseteq S_{\varepsilon,r}$,

(v) For any $v \in S_{\varepsilon,r}$, $P_G(v,g) \subseteq S_{\varepsilon,r}$.

Proof. The properties follow immediately from the de nition of the relative error model in Equation (1) and the de nition of $S_{\varepsilon,r}$ in Equation (5).

(i) $\forall v \in S_{\varepsilon,r}$, $f(v) > \frac{1}{1-\varepsilon} f(r)$ and by Equation (1), $f(r) \geq (1-\varepsilon) d_G(r,g) = (1-\varepsilon)\mathrm{OPT}$.

(ii) $\forall v \in S_{\varepsilon,r}$, $d_G(v,r) \leq \frac{1}{1-\varepsilon} f(r)$, and by Equation (1), $f(r) \leq (1+\varepsilon)\mathrm{OPT}$.

(iii) By triangle inequality, for any $v \in G$

$$d_G(v,g) \leq d_G(v,r) + d_G(r,g) = d_G(v,r) + \mathrm{OPT}$$

and so by property (ii) above, for any $v \in S_{\varepsilon,r}$ the result follows.

(iv) $\forall v \in P_G(r,g)$, $d_G(r,v) \leq d_G(r,g) = \mathrm{OPT}$. Thus property (i) above implies $\forall v \in P_G(r,g)$, $v \in S_{\varepsilon,r}$.

(v) Consider $v \in S_{\varepsilon,r}$, and let $u \stackrel{\text{def}}{=} \operatorname{argmin}\{ d_G(u,g) \mid u \in P_G(v,r)\}$. Because G is a tree,

$$P_G(v,g) = P_G(v,u) \cup P_G(u,g).$$

In particular, $P_G(v,u) \subseteq P_G(v,r)$; observe that because $v \in S_{\varepsilon,r}$, $\forall w \in P_G(v,r)$, $d_G(w,r) \leq d_G(v,r) \leq \frac{1}{1-\varepsilon} f(r)$ and thus by the de nition of $S_{\varepsilon,r}$, $P_G(v,r) \subseteq S_{\varepsilon,r}$. We've thus concluded that $P_G(v,u) \subseteq S_{\varepsilon,r}$.

For the second portion of the path, $P_G(u,g)$, observe that by the de nition of $u = \operatorname{argmin}\{ d_G(u,g) \mid u \in P_G(v,r)\}$, it must be that $u \in P_G(r,g)$. Thus $P_G(u,g) \subseteq P_G(r,g)$ and so by property (iv), $P_G(u,g) \subseteq S_{\varepsilon,r}$.

□

With these properties, we now bound the distance travelled on the ith step of the algorithm:

Lemma 15. Let G be a weighted tree, with predictions satisfying Equation (1) with respect to parameter $\varepsilon < 1$. Then, the distance travelled by Algorithm 2 on the ith iteration satis es

$$(1-\varepsilon) d_{G_i}(v_{i-1}, v_i) \leq \ell_i + 2\varepsilon d_G(v_i, g). \tag{9}$$

Additionally, if the predictions on G are multiplicative and decremental,

$$d_{G_i}(v_{i-1}, v_i) \leq \ell_i + \varepsilon d_G(v_i, g). \tag{10}$$

Proof of Lemma 15. We observe that, by de nition of Algorithm 2, for all iterations i, $V_{i-1} \subseteq S_{\varepsilon,r}$. Consider w_i the rst vertex outside of V_{i-1} encountered when traversing $P(v_{i-1}, g)$, the shortest path from v_{i-1} to g. Note that $w_i \in \partial V_{i-1}$, and that by property (v) of Lemma 14, $w_i \in S_{\varepsilon,r}$.

Thus, on iteration i, $\exists w_i \in \partial V_{i-1} \cap S_{\varepsilon,r} \cap P(v_{i-1}, g)$, so by the de nition of Algorithm 2 such w_i satis es

$$f(v_i) + d_{G_i}(v_{i-1}, v_i) \leq f(w_i) + d_{G_i}(v_{i-1}, w_i).$$

In particular, because $w_i \in P(v_{i-1}, g) \cap \partial V_{i-1}$ and by the tree properties of G, we have

$$d_{G_i}(v_{i-1}, w_i) = d_G(v_{i-1}, w_i) = d_G(v_{i-1}, g) - d_G(w_i, g).$$

We can thus upper bound

$$\begin{aligned} d_{G_i}(v_{i-1}, v_i) &\leq d_G(v_{i-1}, g) - d_G(w_i, g) + f(w_i) - f(v_i) \\ &= d_G(v_{i-1}, g) - d_G(v_i, g) + d_G(v_i, g) - f(v_i) + f(w_i) - d_G(w_i, g) \\ &= \ell_i - \varepsilon_{v_i} d_G(v_i, g) + \varepsilon_{w_i} d_G(w_i, g) \end{aligned}$$

where $\varepsilon_{v_i}, \varepsilon_{w_i} \in [-\varepsilon, \varepsilon]$ are the constants whose existence is implied by Equation (1) such that

$$f(v) = (1 + \varepsilon_v) d_G(v, g). \tag{11}$$

Consider two cases: first, consider the case when $\varepsilon_{w_i} > 0$. Because $w_i \in P(v_{i-1}, g)$,

$$d_G(w_i, g) \leq d_G(v_{i-1}, g) \leq d_G(v_{i-1}, v_i) + d_G(v_i, g).$$

Combining this bound and the fact that $\varepsilon_{v_i}, \varepsilon_{w_i} \in [-\varepsilon, \varepsilon]$ yields:

$$\varepsilon_{v_i} d_G(v_i, g) + \varepsilon_{w_i} d_G(w_i, g) \leq 2\varepsilon d_G(v_i, g) + \varepsilon d_G(v_{i-1}, v_i).$$

Thus in this case, the desired bound in (9) is satisfied.

In the second case, when $\varepsilon_{w_i} \leq 0$,

$$\varepsilon_{v_i} d_G(v_i, g) + \varepsilon_{w_i} d_G(w_i, g) \leq \varepsilon_{v_i} d_G(v_i, g) \leq \varepsilon d_G(v_i, g)$$

which is trivially upper bounded by $2\varepsilon d_G(v_i, g) + \varepsilon d_G(v_{i-1}, v_i)$. Thus, in both cases, the desired bound in (9) is satisfied.

In the case of decremental errors, $\varepsilon_v \leq 0 \; \forall v \in V$. Thus the latter case always applies, so we can bound

$$d_{G_i}(v_{i-1}, v_i) \leq \varepsilon_i + \varepsilon_{v_i} d_G(v_i, g) + \varepsilon_{w_i} d_G(w_i, g) \leq \varepsilon_i + \varepsilon d_G(v_i, g)$$

thus establishing the bound in (10). □

We are now equipped to prove Theorem 2.

Proof of Theorem 2. We will show that Algorithm 2 achieves the desired competitive ratio. We begin by establishing that the algorithm will terminate at the goal node g: by Property (iv) in Lemma 14, $P(r, g) \subseteq S_{\varepsilon, r}$, and further by definition $P(r, g)$ is connected, so Algorithm 2 initialized at r will explore a connected subgraph of G that contains g, and will thus terminate at g.

We now bound the total distance travelled by Algorithm 2. Consider the general case, when errors can be incremental or decremental. Then, by Lemma 15,

$$ALG = \sum_{i \in [T]} d_{G_i}(v_{i-1}, v_i) \leq \sum_{i \in [T]} (1+\varepsilon) d_{G_i}(v_{i-1}, v_i) + \varepsilon d_{G_i}(v_{i-1}, v_i)$$

$$\leq \sum_{i \in [T]} \varepsilon_i + \sum_{i \in [T]} 2\varepsilon d_G(v_i, g) + \sum_{i \in [T]} \varepsilon d_G(v_{i-1}, v_i) = OPT + 2\varepsilon \left(\sum_{i \in [T]} d_G(v_i, g) \right) + \varepsilon ALG.$$

Because $V_i \subseteq S_{\varepsilon, r}$ for all iterations i, and by property (iii) in Lemma 14,

$$2\varepsilon \sum_{i \in [T]} d_G(v_i, g) \leq 2\varepsilon |S_{\varepsilon, r}| \frac{2}{1-\varepsilon} OPT.$$

Thus, re-arranging,

$$(1-\varepsilon) ALG \leq OPT \left(1 + |S_{\varepsilon, r}| \frac{4\varepsilon}{1-\varepsilon} \right);$$

yielding the claimed competitive ratio from the trivial upper bound $|S_{\varepsilon, r}| \leq n$.

In the case of decremental errors, Lemma 15 and a similar argument give

$$ALG = \sum_{i \in [T]} d_{G_i}(v_{i-1}, v_i) \leq \sum_{i \in [T]} \varepsilon_i + \sum_{i \in [T]} \varepsilon d_G(v_i, g) \leq OPT \left(1 + |S_{\varepsilon, r}| \frac{2\varepsilon}{1-\varepsilon} \right);$$

yielding the claimed competitive ratio. □

To prove Theorem 3, we'll use the following lemmas: the first (Lemma 16) establishes that certain algorithms never explore nodes too far from g. We emphasize that the below lemma makes use of distances in the full G, not G_i. In the case of weighted trees, these two distances are always identical: on a tree, for all iterations i and for any $u, v \in V_i \cup \{\varnothing\}$, $d_{G_i}(u, v) = d_G(u, v)$. The second lemma (Lemma 17) bounds the distance travelled by these algorithms on any given iteration.

Lemma 16. Consider the exploration problem on G a weighted, undirected graph with predictions f satisfying Equation (1). Consider the update rule used in Algorithm 3:

$$v_i = \operatorname*{argmin}_{v \in V \setminus V_{i-1}} \{ d_G(v_{i-1}, v) + f(v) \}; \tag{12}$$

and assume > 0 satisfies < 1 ". Then, for every iteration $i \in [T]$, the node v_i visited by the algorithm on the i^{th} iteration satisfies

$$d_G(v_i, g) \quad \frac{1 + " + }{1 \quad (" +)} \mathrm{OPT}:$$

Proof of Lemma 16. Let r_i be the first vertex outside of V_{i-1} encountered when traversing $P_G(r, g)$ a shortest path from the root r to g. Note that $r_i \in V \setminus V_{i-1}$, and thus by Equation (12),

$$d_G(v_{i-1}, v_i) + f(v_i) \quad d_G(v_{i-1}, r_i) + f(r_i):$$

In particular, by the triangle inequality we can bound

$$\begin{aligned} f(v_i) \quad & (d_G(v_{i-1}, r_i) \quad d_G(v_{i-1}, v_i)) + f(r_i) \\ & d_G(r_i, v_i) + f(r_i) \\ & (d_G(r_i, g) + d_G(v_i, g)) + f(r_i): \end{aligned}$$

Given $r_i \in P_G(r, g)$, $d_G(r_i, g) \quad d_G(r, g) = \mathrm{OPT}$. Moreover, recalling that the predictions f must satisfy Equation (1), let $"_{v_i}$ and $"_{r_i}$ be the relative errors at vertex v_i and r_i respectively (defined as in Equation (11)). Then we can rewrite the bound above as

$$(1 + "_{v_i}) d_G(v_i, g) \quad (d_G(r_i, g) + d_G(v_i, g)) + (1 + \ "_{r_i}) d_G(r_i, g);$$

and hence:

$$(1 + "_{v_i} \quad) d_G(v_i, g) \quad (\ + 1 + \ "_{r_i}) \mathrm{OPT}:$$

Using the fact that $"_{v_i}, "_{r_i} \in [\ ", "]$ and the assumption that $\ $ satisfies < 1 ", we can use the above bound to conclude

$$d_G(v_i, g) \quad \frac{1 + " + }{1 \quad (" +)} \mathrm{OPT}:$$

In particular, this holds on any iteration independently of i, so we obtain the desired result. □

Lemma 17. Consider the exploration problem on G a weighted, undirected graph with predictions f satisfying Equation (1). Assume > 0 satisfies

$$\frac{1 + "}{2} < \ < 1 \ ": \tag{13}$$

Then the distance traversed by Algorithm 3 on the i^{th} iteration is bounded by

$$d_{G_i}(v_{i-1}, v_i) \quad \frac{}{2 \quad 1 \quad " \ _i} + \frac{2"}{2 \quad 1 \quad "} d_G(v_i, g):$$

Proof of Lemma 17. Let $P_G(u, v)$ denote an (arbitrary) shortest path between nodes u and v in G. Algorithm 3 follows a shortest path in G_i from v_{i-1} to v_i. Let w_i be the first vertex outside of V_{i-1} encountered when traversing $P_G(v_{i-1}, g)$. Note that as a consequence $w_i \in V \setminus V_{i-1}$, and hence by (12) we have

$$d_{G_i}(v_{i-1}, v_i) + f(v_i) \quad d_{G_i}(v_{i-1}, w_i) + f(w_i):$$

Since $w_i \in P_G(v_{i-1}, g)$,

$$d_{G_i}(v_{i-1}, w_i) = d_G(v_{i-1}, w_i) = d_G(v_{i-1}, g) \quad d_G(w_i, g):$$

Re-arranging and using the above fact, we obtain

$$\begin{aligned} d_{G_i}(v_{i-1}, v_i) \quad & d_{G_i}(v_{i-1}, w_i) + f(w_i) \quad f(v_i) \\ = \quad & d_G(v_{i-1}, g) \quad d_G(w_i, g) \ + f(w_i) \quad f(v_i) \end{aligned}$$

$$= \quad d_G(v_{i-1}; g) \quad d_G(v_i; g) \; + \quad d_G(v_i; g) \quad f(v_i) \; + \; f(w_i) \quad d_G(w_i; g) \; :$$

Let $\varepsilon_{v_i}; \varepsilon_{w_i}$ be the relative prediction errors at v_i and w_i respectively (de ned as in Equation (11)), and recall the de nition of $_i$ in (3). Then we can rewrite the above as

$$d_{G_i}(v_{i-1}; v_i) \quad _i + (\quad 1 \quad \varepsilon_{v_i}) d_G(v_i; g) + (1 + \varepsilon_{w_i} \quad) d_G(w_i; g):$$

Because $w_i \; 2 \; P_G(v_{i-1}; g)$, we have that

$$d_G(v_{i-1}; w_i) + d_G(w_i; g) = d_G(v_{i-1}; g) \quad d_G(v_{i-1}; v_i) + d_G(v_i; g):$$

Moreover, by upper bound on in (13), $(1 + \varepsilon_{w_i} \quad) \quad 0$, so we can revise our upper bound:

$$d_{G_i}(v_{i-1}; v_i) \quad _i + (\quad 1 \quad \varepsilon_{v_i}) d_G(v_i; g) + (1 + \varepsilon_{w_i} \quad) \; d_G(v_{i-1}; v_i) + d_G(v_i; g) \; :$$

Re-arranging and recalling $\varepsilon_{v_{i-1}}; \varepsilon_{w_i} \; 2 \; [\quad \varepsilon; \varepsilon]$ yields

$$(2 \quad 1 \quad \varepsilon) d_{G_i}(v_{i-1}; v_i) \quad _i + 2\varepsilon d_G(v_i; g)$$

Leveraging the lower bound on in (13), we can divide to obtain the desired result. □

Theorem 3 follows immediately from Lemmas 16 and 17:

Proof of Theorem 3. Observe that for $\varepsilon \; 2 \; (0; 1=3)$, $= 2=3$ always satis es Equation (13), independently of the value of ε. Thus for this setting, Lemma 17 implies that the update cost on a single iteration of Algorithm 3 is bounded as

$$d_{G_i}(v_{i-1}; v_i) \quad \frac{2}{1 \quad 3\varepsilon} \quad _i + \frac{6\varepsilon}{1 \quad 3\varepsilon} d_G(v_i; g): \tag{14}$$

In particular, for G a weighted tree, on all iterations i, for all $u; v \; 2 \; V_i \; [\; @V$,

$$d_{G_i}(u; v) = d_G(u; v):$$

This, in combination with the choice of $= 2=3$ implies that Lemma 16 applies, so for all vertices v_i visited by the algorithm, we have

$$d_G(v_i; g) \quad \frac{5 + 3\varepsilon}{1 \quad 3\varepsilon} \mathrm{OPT}: \tag{15}$$

We can then upper bound the last term in Equation (14) and obtain:

$$d_{G_i}(v_{i-1}; v_i) \quad \frac{2}{1 \quad 3\varepsilon} \quad _i + \frac{6\varepsilon}{1 \quad 3\varepsilon} \quad \frac{5 + 3\varepsilon}{1 \quad 3\varepsilon} \mathrm{OPT}$$

Thus, letting T denote the total number of iterations made by Algorithm 3, summing over all iterations yields

$$\mathrm{ALG} = \sum_{i \; 2 \; [T]} d_{G_i}(v_{i-1}; v_i) \quad \frac{2}{1 \quad 3\varepsilon} \sum_{i \; 2 \; [T]} \quad _i + \frac{6\varepsilon}{1 \quad 3\varepsilon} \quad \frac{5 + 3\varepsilon}{1 \quad 3\varepsilon} \mathrm{OPT} \quad T:$$

In particular, $\sum_{i \; 2 \; [T]} \quad _i = \mathrm{OPT}$, and as every iteration i must end at some distinct v_i satisfying (15),

$$T \quad B \quad g; \frac{5 + 3\varepsilon}{1 \quad 3\varepsilon} \mathrm{OPT} \quad n:$$

We then have:

$$\begin{aligned} \mathrm{ALG} \quad & \frac{2}{1 \quad 3\varepsilon} \mathrm{OPT} + \frac{6\varepsilon}{1 \quad 3\varepsilon} \quad \frac{5 + 3\varepsilon}{1 \quad 3\varepsilon} \mathrm{OPT} \quad T \\ & \mathrm{OPT} \quad 2 + \frac{6\varepsilon}{1 \quad 3\varepsilon} + \frac{6\varepsilon}{1 \quad 3\varepsilon} \quad \frac{5 + 3\varepsilon}{1 \quad 3\varepsilon} \quad n \quad : \end{aligned}$$

Giving the result in the statement of the theorem. □

B.2.2 Lower bounds For Exploration With Relative Error

Proof of Theorem 8. We consider a star in which every edge has weight w_1, to this, we add a new vertex g connected to one of the outside vertices by an edge of weight w_2. We consider the case when the initial position r is the central node of the star. In this case, the optimal algorithmic cost is

$$\mathrm{OPT} = d_G(r, g) = w_1 + w_2.$$

For the given $\varepsilon \in (0, 1)$, consider choice of w_1 and w_2 such that

$$\varepsilon = \frac{w_1}{w_1 + w_2}.$$

Note that in particular, such a setting of weights allows for the following predictions: every node except for the root and the goal can have prediction

$$f(v) = (1 - \varepsilon)(2w_1 + w_2).$$

For the above setting of w_1 and w_2, this satisfies Equation 1 with respect to ε. In particular, for a searcher starting at the root, all neighbors of the root appear identical.

In this error regime, every algorithm for the exploration problem has to explore all branches of the star before finding g in the worst-case. Thus any algorithm has to incur cost at least

$$\mathrm{ALG} = 2w_1 (n - 3) + (w_1 + w_2) = w_1(2(n - 3) + 1) + w_2.$$

The competitive ratio in this instance is thus

$$\frac{\mathrm{ALG}}{\mathrm{OPT}} = (2(n - 3) + 1)\frac{w_1}{w_1 + w_2} + \frac{w_2}{w_1 + w_2} = (2(n - 3) + 1)\varepsilon + (1 - \varepsilon) = (1 + (n - 1)\varepsilon).$$

□

Proof of Proposition 9. Consider a distribution over instances obtained by taking the instance defined in the proof of Theorem 8, selecting a neighbor v of the root vertex r uniformly at random, and replacing the edge incident to the goal vertex g with an edge gv of weight w_2.

Any deterministic algorithm for the exploration problem running on an instance sampled from this distributions incurs expected cost at least:

$$\mathrm{ALG} = (n - 3)\frac{1}{2}2w_2 + \mathrm{OPT} = ((n - 3)\varepsilon + 1)\,\mathrm{OPT} \geq \left(1 + \frac{n\varepsilon}{2}\right)\mathrm{OPT}.$$

The lower bound for randomized algorithms then follows from applying Yao's minimax principle (Yao, 1977). □

B.3 Proofs For Section 5

Throughout this subsection, we will use the following properties of ϕ_0 and ϕ_1 established by Banerjee et al. (2023):

Lemma 18. Corollary 5.4 and Lemma 5.10 in Banerjee et al. (2023). Given G an unweighted graph, for any $u, v \in G$,

$$\phi_0(u) + \phi_0(v) \leq d_G(u, v).$$

For G weighted,

$$\phi_1(u) + \phi_1(v) \leq 2d_G(u, v).$$

B.3.1 Planning Bounds Via Metric Embeddings

The distortion of an embedding can be related to its Lipschitz constant and that of its inverse: the Lipschitz constant of f is defined as:

$$\|f\|_{\mathrm{Lip}} \stackrel{\mathrm{def}}{=} \max_{x_1, x_2 \in X} \frac{d_Y(f(x_1), f(x_2))}{d_X(x_1, x_2)}.$$

Note that any map with non-trivial distortion must be injective, and thus considering $\phi^{-1} : Y \to X$,

$$\mathrm{dist}(\phi) = \|\phi\|_{\mathrm{Lip}} \cdot \|\phi^{-1}\|_{\mathrm{Lip}}.$$

To prove Lemma 10, we'll use the following fact to relate tours in G to tours in some embedding.

Lemma 19. Consider an embedding $\phi : G \to G'$ for $G = (V, E)$ and $G' = (V', E')$. Then for any $S \subseteq V$,

$$\mathrm{tour}_G(S) \leq \|\phi^{-1}\|_{\mathrm{Lip}} \cdot \mathrm{tour}_{G'}(\phi(S)).$$

Proof of Lemma 19. Recall the definition of $\mathrm{tour}_G(S)$ given in Equation (2):

$$\mathrm{tour}_G(S) \stackrel{\text{def}}{=} \max_{v \in S} \min_{W \in \mathcal{W}(v,S)} \mathrm{length}_G(W),$$

where $\mathcal{W}(v, S)$ is the set of walks in G starting at vertex v and visiting every vertex in S. Consider any walk $W = (u_1, \ldots, u_k)$ in G' starting at some $u_1 \in \phi(S)$ and visiting all of $\phi(S)$. Let $W' = (u'_1, \ldots, u'_{k'})$ be the subsequence of W containing only the points in $\phi(S)$. Note that W' contains all of the points in $\phi(S)$. We have:

$$\mathrm{length}_{G'}(W) = \sum_{i=1}^{k-1} d_{G'}(u_i, u_{i+1}) \geq \sum_{i=1}^{k'-1} d_{G'}(u'_i, u'_{i+1}) \tag{16}$$

$$\geq \sum_{i=1}^{k'-1} \frac{1}{\|\phi^{-1}\|_{\mathrm{Lip}}} \, d_G(\phi^{-1}(u'_i), \phi^{-1}(u'_{i+1})). \tag{17}$$

So, letting W'' be the walk visiting the vertices $(\phi^{-1}(u'_i))_{i=1}^{k'}$ in order while walking the shortest path in G between them. We then have:

$$\mathrm{length}_{G'}(W) \geq \sum_{i=1}^{k'-1} \frac{1}{\|\phi^{-1}\|_{\mathrm{Lip}}} \, d_G(\phi^{-1}(u'_i), \phi^{-1}(u'_{i+1})) = \frac{1}{\|\phi^{-1}\|_{\mathrm{Lip}}} \, \mathrm{length}_G(W'').$$

In particular, for any starting point $v \in S$ and any walk in $W \in \mathcal{W}(\phi(v), \phi(S))$ there exists some walk $W'' \in \mathcal{W}(v, S)$ such that:

$$\mathrm{length}_G(W'') \leq \|\phi^{-1}\|_{\mathrm{Lip}} \cdot \mathrm{length}_{G'}(W),$$

so that, for every $v \in S$:

$$\min_{W \in \mathcal{W}(v,S)} \mathrm{length}_G(W) \leq \|\phi^{-1}\|_{\mathrm{Lip}} \min_{W \in \mathcal{W}(\phi(v), \phi(S))} \mathrm{length}_{G'}(W) \leq \|\phi^{-1}\|_{\mathrm{Lip}} \max_{u \in \phi(S)} \min_{W \in \mathcal{W}(u, \phi(S))} \mathrm{length}_{G'}(W),$$

and hence:

$$\max_{v \in S} \min_{W \in \mathcal{W}(v,S)} \mathrm{length}_G(W) \leq \|\phi^{-1}\|_{\mathrm{Lip}} \max_{u \in \phi(S)} \min_{W \in \mathcal{W}(u, \phi(S))} \mathrm{length}_{G'}(W),$$

completing the proof. □

We now use Lemma 19 to establish Lemma 10.

Proof of Lemma 10. Given a real-valued function $f : V \to \mathbb{R}$, we denote the sublevel set of f about threshold c as

$$L_f(c) \stackrel{\text{def}}{=} \{v \in V : f(v) \leq c\}.$$

For the first part of the result, let G be an unweighted graph and consider the sublevel set $L_{\phi_0}(\cdot)$. By definition of the Lipschitz constant of $\phi : G \to G'$, for all $u, v \in G$

$$d_{G'}(\phi(u), \phi(v)) \leq \|\phi\|_{\mathrm{Lip}} \, d_G(u, v).$$

Thus by Lemma 18, the embedding of the sublevel set has bounded diameter: let $u, v \in L_{\phi_0}(\cdot)$ such that $\mathrm{diam}(L_{\phi_0}(\cdot)) = d_{G'}(\phi(u), \phi(v))$. Then

$$\mathrm{diam}(L_{\phi_0}(\cdot)) = d_{G'}(\phi(u), \phi(v)) \leq \|\phi\|_{\mathrm{Lip}} \, d_G(u, v) \leq \|\phi\|_{\mathrm{Lip}} (\phi_0(u) + \phi_0(v)) \leq \|\phi\|_{\mathrm{Lip}} \cdot 2.$$

Using this result along with the bound from Lemma 19 and the assumption that G' is $c_{G'}$ easily-tourable, we can bound

$$\begin{aligned}\mathrm{tour}_G(L_{\ell_0}(\ell)) &\le k^{-1}k_{\mathrm{Lip}}\ \mathrm{tour}_{G'}(L_{\ell_0}(\ell))\\ &\le k^{-1}k_{\mathrm{Lip}}\ c_{G'}\mathrm{diam}(L_{\ell_0}(\ell))\\ &\le k^{-1}k_{\mathrm{Lip}}\ c_{G'}\ 2k\ k_{\mathrm{Lip}}\\ &= 2c_{G'}.\end{aligned}$$

We now use this bound to analyze the cost of Algorithm 4. Algorithm 4 sequentially visits sublevel sets of ℓ_0. On an iteration k corresponding to threshold ℓ_k, the algorithm visits each node in $L_{\ell_0}(\ell_k) \setminus V$ by computing a constant-factor approximation to the following problem: for $\{v_1,\dots,v_{|L_{\ell_0}(\ell_k)|}\}$ the nodes of $L_{\ell_0}(\ell_k)$ and $\Pi(n)$ the set of permutations on integers $1,\dots,n$, the algorithm computes

$$\pi_k \stackrel{\text{def}}{=} \operatorname*{argmin}_{\pi\in\Pi(|L_{\ell_0}(\ell_k)|)} \sum_{i=1}^{|L_{\ell_0}(\ell_k)|} d_G(v_{\pi(i)}, v_{\pi(i+1)}).$$

The total distance travelled on the iteration k is thus

$$\sum_{i=1}^{|L_{\ell_0}(\ell_k)|} d_G(v_{\pi(i)}, v_{\pi(i+1)}) \le \max_{v\in L_{\ell_0}(\ell_k)} \min_{W\ni(v,L_{\ell_0}(\ell_k))} \mathrm{length}_G(W) = \mathrm{tour}_G(L_{\ell_0}(\ell_k)).$$

Algorithm 4 begins with $\ell_0 = 1$ and doubles the threshold on each iteration, such that $\ell_k = 2^k$. In particular, $\ell_0(g) = E_0$, so the algorithm is guaranteed to terminate by the time it has visited every node of $L_{\ell_0}(\ell_k)$ for the first sufficiently large threshold $\ell_k \ge E_0$. The algorithmic cost can thus be bounded as

$$\begin{aligned}\mathrm{ALG} &\le d_G(r, L_{\ell_0}(1)) + \sum_{k=0}^{\lceil\log_2(E_0)\rceil} \mathrm{tour}_G(L_{\ell_0}(2^k))\\ &\le d_G(r, L_{\ell_0}(1)) + 2c_{G'} \sum_{k=0}^{\lceil\log_2(E_0)\rceil} 2^k\\ &\le d_G(r, L_{\ell_0}(1)) + 2c_{G'}(4E_0 - 1).\end{aligned}$$

Using Lemma 18, we can bound the transition cost from r to the first sublevel set as

$$d_G(r, L_{\ell_0}(1)) \le d_G(r,g) + d_G(g, L_{\ell_0}(1)) \le \mathrm{OPT} + (\ell_0(g) + 1) = \mathrm{OPT} + E_0 + 1.$$

Combining these yields

$$\mathrm{ALG} \le \mathrm{OPT} + E_0(8c_{G'} + 1) \le 2c_{G'} + 1 = \mathrm{OPT} + O(c_{G'}E_0),$$

as desired.

For the second part of the result, let G be an graph with integer-valued distances consider the sublevel set $L_{\ell_1}(\ell)$, and note that Lemma 19 holds for both weighted and unweighted graphs. The proof then follows analogously to the above argument, using the appropriate bound relating ℓ_1 to distances in G from Lemma 18. □

B.3.2 Lower Bounds For Planning Problems

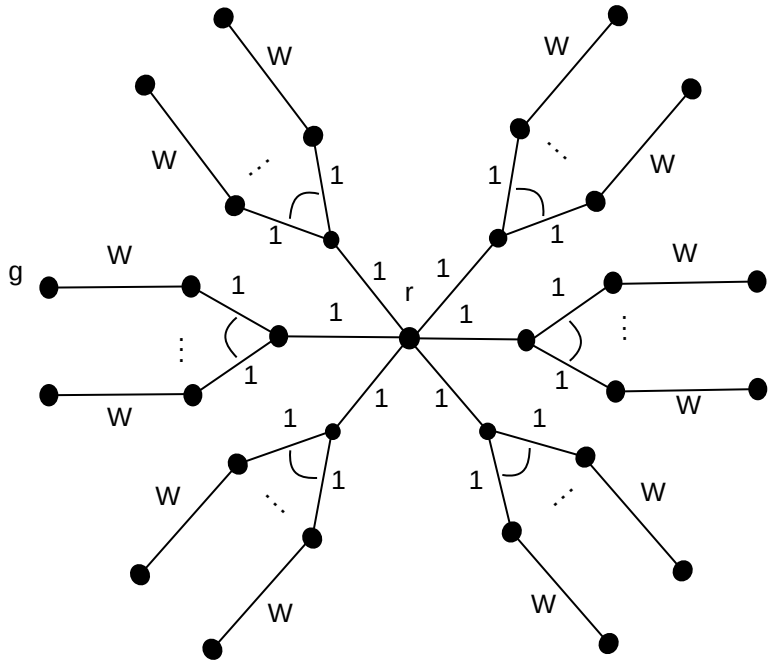

Figure 6: Reproduction of Figure 1. The lower bound construction for the proof of Lemma 11

Proof of Lemma 11. We construct a family of graphs with uniform predictions and analyze the worst-case cost incurred by any algorithm for the planning problem.

For a given and $W \in \mathbb{N}$ let $G_{;W}$ be following graph: consider a root node r with child nodes $v_1, \ldots, v$. Let every edge (r, v_i) have edge weight 1. Each child node v_i then has 1 descendants $u_1^i, \ldots, u^i{}_{1}$ with edge weights 1 for each edge (v_i, u_j^i). Each of these descendants has a single child node w_j^i to which u_j^i is attached with edge weight W. This construction is illustrated in Figure 1.

We consider the planning problem when the searcher is initialized at the root node r described above, and the goal is the leaf node w_1^1. We consider the case when error in the predictions is such that each subtree rooted at v_i appears to have the same predictions. In this construction, predictions are equal to true distance-to-goal for all nodes which are descendents of v_1 for $i \neq 1$, and error is allocated only over descendents of v_1. We first calculate the total error in such predictions:

$$E_1 = |f(v_1) - d_G(v_1, g)| + |f(g)| + \sum_{j=2}^{} |f(u_j^1) - d_G(u_j^1, g)| + |f(w_j^1) - d_G(w_j^1, g)| = 2W + 6 + 4 :$$

Under these predictions, all nodes on a given level from the root appear identical to the searcher. As a result, in the worst-case any algorithm for this problem must visit every node and incur cost ALG at least as large as the shortest tour of the graph starting at r and ending at g. Hence:

$$ALG \geq W(2^{2} - 2 - 1) + 2(^{2} - 1):$$

On the other hand, we have $OPT = d_G(r, g) = W + 2$. This gives:

$$ALG - OPT \geq 2W(^{2} - 1) + 2(^{2} - 2):$$

In order to understand how this cost scales with our parameters of interest, we now establish bounds on the doubling constant of G that show that $= (^{2})$. Recall that the doubling constant is defined to be the

minimum value of such that for any radius R, any ball of radius R can be covered with at most many balls of radius $R/2$. Observe that the number of nodes n always upper bounds the doubling constant, so $2^{2}+ +1$. To lower bound the doubling constant, consider the ball of radius $W+2$ centered at the root node r, and note that this ball contains the entire graph. For W large (W 4), $^{2}+1$ many balls of radius $W/2+1$ are required to cover the nodes of the graph, hence ^{2} 1 .

The fact that a 1 then follows from observing that is independent of W while ALG OPT = (W^{2}). The fact that if $a = 1$, b 1 similarly follows from the above along with $E_1 = (W +)$. □

Proof of Lemma 13. To establish that a 1 and that $a + b$ 3, we consider the same construction outlined in the proof of Lemma 11 above. The same argument implies a 1. Setting $W =$ yields a family of problem instances on integer-weighted trees for which ALG OPT = (^{3}) where $E_1 = ( )$. Thus on this family of instances any algorithm which is guaranteed to incur cost ALG OPT = $O(E_1^{a\ b})$ must have $a + b$ 3.

To establish that b 1, we consider a construction in which E_1 scales independently of . Consider the weighted star with edge weights w, and assume the searcher is initialized at the central root node r and the goal node g is a leaf node as illustrated on the right side of Figure 5. We consider the same predictions constructed in the proof of Theorem 6: all nodes except for g have $f(v) = d_G(v; g)$, and $f(g) = 2w$ so that predictions at all leaf nodes appear uniform. Then in the worst case, the searcher must visit every node in the graph, incurring traversal cost

$$ALG = w(2 \quad 1):$$

However, the total $`_1$ norm of the vector of errors is $E_1 = 2w$ independent of , and OPT = w, so the result follows. □

Lemma 20. Let A be any algorithm for the planning problem on weighted trees which is guaranteed to incur cost: OPT + $O(E_1^{a\ b})$ on graph searching instance I, where is the minimum distortion of embedding the instance graph into the path. Then a 1 and b 1.

Proof. The proof is entirely analogous to that of the second part of Lemma 13. For any value of E_1, one can make use of the same construction (the weighted star, with r the central node and g a leaf) with weights $w = E_1/2$ on each edge.

The result then follows from observing that for the family of constructed graphs, = and the analogous calculations. □

B.4 Planning On Trees With Integer-Valued Distances

In this section, we prove Lemma 12. The result follows from arguments analogous to those outlined in Banerjee et al. (2023) Section 5.1: we rst state and prove three necessary lemmas.

Lemma 21 (Analogous to Lemma 5.3 in Banerjee et al. (2023)). Given G with positive integer distances $d : V \quad V \to Z_{0}$, for any $U \quad V$

$$jS \setminus M(U)j \quad \sum_{u \in U} \quad '_1(u);$$

where

$$M(U) \stackrel{def}{=} f v \in V : d(v; u) = d(v; u^{0})\ 8u; u^{0} \in Ug:$$

Proof Lemma 21. Given $d : V \quad V \to Z_{0}$, for any $U \quad V$, $8w \in M(U)$ let $u_w; v_w$ denote elements of U such that $d_G(w; u_w)$ 6 $d_G(w; v_w)$. In particular, because $d(\ ;\ )$ is integer-valued, $8w \in M(U)$

$$jd_G(w; u_w) \quad d_G(w; v_w)j \quad 1:$$

In particular, for any $S \quad V$

$$\sum_{u \in U} \quad '_1(u) = \sum_{u \in U} \sum_{w \in V} jf(w) \quad d_G(w; u)j$$
$$\sum_{u \in U} \sum_{w \in S \setminus M(U)} jf(w) \quad d_G(w; u)j$$

$$\begin{aligned}
&\sum_{w \in S \cap M(U)} |f(w) - d_G(w, u_w)| + |f(w) - d_G(w, v_w)| \\
&\sum_{w \in S \cap M(U)} |d_G(w, u_w) - d_G(w, v_w)| \\
&|S \cap M(U)|.
\end{aligned}$$

□

Lemma 22 (Generalization of Lemma 5.10 in Banerjee et al. (2023)) For any $u, v \in V$, we have:

$$\varphi_1(u) + \varphi_1(v) \geq 2d(u, v) + \sum_{w \in V \setminus \{u,v\}} |d(u, w) - d(v, w)|.$$

Proof of Lemma 22. The proof is a straight-forward application of the triangle inequality:

$$\begin{aligned}
\varphi_1(u) + \varphi_1(v) &= \sum_{w \in V} |d(u, w) - f(w)| + |d(v, w) - f(w)| \\
&\geq \sum_{w \in V} |d(u, w) - f(w) - d(v, w) + f(w)| \\
&= \sum_{w \in V} |d(u, w) - d(v, w)| \\
&= 2d(u, v) + \sum_{w \in V \setminus \{u,v\}} |d(u, w) - d(v, w)|.
\end{aligned}$$

□

We also utilize the following bound on the size of the minimum Steiner tree of any sublevel set of φ_1: recall that for a real-valued function $f : V \to \mathbb{R}$, we denote the sublevel set of f about threshold c as

$$L_f(c) \stackrel{\text{def}}{=} \{v \in V : f(v) \leq c\}.$$

Lemma 23. For G a connected tree with at least three nodes, integer edge weights, and maximum degree Δ, let C denote the set of vertices in the minimum Steiner tree containing all vertices in $L_{\varphi_1}(\ell)$. Then

$$|C| \leq \Delta.$$

Proof of Lemma 23. By definition, $L_{\varphi_1}(\ell) \subseteq C$. Let $u_1, u_2 \in L_{\varphi_1}(\ell)$ such that

$$d(u_1, u_2) = \operatorname{diam}(L_{\varphi_1}(\ell)).$$

If $\nexists w \in C$ such that $d(u_1, w) = d(u_2, w)$, then Lemma 21 implies

$$|C| = |C \cap M(u_1, u_2)| \geq \varphi_1(u_1) + \varphi_1(u_2) \geq 2.$$

In particular, for G connected with at least three nodes, $\Delta \geq 2$, so the desired bound holds.

We now consider the case when $\exists w \in C$ such that $d(u_1, w) = d(u_2, w)$. Let $q_1, \ldots, q_k$ denote the neighbors of w and let $T_i \subseteq C$ denote the subtree of descendants of w that contains q_i. Assume without loss of generality that $T_1 \ni u_1$ and $T_2 \ni u_2$. Note that, because C is defined to be minimal, $\forall i \in [k]$ $L_{\varphi_1}(\ell) \cap T_i \neq \emptyset$. Let $u_3, \ldots, u_k$ be points such that $u_i \in L_{\varphi_1}(\ell) \cap T_i$. Consider any $x \in C \setminus \{w\}$. Then $\exists j \in [k]$ such that $x \in T_j$. This case is illustrated in Figure 7.

Assume without loss of generality that $j \neq 1$ (this can be assumed WLOG because if $x \in T_1$, then the below argument can be carried out with respect to u_2). Because G is a tree and $x \notin T_1$,

$$d(u_1, x) = d(u_1, w) + d(w, x)$$

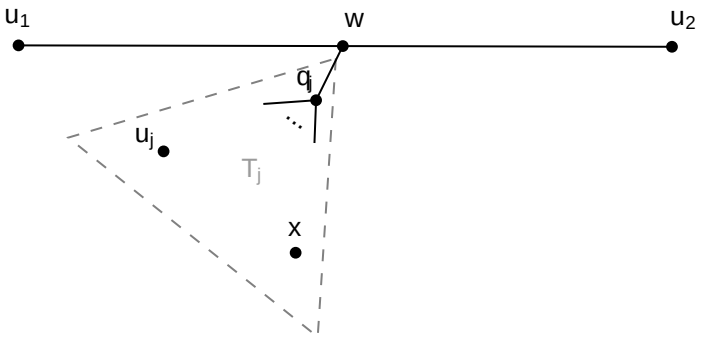

Figure 7: Visual aid for proof of Lemma 23, for the case where $\exists w \in C$ such that $d(u_1, w) = d(u_2, w)$.

By choice of u_1, u_2, $\text{diam}(L_{\ell_1}(\ell))) = 2\, d(u_1, w)$ so in particular $d(u_1, w) \geq d(u_j, w)\ \forall u_j \in L_{\ell_1}(\ell)$. Thus

$$d(u_1, x) = d(u_1, w) + d(w, x) \geq d(u_j, w) + d(w, x).$$

Moreover, for all $v \in T_j$, $d(v, w) = d(v, q_j) + d(q_j, w)$ by definition of subtree T_j, so for non-zero weights,

$$d(u_j, w) + d(w, x) > d(u_j, q_j) + d(x, q_j) \geq d(u_j, x)$$

We thus conclude $d(u_1, x) > d(u_j, x)$, which in particular implies $x \notin M(\{u_1, \ldots, u_k\})$.

Thus for all $(C \setminus \{w\}) \cap M(\{u_1, \ldots, u_k\}) = \emptyset$, so

$$|C| - 1 = |C \setminus M(\{u_1, \ldots, u_k\})| \geq \sum_{i=1}^{k} \phi_{\ell_1}(u_i) \geq 1.$$

We have thus established the result in both cases (i.e. when $C \cap M(\{u_1, \ldots, u_k\}) = \emptyset$ and when $C \cap M(\{u_1, \ldots, u_k\}) \neq \emptyset$). □

Proof of Lemma 12. Consider the cost of visiting every node in C, a minimum Steiner tree containing the sublevel set $L_{\ell_1}(\ell)$. Because C is minimal,

$$\text{diam}(C) \geq \text{diam}(L_{\ell_1}(\ell))$$

where the last inequality follows from Lemma 22. In particular, traversing C to visit every node incurs travel cost at most $\text{diam}(C) \cdot |C| \leq |C|^2$.

Algorithm 4 with objective ℓ_1 proceeds by iteratively visiting every node in the sublevel set $L_{\ell_1}(\ell)$ by computing and traversing a minimum Steiner tree C that contains the sublevel set. Algorithm 4 begins with $\ell_0 = 1$ and doubles the threshold on each iteration, such that $\ell_k = 2^k$. In particular, $\ell_1(g) = E_1$, so the algorithm is guaranteed to terminate by the time it has visited every node of $L_{\ell_1}(\ell_k)$ for the first sufficiently large threshold $\ell_k \geq E_1$. The algorithmic cost of Algorithm 4 with objective ℓ_1 is thus bounded by

$$\begin{aligned} ALG &= d(r, L_{\ell_1}(1)) + \sum_{k=0}^{\lceil \log_2(E_1) \rceil} \text{diam}(C_{2^k}) \cdot |C_{2^k}| \\ &\leq d(r, L_{\ell_1}(1)) + \sum_{k=0}^{\lceil \log_2(E_1) \rceil} (2^k)^2 \\ &\leq d(r, L_{\ell_1}(1)) + \frac{1}{3}(16E_1^2 - 1) \end{aligned}$$

Additionally, leveraging Lemma 22 and the fact that $\ell_1(g) \geq cE_1$,

$$d(r, L_{\ell_1}(1)) \leq d(r, g) + d(g, L_{\ell_1}(1)) \leq OPT + \frac{1}{2}(E_1 + 1).$$

Combining these bounds yields the desired result in the regime $E_1 \geq 1$. □

C RELATION BETWEEN EMBEDDING DISTORTION AND DOUBLING DIMENSION

We show that that every graph with a low-distortion embedding into the path also has small doubling dimension / doubling constant, as per the following lemma. Recall that the doubling constant of a metric space is the smallest value such that, for any choice of radius $R \in \mathbb{R}$, every ball of radius R can be covered with the union of balls of radius $R/2$, and that the doubling dimension is given by $\log_2$.

Lemma 24. Let G be an undirected graph on n vertices admitting an embedding into a path on n vertices with distortion and let be the doubling constant of G. Then:

$$d \quad 8 \quad e.$$

In contrast there exist graphs with constant doubling dimension that admit no embeddings into the unweighted path of distortion independent of n. For example, the 2D planar grid graph on n vertices has constant doubling dimension / doubling constant, but a simple argument shows that every embedding of the 2D planar grid into the path has distortion $(\sqrt{n})$.

Proof of Lemma 24. Let $G = (V, E)$ be an undirected graph which embeds into $[n]$ with distortion . Let be an embedding which achieves this distortion. For any $R > 0$, let $B_G(u, R) \subseteq V$ denote the ball in G centered at u of radius R, and let $B_{[n]}((u), R)$ denote the ball of radius R in $[n]$ centered at (u). Let $(S) \subseteq [n]$ denote the image of $S \subseteq V$ under . For any radius R and any $u \in V$, by the definition of the Lipschitz constant we can bound

$$d_{[n]}((u), (v)) \quad k \quad k_{Lip}\, d_G(u, v) \quad k \quad k_{Lip}\, R \qquad \forall v \in B_G(u, R)$$

so $(B_G(u, R)) \quad B_{[n]}((u), k \; k_{Lip}\, R)$. In particular, for $S_1 \quad V$, $(S_1) \quad S_2$ implies $S_1 \quad {}^{-1}(S_2)$, so we conclude $B_G(u, R) \quad {}^{-1}(B_{[n]}((u), k \; k_{Lip}\, R))$.

Consider $B_{[n]}((u), k \; k_{Lip}\, R)$. Fix and let k denote the cardinality of an -covering of $B_{[n]}((u), k \; k_{Lip}\, R)$. Observe that for any $c, \; > 0$ and any $v \in [n]$, $B_{[n]}(v, c)$ admits an -covering of cardinality at most $\lceil 2c/ \rceil$, so $k \quad \lceil 2k \; k_{Lip}\, R/ \rceil$. Let $\{x_1, \ldots, x_k\} \subseteq [n]$ denote the centers of the covering balls. Given $B_G(u, R) \quad {}^{-1}(B_{[n]}((u), k \; k_{Lip}\, R))$, we observe that

$$B_G(u, R) \quad \bigcup_{i=1}^{k} \; {}^{-1} \; B_{[n]}(x_i, \;) \; :$$

In particular, $\forall i \in [k]$, for any $x, y \in B_{[n]}(x_i, \;)$, the definition of the Lipschitz constant implies

$$d_G(\quad {}^{-1}(x), \quad {}^{-1}(y)) \quad k \quad {}^{-1} k_{Lip}\, d_{[n]}(x, y) \quad k \quad {}^{-1} k_{Lip} \quad 2 \; .$$

Thus $\operatorname{diam} \; {}^{-1} \; B_{[n]}(x_i, \;) \quad 2 \; k \quad {}^{-1} k_{Lip}$. In particular, this implies that $\forall i \in [k]$ such that ${}^{-1} \; B_{[n]}(x_i, \;) \quad \in \; ,$ $\exists v_i \in V$ such that ${}^{-1} \; B_{[n]}(x_i, \;) \quad B_G(v_i, 2 \; k \quad {}^{-1} k_{Lip})$. Thus

$$B_G(u, R) \quad \bigcup_{i=1}^{k} \; {}^{-1} \; B_{[n]}(x_i, \;) \quad \bigcup_{i=1}^{k} B_G(v_i, 2 \; k \quad {}^{-1} k_{Lip}) :$$

We have thus produced a covering of $B_G(u, R)$ using k balls of radius $2 \; k \quad {}^{-1} k_{Lip}$. We now choose so that $2 \; k \quad {}^{-1} k_{Lip} = R/2$, namely let $= R/(4k \quad {}^{-1} k_{Lip})$. The cardinality of the covering is then

$$k \quad \frac{2k \; k_{Lip}\, R}{} \quad = \quad 2k \; k_{Lip}\, R \; \frac{4k \quad {}^{-1} k_{Lip}}{R} \quad = \quad \lceil 8k \; k_{Lip} \; k \quad {}^{-1} k_{Lip} \rceil$$

Using the fact that $= k \; k_{Lip} \; k \quad {}^{-1} k_{Lip}$ we conclude that the doubling constant of G is at most $\lceil 8 \rceil$. □

D EXPERIMENTAL DETAILS

In this section, we provide a detailed descriptions of the experiments discussed in Section 6 of the main body of the paper. We outline two sets of experiments: the first set of experiments is used to evaluate the performance of Algorithm 1 in the presence of absolute error, the second set is used to evaluate the performance of Algorithm 2 in the presence of relative error. Both these sets of experiments focus on stochastic error.

All experiments in this section were run on a 2019 MacBook Pro with a 1.4 GHz Quad-Core Intel Core i5 Processor with 16 GB of RAM. No GPUs were used for this experiment.

Figure 8: A larger rendering of the left subfigure in Figure 2 in the main body of the paper.

Absolute Error The first set of experiments corresponds to the left side of Figure 2 (replicated above as Figure 8). Here, we generate an error vector $\vec{e} \in \mathbb{R}^n$ according to the following procedure: we fixed a value E_1 representing the total desired ℓ_1-norm of the vector of errors, and then we sampled a vector $\vec{e}_{\text{unsigned}}$ uniformly at random from the scaled simplex with ℓ_1-norm equal to E_1, i.e.:

$$\vec{e}_{\text{unsigned}} \quad E_1 \quad n \stackrel{\text{def}}{=} \{\vec{x} \in \mathbb{R}^n \mid \vec{x} \quad 0; \|\vec{x}\|_1 = E_1\}.$$

We then assign a random sign to each entry of $\vec{e}_{\text{unsigned}}$ to obtain $\vec{e}$; this is done by multiplying each $\vec{e}_{\text{unsigned}}[v]$ by a Rademacher random variable $_v$. Fixing a graph G, the predictions at each vertex $v \in V$ are then given by $f(v) = d_G(v, g) + _v \vec{e}_{\text{unsigned}}[v]$. This is repeated over many instance graphs selected from four classes: Random Tree, Random Lobster, Erdos-Rényi and Circular Ladder (See paragraph Graph Families below). Whenever the family of graphs chosen is stochastic, as it is the case for all classes except for Circular Ladder, the graph is also resampled from its family at each iteration, so that the expectation is taken over the sampling of the graph topology as well as the random error. For each of these problem instances, we run Algorithm 1 and record the difference between the total distance ALG travelled by the algorithm to find g, and the true shortest-path distance OPT from the starting point to g, and we plot E_1 against it. We report mean and standard deviation of ALG OPT over 2000 independent trials.

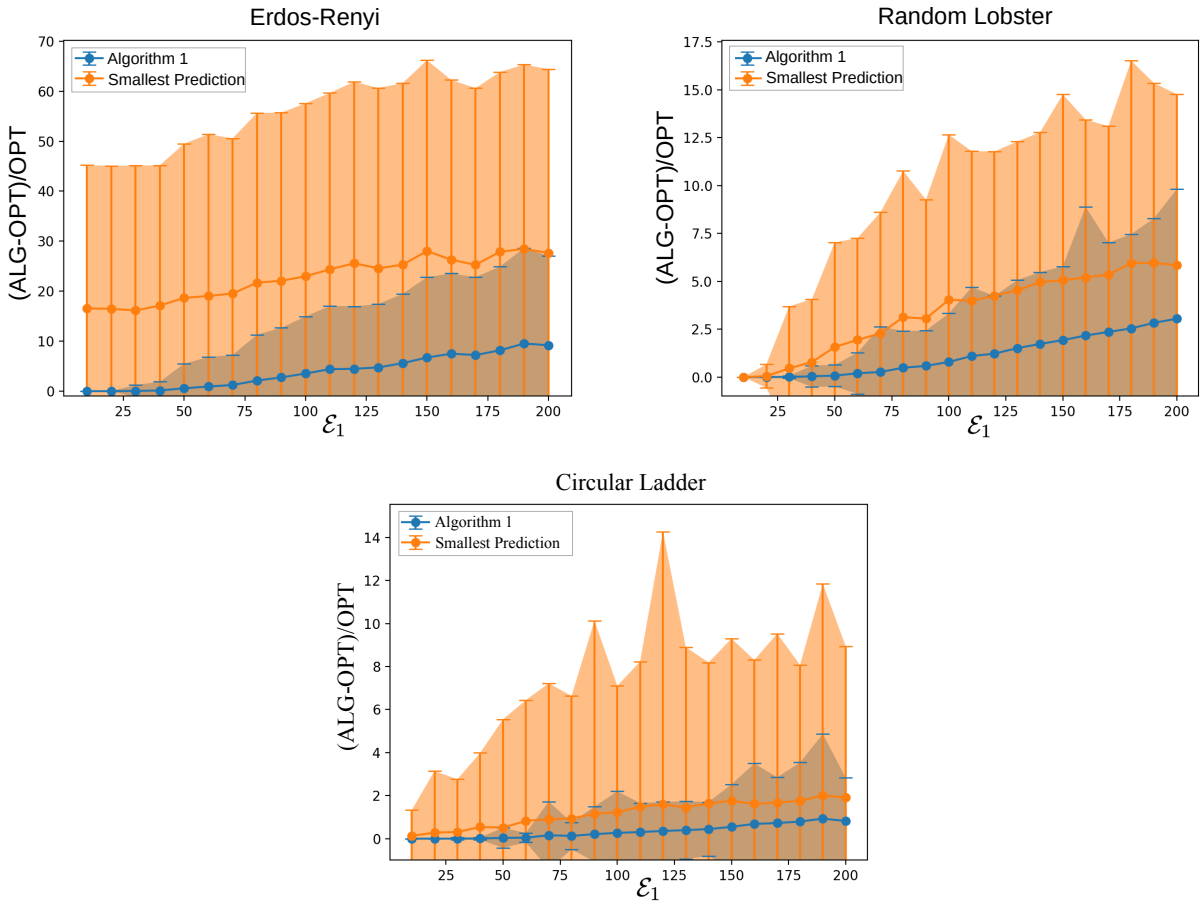

Figure 9: A comparison of the performance of Algorithm 1 with the Smallest Prediction heuristic. Each sub-figure represents one family of graphs: the top-left corresponds to random Erdös-Rényi graphs, the top-right corresponds to Random Lobster, and the middle one at the bottom corresponds to circular ladder (See **Graph Families** at the end of this section). In each figure, we plot the average and the standard deviation of the performance of Algorithm 1 and that of the Smallest Prediction heuristic against the magnitude of the error vector $\mathcal{E}_1$.

Comparison to Smallest Prediction Heuristic We then compare the performance of Algorithm 1 to the Smallest Prediction heuristic defined in Section 6 (Figure 9). Recall that in Smallest Prediction, at each iteration i, the agent travels to the an arbitrary vertex $v_i \in \text{argmin}_{v \in V_{i-1}} f(v)$. We consider the same families of graphs as in the previous section. For each family, we compare the performance of our algorithm with that of Smallest Prediction for different values of the error magnitude $\mathcal{E}_1$. The instances, including the errors, are generated like in the previous set of experiments. Performance is measured as the total distance travelled by the agent, minus the true distance OPT from r to g, as a fraction of OPT. Just like in the previous experiments, we run 2000 trials for every value of $\mathcal{E}_1$ and report the average and standard deviation of the performance across those trials.

Figure 10: A larger rendering of the right subfigure of Figure 2 in the main body of the paper.

Relative Error In the right subfigure of Figure 2, for each value of ε the predictions are generated by setting $f(v) = (1 + \varepsilon_v) \cdot d_G(v, g)$ where ε_v is sampled from a Gaussian distribution with mean 0 and standard deviation $\varepsilon/2$ conditioned on the event: $\varepsilon_v \in [-\varepsilon, \varepsilon]$. We run Algorithm 3 and plot the value of the competitive ratio ALG/OPT against the value of ε for $\varepsilon \in [0, 0.3]$, and report the mean and standard deviation incurred over 2000 independent trials.

Scaling with Number of Nodes Finally, we plot the performance of Algorithm 3 as a function of n (Figure 11). For this experiment we generate instances with different numbers of vertices and plot report the average and standard deviation of the respective empirical competitive ratios (ALG/OPT). We run 2000 trials for each family of graphs and for each number $n \in \{50, 100, 500, 1000\}$ of nodes. The error is generated as in the previous section with $\varepsilon = 0.2$.

Graph Families In the above experiments we consider the four graph families described below. All the graphs considered are undirected and unweighted. In Figure 2, we sample the below graphs on $n = 100$ nodes, and in Table 2, we sample them on $n = 300$ nodes.

- Erdös-Renyi Random Graphs: Erdös-Rényi $G_{n;p}$ graphs (Erdős et al., 1960) are a popular random graph model in the literature. We sample from the distribution of Erdös-Rényi graphs with n nodes and edge probability $p = 0.1$, conditioned on the graph being connected;
- Random Trees: Trees are just connected acyclic graphs. We sample trees on n vertices uniformly at random;
- Random Lobster Graphs: A lobster graph is a tree which becomes a caterpillar graph when its leaves are removed, we sample random lobster graphs on n vertices;
- Circular Ladder Graphs: A circular ladder graph is a graph obtained by gluing the endpoints of a ladder graph, i.e. it's a graph on vertices $\{1, ..., 2k\}$ where the edges are of the form $(i, i+1)$ for $i = 1, ..., k-1$ and $i = k+1, ..., 2k$, and $(i, k+i)$ for $i = 1, ..., k$ as well as $(1, k)$, $(k+1, 2k)$. We consider the circular ladder graph on n vertices.

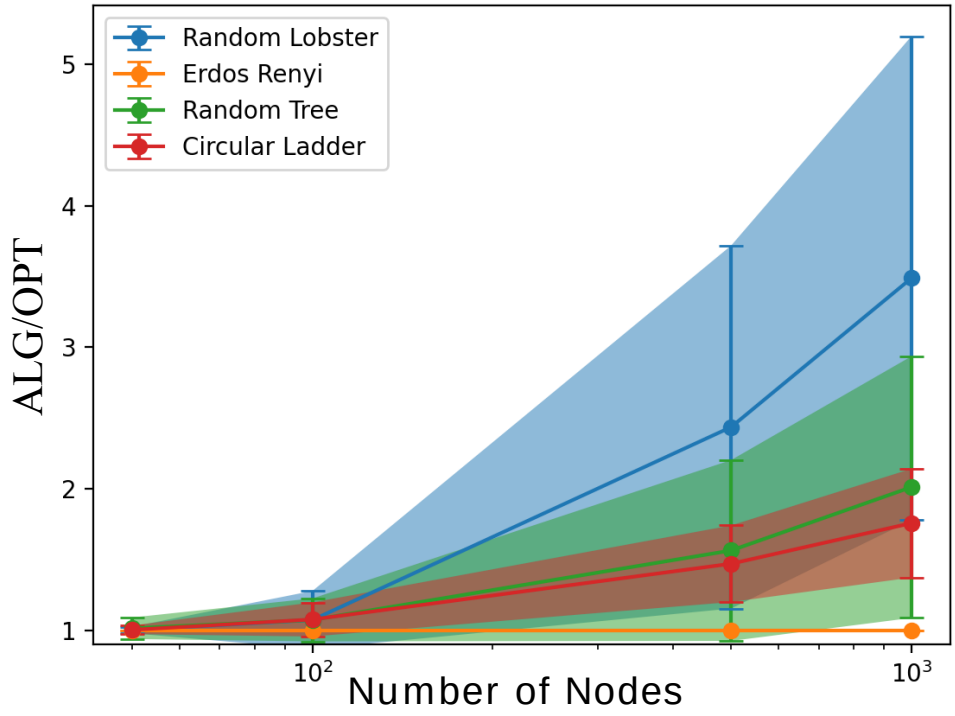

Figure 11: The empirical competitive ratio of Algorithm 3 for different graph families and for different values of n. On the x-axis: the number of vertices n in the instance graphs considered. We consider $n = 50, 100, 500, 1000$. On the y-axis: the ratio between the distance travelled by the agent, and the true distance OPT from r to g in G.