TREE SEARCH WITH DISTRIBUTIONAL PREDICTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning-augmented algorithms use machine-learned predictions to improve classical algorithmic guarantees when the predictions are accurate, while retaining rigorous performance guarantees when they are not. We study this paradigm for search on trees. Given a tree T containing an unknown target vertex t, an algorithm may query any vertex v and learn which neighbor of v lies on the unique path from v to t. The goal is to find t using as few queries as possible.

We consider the distributional setting, in which the target is drawn from an unknown distribution p and the algorithm is given a predicted distribution $\widehat{p}$ of unknown quality. We give an algorithm with expected query complexity $O\left(H(p) + k \log \eta\right)$, where $H(p)$ is the Shannon entropy of the true distribution and η is the earth mover's distance between p and $\widehat{p}$ in the tree metric. We also provide a matching lower bound that shows our algorithm is asymptotically tight.

Finally, experiments on real-world and synthetic trees show that our prediction-based algorithm can use substantially fewer queries than a simple baseline that trusts the prediction completely.

1 INTRODUCTION

Learning-augmented algorithms, also known as algorithms with predictions, aim to combine the rigorous guarantees of classical algorithms with predictions that may be obtained from historical data or machine-learning systems. Rather than assuming that a prediction is always correct, the goal is to design algorithms whose performance improves when the prediction is accurate, degrades smoothly as the prediction becomes less accurate, and still admits meaningful guarantees when the prediction is poor.

The simplest example is searching in a sorted array. Without any additional information, binary search requires $O(\log n)$ queries. Suppose instead that we are given a predicted location $\widehat{\alpha}(t)$ for the true location $\alpha(t)$ of the target. If $\eta = |\alpha(t) - \widehat{\alpha}(t)|$ is the prediction error, then the classical "doubling binary search" algorithm by Bentley & Yao (1976) uses only $O(\log \eta)$ queries. Thus an accurate prediction can lead to a substantial improvement, while the query complexity degrades smoothly as the prediction becomes less accurate. We also note that the error term is bounded above by n, so this algorithm is never asymptotically worse than the binary search algorithm that simply ignores the prediction

It is easy to see that a sorted array can equivalently be viewed as a path, where querying a vertex reveals the direction of the target. This viewpoint naturally suggests a generalization from paths to trees, leading to the *Search on Trees* problem: there is an unknown target vertex t in a tree T, and querying a vertex v reveals the unique neighbor of v lying on the path from v to t. Recent work by Dinitz & Dong (2026) studied the point-prediction version of this problem, where the algorithm is given a predicted vertex s. They showed that the $O(\log \operatorname{dist}(s,t))$ guarantee for paths cannot be obtained on arbitrary trees. However, for a tree of pathwidth k, they gave an algorithm with query complexity $O(k \log \operatorname{dist}(s,t))$. They also constructed a family of hard instances showing that this dependence is asymptotically tight.

However, point predictions are often less natural in machine learning settings. In many applications, our prediction is either generated by looking at the empirical distribution of previous (or related) instances, or is the (stochastic) output of some ML system. So in many settings, it is natural to consider a *distribution* as a prediction. This generalizes the point prediction setting, since the predicted distribution could always be a point distribution. But can we use these more general predictions?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Motivated by this question, Dinitz et al. (2024) studied the Search on Paths (sorted arrays) problem with distributional prediction, and showed that distributional predictions really can be more powerful than point predictions. In this paper we study a significant generalization of their work, namely, the distributional prediction version of Search on Trees.

Definition 1.1. *In the* Distributional Search on Trees *problem, we are given a tree $T = (V, E)$ and there is an unknown target node $t \in V$ drawn from an unknown distribution p over all the vertices. In the prediction setting we are also given a predicted distribution $\widehat{p}$ of* unknown *quality over all the vertices. We can query any node $x \in V$, and that query will return the unique neighbor of x that is on the path from x to t. The goal is to find (query) t using as few queries as possible in expectation.*

It turns out that if $p = \widehat{p}$ then it is already known that the *weighted* centroid algorithm (the natural generalization of the centroid algorithm where we weight by the distribution) is a 2-approximation of the instance-optimal search strategy (Berendsohn et al., 2023). But in the algorithms with predictions framework we do not want to assume that $p = \widehat{p}$, and do not even want to assume that we know how "close" p and $\widehat{p}$ are. So our goal is to design an algorithm for Problem 1.1 which has expected query complexity which is small when p is close to $\widehat{p}$ (often called *consistency*), degrades smoothly as $\widehat{p}$ and p diverge (often called *smoothness*), but never gets too large (often called *robustness*).

2 Our Results

Before we present the actual algorithm, we summarize what we have already known.

1. If the predicted distribution $\widehat{p}$ is accurate, then the weighted centroid algorithm, as discussed in Section 1, is *asymptotically* optimal.
2. If the prediction is just a point, we can run the point prediction algorithm by Dinitz & Dong (2026).

Of course, both cases on their own seem somewhat lacking for Problem 1.1. We are always given a predicted distribution $\widehat{p}$ of *unknown* quality, so running weighted centroid might not actually land us to a node that is relevant. And we certainly should not expect p to be concentrated just on a point.

But what if we combine the two? Inspired by Dinitz et al. (2024), our algorithm (informally) does the following. In iteration i, we perform 2^i weighted centroid queries to quickly eliminate regions with large predicted mass, and then run the point prediction algorithm by Dinitz & Dong (2026) for 2^i aueries from each of the "boundary vertices" (defined more formally in Section 5) of the remaining feasible subtrees. In this way, our algorithm can exploit $\widehat{p}$ when it is accurate while still recovering efficiently when the target lies far from where $\widehat{p}$ predicts.

One important subtlety here, which does not appear in the path case of Dinitz et al. (2024), is that there can be many boundary vertices. The natural notion of a "boundary vertex" for a subtree would be "nodes with at least one neighbor outside of the subtree". But it is easy to see that after a sequence of weighted-centroid queries, the remaining feasible subtree may have many such "boundary vertices". So naively running our point-prediction algorithm from every one of them would be prohibitively expensive. This is quite different from the path case, where there can only be two such boundary nodes.

To get around this obstacle, we show that it is actually possible to "compress" these many boundary vertices to only two candidate anchors, without increasing their distance to the target. After this compression we can simply run our bounded-pathwidth point-prediction algorithm from the two remaining anchors.

To get some high-level idea of how we compress the boundary vertices, we assign unit weight to each boundary vertex and zero weight to every other vertex of the feasible subtree, and query a weighted centroid. If the answer is here, we are done. Otherwise, we update the feasible subtree to be the component indicated by the answer, and update the boundary vertices to be those in the new feasible subtree plus the neighbor returned by the oracle. By the centroid property, this reduces the number of boundary vertices by a constant factor, so after logarithmically many queries we have at most two boundary vertices remaining. These two vertices are our candidate anchors, and we can run our point-prediction algorithm from them. Putting this all together gives the following theorem, which is our main upper bound.

Theorem 2.1. *There is a deterministic algorithm for the Distributional Search on Trees problem such that, for every tree T of pathwidth at most k, every true distribution p, and every predicted distribution $\widehat{p}$, the algorithm finds a target drawn from p using $O\left(H(p) + k \log \eta\right)$ queries in expectation, where $H(p)$ is the entropy of p and η is the earth mover's distance between p and $\widehat{p}$.*

Robustness and Running Time. Our focus is on query complexity, following Mitzenmacher & Vassilvitskii (2021), Dinitz et al. (2024), and Dinitz & Dong (2026). Robustness can be obtained essentially for free by running our algorithm in parallel with the standard centroid-search algorithm and stopping when either one finds the target. This gives query complexity $O(\min\{\log n, H(p) + k \log \eta\})$. We therefore do not discuss robustness separately in the remainder of the paper. As for computational efficiency, fortunately, Dinitz & Dong (2026) showed that the point-prediction algorithm and the required spine decompositions can be implemented in polynomial time. The additional weighted-centroid and boundary-compression operations used in our distributional algorithm are also polynomial-time, so our overall algorithm remains computationally efficient. Indeed, as we will show in Section 8, the polynomial-time runtime of our algorithms enables us to run experiments on relatively large graphs.

3 A Matching Lower Bound

Given Theorem 2.1, a natural next question is whether its dependence on entropy and prediction error is necessary. We show that both terms are independently necessary. For the prediction-error term, we reduce the point-prediction problem to the distributional setting and invoke the lower bound of Dinitz & Dong (2026), which shows that, for trees of pathwidth k, there are prediction–target pairs (s, t) for which every deterministic or zero-error randomized algorithm requires $\Omega(k \log \mathrm{dist}(s, t))$ queries in expectation.

The lower-bound construction of Dinitz & Dong (2026) fixes a tree and a predicted vertex s, and considers a hard distribution over possible targets in this tree. We associate each possible target t with a separate distributional-search instance, whose true distribution $p = \delta_t$ places unit mass at t and whose predicted distribution $\widehat{p} = \delta_s$ places unit mass at s. The original hard distribution over targets therefore induces a distribution over these instances. Within each individual instance, however, the target is deterministic, so $H(p) = 0$, and $\eta = \mathrm{EMD}(\delta_t, \delta_s) = \mathrm{dist}(s, t)$.

Since the true distribution is unknown to the algorithm, and every instance presents the same tree and predicted distribution, identifying the target requires exactly the same oracle queries as in the point-prediction setting. The original lower bound therefore carries over: every deterministic or zero-error randomized algorithm has some distributional-search instance on which it requires $\Omega(k \log \eta)$ queries in expectation, even though the true distribution in that instance has zero entropy.

The entropy term is also necessary, even when the prediction is exact. Taking $\widehat{p} = p$ on a path recovers the classical setting of binary search with a known target distribution, where the lower bound of Mehlhorn (1975) gives expected query complexity $\Omega(H(p))$. Thus, both the entropy term and the prediction-error term are independently necessary, yielding the following theorem.

Theorem 3.1. *For every $k \geq 1$, there exists an infinite family of instances of* Distributional Search on Trees *on trees of pathwidth k and maximum degree at most 3 such that every deterministic or zero-error randomized algorithm has some instance in the family on which it requires $\Omega\left(H(p) + k \log\left(\eta\right)\right)$ queries in expectation, where $\eta = \mathrm{EMD}(p, \widehat{p})$ is the earth mover's distance between the true distribution p and the predicted distribution $\widehat{p}$. Moreover, the instances witnessing the prediction-error term can be chosen so that $H(p) = 0$.*

4 Related Work

Learning-Augmented Algorithms. Learning-augmented algorithms have received substantial attention as machine-learned predictions have become increasingly available in algorithmic applications. Following the influential work of Lykouris & Vassilvitskii (2021), this framework has been explored across a wide range of settings, including online problems such as ski rental (Purohit et al., 2018), scheduling (Lattanzi et al., 2020; Gupta et al.), knapsack (Im et al., 2021; Lechowicz et al., 2024; Daneshvaramoli et al., 2025), set cover (Bamas et al., 2020), It has also been applied to accelerate

classical combinatorial algorithms (Dinitz et al., 2021), design dynamic algorithms (van den Brand et al., 2024), study mechanism-design problems (Agrawal et al., 2024) and computational geometry (Cabello & Giannopoulos, 2025; Cabello et al., 2026), among many other direction.

Algorithms with Distributional Predictions. Among the many forms of predictions considered in learning-augmented algorithms, distributional predictions have also received increasing attention. This model has been studied in several classical algorithmic settings, including binary search (Dinitz et al., 2024), ski rental (Cui & Dinitz, 2026; Kang et al., 2025; Kim & Fan, 2026), and contract scheduling (Angelopoulos et al., 2024).

Search in trees and graphs. There is also a substantial literature on search beyond paths in the absence of predictions. Onak & Parys (2006) studied search on trees and showed that centroid search uses $O(\log n)$ queries. A related problem is "Search Trees on Trees" (STT), where a distribution over the vertices is known exactly and the goal is to optimize the resulting search strategy. In this setting, Berendsohn et al. (2023) showed that weighted-centroid search gives a 2-approximation to the optimum. Related search models have also been studied on more general graphs (Emamjomeh-Zadeh et al., 2016). More recently, Dinitz & Dong (2026) studied Search on Trees with point predictions, where the algorithm is given a predicted target vertex. They showed that the $O(\log \mathrm{dist}(s,t))$ guarantee available on paths cannot hold on arbitrary trees, but that for trees of pathwidth k, a recursive spine-based algorithm achieves $O(k \log \mathrm{dist}(s,t))$ query complexity, with a matching asymptotic lower bound. The present work extends this line of study to distributional predictions, where the algorithm is given an arbitrary predicted distribution rather than a single predicted vertex.

5 Preliminaries and Notation

Given a tree T, we denote its vertex set by $V(T)$ and its edge set by $E(T)$. For vertices $u, v \in V(T)$, let $\mathrm{dist}_T(u,v)$ denote the length of the unique simple path between u and v in T. If P is a path in T and $u, v \in V(P)$, then $\mathrm{dist}_P(u,v)$ denotes the number of edges on the subpath of P between u and v.

We first formalize the oracle model used throughout the paper.

Definition 5.1 (Direction oracle)**.** *Let $T = (V(T), E(T))$ be a tree, and let $t \in V(T)$ be a hidden target vertex. The direction oracle for t is the map $\mathrm{dir}_t : V(T) \to V(T) \cup \{\textsf{here}\}$ defined as follows. If $v = t$, then $\mathrm{dir}_t(v) = \textsf{here}$. Otherwise, $\mathrm{dir}_t(v)$ is the unique neighbor u of v that lies on the simple path from v to t in T. An oracle query at v returns $\mathrm{dir}_t(v)$.*

In the *Distributional Tree Search* problem, the algorithm is given a predicted distribution $\widehat{p} \in \Delta(V(T))$, while the target is drawn from an unknown true distribution $p \in \Delta(V(T))$. The algorithm may adaptively query vertices of T through the direction oracle dir_t. The algorithm succeeds when it outputs t; equivalently, it may stop once it queries a vertex v with $\mathrm{dir}_t(v) = \textsf{here}$. The query complexity of an algorithm is the number of oracle queries it makes. When the target is clear from context, we write dir instead of dir_t.

Now we recall the standard notion of earth mover's distance (optimal transport).

Definition 5.2 (Earth Mover's Distance)**.** *A coupling of p and $\widehat{p}$ is a nonnegative matrix $\pi \in \mathbb{R}_{\geq 0}^{V \times V}$ satisfying $\sum_{y \in V} \pi(x,y) = p_x$ and $\sum_{x \in V} \pi(x,y) = \widehat{p}_y$. Let $\Pi(p, \widehat{p})$ be the set of all such couplings. The earth mover's distance in the tree metric is $W_1^T(p, \widehat{p}) := \min_{\pi \in \Pi(p,\widehat{p})} \sum_{x,y \in V} \pi(x,y) \mathrm{dist}_T(x,y)$.*

Now we consider the definition of boundary vertices of a subtree.

Definition 5.3. *The boundary set of a subtree F of T is the set $\partial F := \{x \in V(F) : x \text{ has a neighbor in } V(T) \setminus V(F)\}$.*

Finally, we will use the standard notion of distance from a node to a set of nodes as follow:

Definition 5.4. *For any $A \subseteq V(T)$ and node $t \in V(T)$, the distance from t to A is $\mathrm{dist}_T(t, A) := \min_{a \in A} \mathrm{dist}_T(t,a)$.*

6 Main Algorithm and Analysis

In this section we describe our *distributional* tree search algorithm and prove it's query complexity.

The algorithm alternates two types of work. It first makes weighted-centroid queries using the entire predicted distribution $\widehat{p}$ on the current feasible subtree. These queries rapidly reduce the absolute predicted mass of the feasible subtree and are responsible for the entropy term. It then searches geometrically from the boundary of the feasible subtree. Since a subtree can have many boundary vertices, our algorithm therefore first need to know how to efficiently compress the boundary into at most 2 nodes.

6.1 The Boundary Compression Sub-routine

Recall that a weighted centroid of a tree with nonnegative vertex weights is a vertex such that every component remaining after its deletion has at most half of the total weight. So to reduce the number of the boundary points, we could assign equal weights to all the boundary points and zero weight to all other nodes. Running weighted centroid on these nodes gives the desired result.

Lemma 6.1 (Boundary compression)**.** *Let F be a connected subtree containing t, and let $A = \partial F$. There is an adaptive procedure that, using $O(\log(|A|))$ queries, either finds t or produces a connected subtree $F' \subseteq F$ containing t and a set $A' \subseteq V(F')$ such that $1 \le |A'| \le 2$ and $\mathrm{dist}_T(t, A') \le \mathrm{dist}_T(t, A)$.*

Proof. While $|A| > 2$, assign unit weight to every vertex of A and zero weight to every other vertex of F, and query a weighted centroid v of F. If $v = t$, stop. Otherwise let y be the returned neighbor, and let C be the component of $F - v$ containing y and hence containing t. Update $F \leftarrow C, A \leftarrow (A \cap C) \cup \{y\}$.

If m is the old size of A, the centroid property gives $|A_{\mathrm{new}}| \le \frac{m}{2} + 1$, and therefore $|A_{\mathrm{new}}| - 2 \le \frac{|A_{\mathrm{old}}|-2}{2}$. Hence $O(\log(|A|))$ queries suffice to reduce A to at most two vertices.

It remains to prove the distance invariant. Let $a \in A$ be closest to t. If $a \in C$, then it remains in the new anchor set. Otherwise, the unique a–t path passes through v and then through y, so y is closer to t than a. Thus the distance from t to the anchor set never increases. □

6.2 The Distributional Algorithm

Now we give the pseudocode for our distributional tree search algorithm with predictions. For stage $i = 0, 1, 2, ...$ we do 2^i iterations of weighted centroid on the remaining feasible subtree. Then we do the boundary reduction and run 2^i steps of the point prediction algorithm. Increasing the step size geometrically, the algorithm stops until it finds the target.

See Algorithm 1 for full detail.

6.3 Analysis

Now we analyze Algorithm 1. We first note that the geometric increase in step size "absorbs" all the cost of the algorithm.

Lemma 6.2 (Geometric stage cost)**.** *Let $j \in \{0, 1, 2, ...\}$ be the first stage in which Algorithm 1 finds target t. Then the algorithm finds t using $O(2^j)$ queries.*

Proof. A query made at a vertex of the current feasible subtree can add at most one new vertex to its inner boundary, while a query outside the feasible subtree cannot increase the boundary. Hence the boundary size is at most the number of preceding queries.

Suppose inductively that $O(2^i)$ queries have been made before stage i. The bisection part and the truncated point-recovery part each use 2^i queries. The boundary has size $O(2^i)$, so the boundary reduction step of Lemma 6.1 uses $O(i)$ queries. Therefore stage i costs $O(2^i)$, and summing the geometric series through stage j shows that the algorithm finds t using $O(2^j)$ queries. □

Algorithm 1 Distributional Tree Search

$F \leftarrow T$
for stage $i = 0, 1, 2, \ldots$ **do**
  for 2^i iterations **do**
    if F is a singleton **then**
      return its unique vertex
    end if
    Choose a $\widehat{p}$-weighted centroid (or centroid if no mass is left) v of F and query v
    if the answer is here **then**
      return v
    end if
    Replace F by the component of $F - v$ indicated by the answer
  end for
  $A \leftarrow \partial F$
  Run the compression procedure of Lemma 6.1, updating F and A, until $|A| \leq 2$
  Let $U \leftarrow F$ and independently initialize the point-prediction algorithm on (U, a) for every $a \in A$
  Execute these at most two procedures in round-robin order for a total of 2^i queries
  if one of the procedures finds the target **then**
    return the target
  end if
  Replace F by the subtree of U consistent with all answers obtained during the recovery phase
end for

We next connect the running time of the algorithm to either the entropy term or the prediction error. Intuitively, the algorithm can only spend many queries on a target t if the predicted distribution is substantially inaccurate around t. If the algorithm reaches a sufficiently large stage, then the repeated weighted-centroid queries have already reduced the predicted mass of the feasible subtree containing t to below $p_t/2$. Moreover, since the recovery phase of the preceding stage still failed, the point-prediction guarantee implies that t must lie sufficiently far from the boundary of this subtree. Thus either t is found after only $O(\log(1/p_t))$ queries, or we obtain a subtree containing t that has both too little predicted mass and a sufficiently large distance from t to its boundary. These properties will allow us to charge the search cost to the earth mover's distance.

Lemma 6.3. *For every target t with $p_t > 0$, at least one of the following holds:*

(i) the algorithm finds t using $O\left(\log \frac{4}{p_t}\right)$ queries;

(ii) there is a connected subtree $S_t \subseteq T$ containing t such that $\widehat{p}(S_t) \leq \frac{p_t}{2}$ and, writing $\delta_t := \mathrm{dist}_T(t, \partial S_t)$, the algorithm finds t using $O(k \log \delta_t)$ queries.

Proof. Let j be the stage where t is found. If $2^j \leq \log \frac{4}{p_t}$, then part (i) follows immediately from Lemma 6.2.

Assume instead that $2^j > \log \frac{4}{p_t}$, which implies $2^{-2^j} < \frac{p_t}{4}$. Let S_t be the feasible subtree immediately after the weighted centroid part of stage $j-1$. Through this point the algorithm has made $1+2+\cdots+2^{j-1} = 2^j - 1$ weighted-centroid queries. Each such query halves the absolute predicted mass of the feasible subtree, so $\widehat{p}(S_t) \leq 2^{-(2^j-1)} = 2^{1-2^j} < \frac{p_t}{2}$.

Let A' be the set of at most two anchors produced by boundary compression in stage $j-1$. By Lemma 6.1, there exists some $a \in A'$ such that $\mathrm{dist}_T(t, a) \leq \mathrm{dist}_T(t, \partial S_t) = \delta_t$.

During the recovery phase of stage $j-1$, the algorithm executes the point-prediction procedures initialized from the vertices of A' in round-robin order for a total of 2^{j-1} queries. Since $|A'| \leq 2$, the procedure initialized from a receives at least $2^{j-2} - 1$ queries. Because j is the first stage in which the target is found, this procedure does not terminate within this budget.

Let $Q(a, t)$ denote the number of queries that the point prediction algorithm, initialized with prediction a, requires to find t. The preceding observation implies $Q(a, t) > 2^{j-2} - 1$. On

the other hand, simply invoking the query complexity of the point prediction algorithm gives $Q(a,t) = O(k \log \mathrm{dist}_T(a,t)) \leq O(k \log \delta_t)$. Consequently, $2^j = O(k \log \delta_t)$.

Finally, Lemma 6.2 implies that the total number of queries made before finding t is $O(2^j)$, and hence $O(k \log \delta_t)$. This proves part (ii). □

Now we show the second term of Lemma 6.3 is actually *on average* bounded by a earth mover's distance term between p and $\widehat{p}$.

Lemma 6.4 (Optimal Transport Cost)**.** *Let L be the set of targets satisfying part (ii) of Lemma 6.3, and let $\eta = W_1^T(p, \widehat{p})$. Then $\sum_{t \in L} p_t k \log \delta_t \leq 2k \log \eta$.*

Proof. Let π^* be an optimal coupling for $W_1^T(p, \widehat{p})$. For a target $t \in L$, the amount of row-t mass that can be transported to vertices inside S_t is at most the total destination mass of S_t: $\sum_{y \in S_t} \pi^*(t,y) \leq \sum_{x \in V} \sum_{y \in S_t} \pi^*(x,y) = \widehat{p}(S_t) \leq \frac{p_t}{2}$. Thus at least $p_t/2$ units of mass from t are transported outside S_t. Every path from t to a vertex outside S_t crosses ∂S_t, and hence every such destination has distance at least δ_t from t. Since the logarithm is increasing, $\sum_{y \in V} \pi^*(t,y) k \log\left(\mathrm{dist}_T(t,y)\right) \geq \frac{p_t}{2} k \log \delta_t$. Summing over $t \in L$ and using Jensen's inequality gives

$$\begin{aligned}
\sum_{t \in L} p_t k \log \delta_t &\leq 2 \sum_{x,y \in V} \pi^*(x,y) k \log\left(\mathrm{dist}_T(x,y)\right) \\
&\leq 2k \log \left(\sum_{x,y \in V} \pi^*(x,y) \mathrm{dist}_T(x,y) \right) \\
&= 2k \log \eta.
\end{aligned}$$

The sets S_t may overlap; this causes no double counting because each target is charged only through its own source row in the coupling. □

We are now finally ready to prove our main theorem.

Theorem 2.1. *There is a deterministic algorithm for the Distributional Search on Trees problem such that, for every tree T of pathwidth at most k, every true distribution p, and every predicted distribution $\widehat{p}$, the algorithm finds a target drawn from p using $O\left(H(p) + k \log \eta\right)$ queries in expectation, where $H(p)$ is the entropy of p and η is the earth mover's distance between p and $\widehat{p}$.*

Proof. Partition the targets into the two cases of Lemma 6.3. For targets of type (i), their contribution to the expected number of queries is $O\left(\sum_{t \in V} p_t \log \frac{4}{p_t}\right) = O(H(p))$.

For targets of type (ii), Lemmas 6.3 and 6.4 show that their contribution to the expected number of queries is $O\left(\sum_{t \text{ of type (ii)}} p_t k \log \delta_t\right) = O(k \log \eta)$. Adding the two bounds shows that the algorithm finds a target drawn from p using $O(H(p) + k \log \eta)$ queries in expectation. □

7 Lower Bound

In this section, we show that both terms in the upper bound from Theorem 2.1 are independently necessary. The entropy lower bound follows from classical binary search with a known target distribution. For the prediction-error term, we reduce the point-prediction problem to the distributional setting and apply the lower bound of Dinitz & Dong (2026). This reduction yields hard instances whose true distributions have zero entropy.

7.1 An Entropy Lower Bound

We first observe that the $\Omega(H(p))$ term follows immediately from the classical lower bound for binary search. Indeed, a path is a special case of a tree, and our query model on a path is precisely comparison-based binary search. Mehlhorn (Mehlhorn, 1975) showed that, even when the access distribution p is known, every binary search strategy has expected query complexity at least $\Omega(H(p))$.

Taking $\widehat{p} = p$, this immediately gives an $\Omega(H(p))$ lower bound for Distributional Search on Trees, even when the predicted distribution is perfectly accurate. It therefore remains to show that the dependence on the prediction error is also necessary, even when $H(p) = 0$.

7.2 A Prediction-Error Lower Bound

We now show that the dependence on prediction error and pathwidth is necessary even when $H(p) = 0$.

Theorem 3.1. *For every $k \geq 1$, there exists an infinite family of instances of* Distributional Search on Trees *on trees of pathwidth k and maximum degree at most 3 such that every deterministic or zero-error randomized algorithm has some instance in the family on which it requires $\Omega\left(H(p) + k \log\left(\eta\right)\right)$ queries in expectation, where $\eta = \mathrm{EMD}(p, \widehat{p})$ is the earth mover's distance between the true distribution p and the predicted distribution $\widehat{p}$. Moreover, the instances witnessing the prediction-error term can be chosen so that $H(p) = 0$.*

Proof. Fix $k \geq 1$ and an integer $\ell \geq \max\{4, k^2\}$. We use the construction of Dinitz & Dong (2026), which provides a tree $T_{k,\ell}$ of pathwidth k and maximum degree at most 3, a fixed predicted vertex s, and a set $B_{k,\ell}$ of admissible targets. Every $t \in B_{k,\ell}$ satisfies $k\ell/2 \leq \mathrm{dist}(s,t) \leq k\ell$. Let μ be the uniform distribution over $B_{k,\ell}$. Their decision-tree lower bound shows that every deterministic search algorithm C that correctly identifies every admissible target satisfies $\mathbb{E}_{t\sim\mu}[Q_C(t)] \geq ck \log \ell$, where $Q_C(t)$ is its number of oracle queries on target t, and $c > 0$ is an absolute constant.

For each $t \in B_{k,\ell}$, define a separate distributional-search instance $\mathcal{I}_t$ on $T_{k,\ell}$. Its true distribution $p^{(t)} = \delta_t$ places unit mass at t, and its predicted distribution $\widehat{p} = \delta_s$ places unit mass at s. Let ν be the distribution over these instances obtained by drawing $t \sim \mu$ and selecting $\mathcal{I}_t$. Thus, ν determines which instance is selected, while $p^{(t)}$ determines the target within the selected instance. Since the latter distribution is concentrated at a single vertex, every instance satisfies $H(p^{(t)}) = 0$ and $\eta_t := \mathrm{EMD}(p^{(t)}, \widehat{p}) = \mathrm{dist}(s,t)$.

Let A be any deterministic or zero-error randomized algorithm for Distributional Search on Trees. We obtain a point-prediction algorithm A' by giving A the tree $T_{k,\ell}$ and predicted distribution δ_s, and forwarding each query to the point-search oracle. This simulation does not require knowledge of t or $p^{(t)}$, because the true distribution is not an input to A. For every target t and every fixed choice ρ of the algorithm's random bits, the simulation produces exactly the same queries and oracle responses as running A on $\mathcal{I}_t$. In particular, $Q_{A'}(t;\rho) = Q_A(\mathcal{I}_t;\rho)$.

For a deterministic algorithm, the cited decision-tree lower bound therefore gives $\mathbb{E}_{t\sim\mu}[Q_A(\mathcal{I}_t)] \geq ck \log \ell$. For a zero-error randomized algorithm, the finite number of admissible targets ensures that, for almost every fixed choice of ρ, the simulated algorithm is correct on every target in $B_{k,\ell}$. Applying the same deterministic lower bound and then averaging over ρ gives $\mathbb{E}_{t\sim\mu}\mathbb{E}_\rho[Q_A(\mathcal{I}_t;\rho)] \geq ck \log \ell$. Equivalently, this is a lower bound on the expected query cost when the instance is drawn from ν.

By averaging, there is some $t \in B_{k,\ell}$ for which $\mathbb{E}_\rho[Q_A(\mathcal{I}_t;\rho)] \geq ck \log \ell$. Since $\ell \geq k^2$, we have $\eta_t = \mathrm{dist}(s,t) \leq k\ell \leq \ell^{3/2}$, and hence $k \log \ell \geq \frac{2}{3} k \log \eta_t$. Consequently, A requires $\Omega(k \log \eta_t)$ queries in expectation on $\mathcal{I}_t$, even though $H(p^{(t)}) = 0$. The deterministic case follows with the expectation over ρ omitted.

Allowing ℓ to range over all integers at least $\max\{4, k^2\}$ gives the required infinite family. Together with the entropy lower bound for exact predictions, this establishes that both terms in the upper bound are independently necessary. □

8 Experimental Setup

We evaluate how errors in a predicted target distribution affect the query complexity of tree search. Our experiments use four trees: two spanning trees derived from road networks and two artificial trees with prescribed pathwidth. On each tree, we gradually replace the support of the predicted distribution while keeping the true distribution fixed. This provides a controlled comparison across five levels of support mismatch and allows us to examine both the overhead of our algorithm under accurate predictions and its behavior as the prediction becomes less accurate.

Tree	Vertices	Edges	Diameter	Pathwidth	Max. degree
Luxembourg road	114,599	114,598	1,985	6	5
US road	126,146	126,145	936	7	6
Artificial, pathwidth 2	1,048,576	1,048,575	3,069	2	3
Artificial, pathwidth 3	1,050,504	1,050,503	503	3	3

Table 1: Properties of the four experimental trees. All edges have unit length, diameters are measured in edges, and all reported pathwidth values are exact.

Datasets and tree construction. The two road networks are the Luxembourg road network and the US road network covering the 48 contiguous states, obtained from Rossi & Ahmed (2015). We interpret each input as an undirected, unweighted simple graph, removing self-loops and duplicate edges. We extract its largest connected component and construct a BFS spanning tree, starting from the vertex with the smallest original identifier and visiting neighbors in increasing identifier order. This makes the tree construction deterministic. All resulting tree edges have unit length.

We also construct two artificial trees with exact pathwidths 2 and 3 using the hard instance construction by Dinitz & Dong (2026).

Table 1 summarizes the four experimental trees. The road trees provide examples derived from network data, while the artificial trees allow us to evaluate the algorithms on larger trees with explicitly controlled pathwidth.

True and predicted distributions. For a tree with n vertices, we sample $\lfloor n/2 \rfloor$ vertices uniformly without replacement and define the true distribution to be uniform on this support. Within each trial, we keep the true distribution fixed and construct predicted distributions by progressively replacing support vertices with randomly sampled vertices outside the true support. Each predicted distribution is uniform on its support, whose size remains unchanged. This produces predictions ranging from an exact match to distributions with disjoint supports.

This experiment models a distribution shift between predicted and actual target locations. A prediction may correctly identify some likely locations while assigning probability to others that do not match the true distribution. Varying the overlap between the supports provides a controlled way to adjust this mismatch and examine how search performance changes as the prediction becomes less informative. Keeping the support size and uniform weights fixed ensures that the comparison reflects changes in the predicted locations rather than changes in how concentrated the distributions are.

Algorithms and query accounting. We compare our algorithm with two versions of weighted centroid search, using either the true distribution or the predicted distribution.

The true-distribution baseline uses p as the vertex weights. At each step, it queries a weighted centroid of the current feasible subtree, so that every component remaining after removing the queried vertex contains at most half of the remaining probability mass. Unless the queried vertex is the target, the oracle response identifies the component in which the search continues. Because this baseline knows the true target distribution, it serves as an informed benchmark for evaluating the cost of using an imperfect prediction.

The predicted-distribution baseline initially follows the same procedure using the predicted weights $\widehat{p}$. While the feasible subtree contains positive predicted mass, it queries a weighted centroid and restricts the search according to the oracle response. However, an inaccurate prediction may assign zero probability to the entire remaining feasible subtree. If this happens before the target is resolved, the baseline stops using the prediction and restarts ordinary centroid search from the remaining feasible subtree tree. In this second phase, vertices receive equal weight, and each centroid query reduces the number of remaining candidate vertices.

For all methods, each direction-oracle call costs one query, and a singleton feasible subtree is resolved without an additional confirmation query. Appendix A records additional experiments details and reports the experimental results.

AI USE STATEMENT

No artificial intelligence tools were used in the preparation of this work. All ideas, proofs, and written content were developed and written independently by the authors.

REFERENCES

Priyank Agrawal, Eric Balkanski, Vasilis Gkatzelis, Tingting Ou, and Xizhi Tan. Learning-augmented mechanism design: Leveraging predictions for facility location. *Math. Oper. Res.*, 49(4):2626–2651, 2024. doi: 10.1287/MOOR.2022.0225. URL https://doi.org/10.1287/moor.2022.0225.

Spyros Angelopoulos, Marcin Bienkowski, Christoph Dürr, and Bertrand Simon. Contract scheduling with distributional and multiple advice, 2024. URL https://arxiv.org/abs/2404.12485.

Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e834cb114d33f729dbc9c7fb0c6bb607-Abstract.html.

Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching. *Information Processing Letters*, 5(3):82–87, 1976. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(76)90071-5. URL https://www.sciencedirect.com/science/article/pii/0020019076900715.

Benjamin Aram Berendsohn, Ishay Golinsky, Haim Kaplan, and László Kozma. Fast Approximation of Search Trees on Trees with Centroid Trees. In Kousha Etessami, Uriel Feige, and Gabriele Puppis (eds.), *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 19:1–19:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-278-5. doi: 10.4230/LIPIcs.ICALP.2023.19. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2023.19.

Sergio Cabello and Panos Giannopoulos. Searching in euclidean spaces with predictions. In Marcin Bieńkowski and Matthias Englert (eds.), *Approximation and Online Algorithms*, pp. 31–45, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-81396-2.

Sergio Cabello, Timothy M. Chan, and Panos Giannopoulos. Delaunay Triangulations with Predictions. In Shubhangi Saraf (ed.), *17th Innovations in Theoretical Computer Science Conference (ITCS 2026)*, volume 362 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 31:1–31:23, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-410-9. doi: 10.4230/LIPIcs.ITCS.2026.31. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2026.31.

Qiming Cui and Michael Dinitz. Ski rental with distributional predictions of unknown quality, 2026. URL https://arxiv.org/abs/2602.21104.

Mohammadreza Daneshvaramoli, Helia Karisani, Adam Lechowicz, Bo Sun, Cameron N Musco, and Mohammad Hajiesmaili. Near-optimal consistency-robustness trade-offs for learning-augmented online knapsack problems. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12459–12489. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/daneshvaramoli25a.html.

Michael Dinitz and Bob Dong. Tree search with predictions, 2026. URL https://arxiv.org/abs/2605.27490.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 10393–10406, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html.

Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, Aidin Niaparast, and Sergei Vassilvitskii. Binary search with distributional predictions. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/a4b293979b8b521e9222d30c40246911-Abstract-Conference.html.

Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pp. 519–532, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341325. doi: 10.1145/2897518.2897656. URL https://doi.org/10.1145/2897518.2897656.

Anupam Gupta, Amit Kumar, Debmalya Panigrahi, and Zhaozi Wang. *An Optimal Online Algorithm for Robust Flow Time Scheduling*, pp. 1214–1238. doi: 10.1137/1.9781611978971.47. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611978971.47.

Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 2733–2743, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/161c5c5ad51fcc884157890511b3c8b0-Abstract.html.

Bosun Kang, Hyejun Park, and Chenglin Fan. Learning-augmented ski rental with discrete distributions: A bayesian approach, 2025. URL https://arxiv.org/abs/2512.07313.

Jihwan Kim and Chenglin Fan. Robust and consistent ski rental with distributional advice, 2026. URL https://arxiv.org/abs/2603.29233.

Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In Shuchi Chawla (ed.), *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pp. 1859–1877. SIAM, 2020. doi: 10.1137/1.9781611975994.114. URL https://doi.org/10.1137/1.9781611975994.114.

Adam Lechowicz, Rik Sengupta, Bo Sun, Shahin Kamali, and Mohammad Hajiesmaili. Time fairness in online knapsack problems. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9kG7TwgLYu.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

Kurt Mehlhorn. Nearly optimal binary search trees. *Acta Informatica*, 5:287–295, 1975. doi: 10.1007/BF00264563.

Michael Mitzenmacher and Sergei Vassilvitskii. *Algorithms with Predictions*, pp. 646–662. Cambridge University Press, 2021. doi: 10.1017/9781108637435.037.

Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 379–388, 2006. doi: 10.1109/FOCS.2006.32.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pp. 9661–9670, 2018.

Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL https://networkrepository.com.

Jan van den Brand, Sebastian Forster, Yasamin Nazari, and Adam Polak. On dynamic graph algorithms with predictions. In *SODA*, pp. 3534–3557. SIAM, 2024.

A EXPERIMENTAL RESULTS

Trials and evaluation. We conduct ten independent trials on each tree. In each trial, we sample 50 targets uniformly without replacement from the true support A and use the same targets for all three methods and all five replacement levels. We report the average number of queries across the sampled targets and trials, giving 500 target evaluations per plotted point. These averages estimate expected query complexity under the true distribution. All trials and outcomes are included.

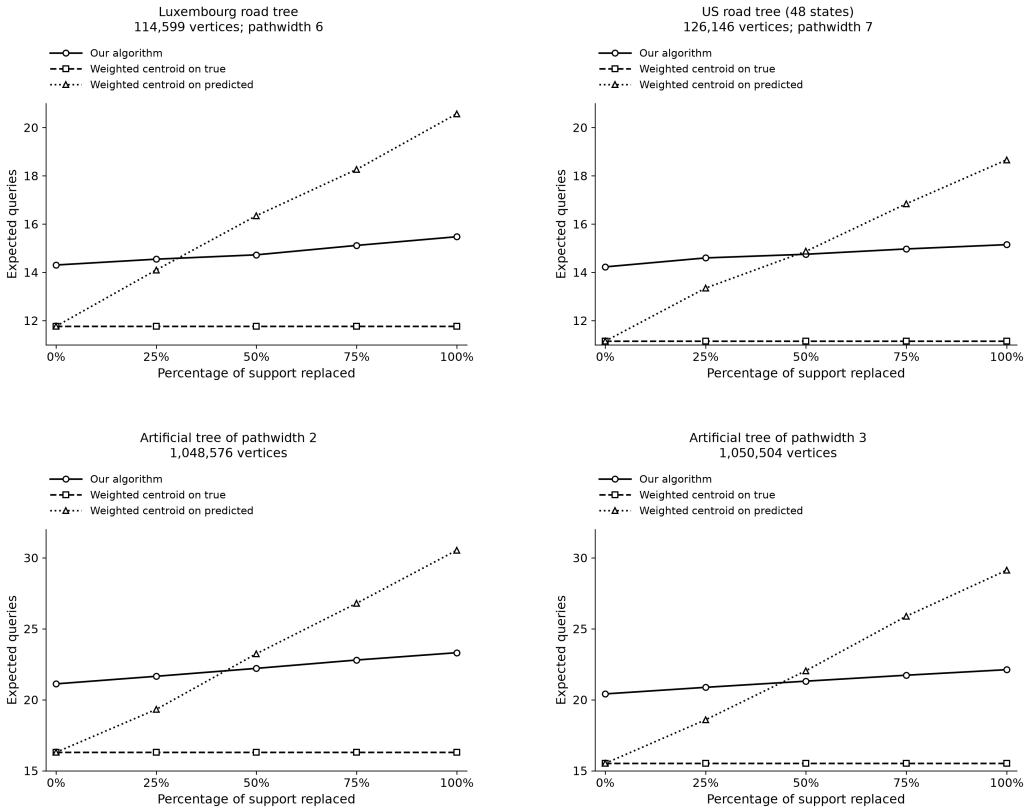


Figure 1: Expected query complexity versus percentage of support replaced. The top row shows the Luxembourg and US road trees; the bottom row shows the artificial trees of pathwidth 2 and 3.

Performance under accurate predictions. Figure 1 shows a consistent tradeoff across the four trees. With no replacement, the predicted and true distributions coincide, so the two weighted-centroid baselines have identical query counts. Our algorithm incurs additional cost from its recovery procedures. On the Luxembourg and US road trees, its mean query counts are 14.30 and 14.23, compared with 11.76 and 11.15 for weighted centroid. On the artificial trees of pathwidth 2 and 3, the corresponding values are 21.12 and 20.41, compared with 16.30 and 15.52. Across the four trees, this is an overhead of approximately 22%–32% under perfect predictions.

The predicted-centroid baseline also has a lower mean query count at 25% replacement on every tree. These observations show that our algorithm does not uniformly improve on prediction-guided centroid search: its additional search procedures have a measurable cost when predictions are accurate or only mildly perturbed.

Performance as support mismatch increases. As the replacement fraction increases, the mean query count of the predicted-centroid baseline rises more sharply than that of our algorithm. At the sampled levels of 50%, 75%, and 100% replacement, our algorithm has a lower mean query count on all four trees. The difference at 50% replacement is small for the US road tree, where the means are 14.75 and 14.87, so this point should be interpreted as a close comparison.

At complete replacement, our algorithm uses 15.48 queries on average on the Luxembourg road tree and 15.15 on the US road tree, compared with 20.57 and 18.66 for the predicted-centroid baseline. On the artificial trees of pathwidth 2 and 3, it uses 23.32 and 22.12 queries, compared with 30.54 and 29.13. These correspond to reductions of approximately 19%–25% relative to the restarting predicted-centroid baseline.

Over the full replacement range, our algorithm's mean query cost increases by approximately 6.5%–10.4%, whereas the predicted-centroid baseline increases by approximately 67.5%–87.7%. The true-distribution reference remains constant because its input distribution and evaluated targets do not change within a trial.

Across both road networks and million-vertex artificial trees, our algorithm maintains stable query costs as prediction error increases. At complete support replacement, it uses approximately 19%–25% fewer queries than the predicted-centroid baseline. This consistent performance across all four trees highlights the strength of our approach: substantial mismatch between the predicted and true distributions leads to only a limited increase in search cost.