A Unified Framework for Human-Powered Categorization

Xian Wu, Tsinghua University WUXIAN12@MAILS.TSINGHUA.EDU.CN
Jian Li LIJIAN83@TSINGHUA.EDU.CN
Guoliang Li LIGUOLIANG@TSINGHUA.EDU.CN

Abstract

We consider the problem of utilizing human power to categorize a large number of objects. In this problem, we are given an initial category hierarchy and a set of objects, and our goal is to find the most efficient strategy that asks the crowd questions in order to assign each object to its appropriate category in the hierarchy, such that the long-term expected monetary cost is minimized. We develop a *machine-crowd* hybrid framework and propose an *online* algorithm, in which we can gradually learn the category distribution and adaptively decide an effective order of asking questions. We prove that even if the true category distribution is known in advance, the problem is computationally intractable. We develop a natural approximation algorithm, and prove that it achieves an approximation factor of 2. We also show that there is a fully polynomial time approximation scheme for the problem. Furthermore, we adopt the Follow the Perturbed Leader strategy to guarantee that the framework achieves nearly the same performance as the offline optimal strategy. We evaluate the effectiveness and efficiency of our algorithms on the real-world crowdsourcing platform.

1. Introduction

During the past few years, crowdsourcing has emerged as a major technique for solving large-scale problems that are considered easy for human beings, but very difficult for computers to solve completely. In this paper, we focus on one such problem: object categorization (Parameswaran et al., 2011). We are given a set of categories and a set of uncategorized (i.e., unlabeled) objects. We would like to ask the crowd to find the most suitable category (i.e., label) for each object. Each question incurs a certain amount of monetary cost. If the number of categories is large (say $\geq 10,000$), we usually need to ask many questions to pin down the final category (we cannot possibly list all 10,000 options in a single multiple-choice question). Hence, our goal is to find the best question-asking strategy which minimizes the overall cost. We assume that the set of categories forms a *hierarchical structure*, which is known to us. This is justified in many real-world applications (for instance, “fish”, “birds” are subcategories of “animals”). Furthermore, the hierarchy naturally defines a set of very intuitive (for human beings) questions, each corresponding to a node in the hierarchy.

For example, Alice is a photographer. She once took a lot of photos during her visit to several cities in Asia. Now she wants to classify the photos into the geographical hierarchy, shown in Figure 1. Due to the large number of photos, she decides to seek help from the crowd. For each photo, each time she selects a node from the hierarchy, and asks people a multiple-choice question of the form, depicted in Figure 2. She repeats the procedure until she finally identifies the most suitable category for that photo.

Motivating Applications: Besides the above toy example, our problem applies to, but is not limited to, the following scenarios:

Product Categorization: Categories of products naturally form a hierarchical structure. Companies seek to optimize their product taxonomies, which makes their product easier for consumers to find. To achieve this, we can ask the crowd to manually categorize products following the hierarchy. More concretely, we can post a picture as well as descriptions of a product, then select a category as the question, and list all of its subcategories as options. Workers on the crowdsourcing platform should either pick a subcategory for this product, or just click the “None of the above” option indicating that the product does not belong to this category or any of its subcategories. It is worth noting that Amazon has been posting product categorization tasks on Amazon Mechanical Turk (AMT) [1], which are quite similar to what we described above.

Hierarchical Entity Resolution: Different entities, such

This paper is under review in VLDB 2016.

[1] http://www.mturk.com/

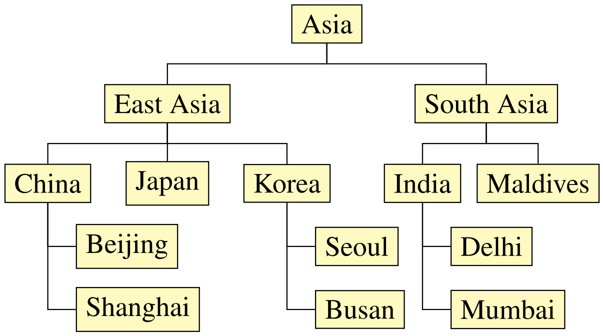
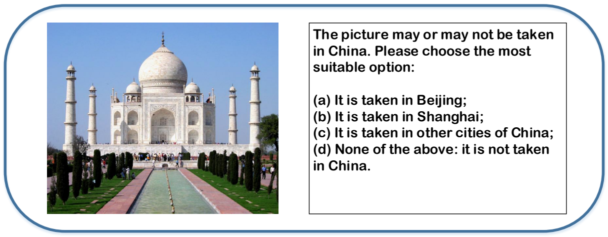


Figure 1. An example of a category hierarchy.



The picture may or may not be taken in China. Please choose the most suitable option:

(a) It is taken in Beijing;
(b) It is taken in Shanghai;
(c) It is taken in other cities of China;
(d) None of the above: it is not taken in China.

Figure 2. An example of a multiple-choice question. This question corresponds to the "China" node in the hierarchy.

as locations (Vesdapunt et al., 2014) and products (Wang et al., 2012) typically form hierarchical structures. If we further assume that the transitive closure property holds in a cluster (see Vesdapunt *et al.* (Vesdapunt et al., 2014)), we could post entity resolution tasks that allow us to identify duplicates in a *parallel* manner. Specifically, we can choose a record from one category, provide the subcategories by listing a record for each subcategory and request human workers to identify duplicates.

For instance, suppose we have known that "Tsinghua University" and "Peking University" are different universities and they both serve as subcategories an entity called "Universities in Beijing". Suppose now we want to identify "THU" (abbreviation for Tsinghua University). We shall ask the question: "Which of the following best categorizes THU? a) Tsinghua University; b) Peking University; c) Other Universities in Beijing; d) None of the above". The answer should be b) Tsinghua university. In this case, we bind "THU" into the "Tsinghua University" cluster.

Image Classification: We would like to categorize the images into the predefined category hierarchy by asking people questions depicted in Figure 2. The task is of central importance in supervised learning as it can provide a large amount labeled data for training the learning algorithm.

For example, consider the hierarchy in Figure 1 and a set of uncategorized photos. The simplest strategy is to follow the original hierarchy depicted in Figure 1. Namely, we always ask the first question corresponding to the root of the hierarchy (a question at "Asia"). Depending on the answer, we ask the second question corresponding to a node in the second level of the hierarchy. Although this strategy is fairly reasonable, there may be better strategies, depending on the distribution of categories. Consider the case where most of the photos are taken in South Asia. A better strategy would be to first ask the question at "South Asia" (this strategy incurs an average cost per photo close to 1, while the previous one at least 2).

From the above example, we can see that it is possible to design a strategy better than the one suggested by the original hierarchy, if we know the category distribution beforehand. However, in many situations, it is impossible to know the *a-priori* distribution (for example, Alice cannot remember even roughly how many pictures she took in each country). Moreover, we will see that even if we know the distribution, it is still quite challenging to come up with a plan that minimizes the total cost.

Therefore, based on the observations we made from the above examples, we propose the following natural but challenging question: *how can we construct a unified plan of asking questions to minimize the overall cost, even if we have no knowledge of the category distribution?*

Our Contributions: The technical contributions of this paper are summarized as follows:

- (Section 2) We propose a hybrid machine-crowd framework for categorizing objects in a crowdsourcing platform.
- (Section 3) We model the plan of asking questions as a decision tree construction problem. Then, we prove that the problem is NP-hard. To approximate the optimal decision tree, we develop a simple and intuitive greedy algorithm, and prove that its approximation ratio is 2. Moreover, we develop a non-trivial fully polynomial time approximation scheme (FPTAS) [2].
- (Section 4) We adopt Follow the Perturbed Leader algorithm as the online learning part. The algorithm gradually learns the true distribution of the object set and can achieve a nearly optimal performance in the worst case, comparing with the best offline optimal decision tree in hindsight.
- (Section 5) Finally, we conduct extensive evaluations on the proposed strategies and compare the performance with that of other strategies, on both synthetic

---

[2] An algorithm A is an FPTAS for an optimization problem P, if for any instance $\mathcal{I}$ and any constant $\epsilon > 0$, A computes solution S in polynomial time *w.r.t.* problem size n and $1/\epsilon$, of which the value of S satisfies the following:

$$|\mathsf{OPT} - \mathsf{val}(A)| \leq \epsilon \mathsf{OPT}$$

where OPT stands for the value of the optimal solution and $\mathsf{val}(A)$ is the solution returned by A.

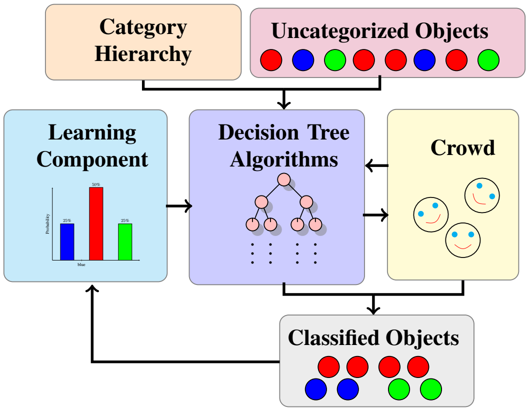


Figure 3. A hybrid machine-crowd framework.

and real data. Our results demonstrate that our strategies can lead to more than 30% cost savings on a real crowdsourcing platform.

2. Framework

2.1. Workflow

Let $\mathcal{O} = \{o_1, \ldots, o_{|\mathcal{O}|}\}$ be the set of objects. Let $\mathcal{T}$ denote the given hierarchy, which can be represented as a tree, in which each node corresponds to a distinct category. With slight abuse of notation, for a node $u \in \mathcal{T}$, we also use u to represent the corresponding category. For any object $o \in \mathcal{O}$, we denote $\mathsf{tar}(o)$ as its target category. For each node $u \in \mathcal{T}$, we use $\mathcal{T}_u$ to denote the subtree rooted at u. Denote by $\mathsf{child}_{\mathcal{T}}(u)$ the set of children of u. Denote by $\mathsf{father}_{\mathcal{T}}(u)$ the father node of u and let $\mathsf{root}(\mathcal{T})$ denote the root of $\mathcal{T}$.

As shown in Algorithm 1, the *hybrid machine-crowd* framework works as follows: Each time the framework takes in an uncategorized object from $\mathcal{O}$ (Line 5). It follows the strategy constructed by the optimization component, proceeds with iteratively generating new tasks and collecting answers from the workers on the crowdsourcing platform, until the most suitable category for the object is obtained (Lines 6-9). The learning component then re-estimates the distribution based on the results of the categorized objects (Line 10). Finally the learning component notifies the optimization component of the possibly changed distribution so that it can adjust the plan of asking questions, correspondingly, hoping to minimize the cost (Line 11).

Definition 1. *(Multiple-choice Question) Given an object $o \in \mathcal{O}$ and an* internal *node $u \in \mathcal{T}$, we request human to answer a multiple-choice question, which corresponds to node u, by selecting an option from the following ones:*

- For each $v \in \mathsf{child}_{\mathcal{T}}(u)$, we have an option: The object belongs to category v (i.e., $\mathsf{tar}(o) \in \mathcal{T}_v$);
- The object does not belong to any of the children hierarchy of u, but does belong to category u itself (i.e., $\mathsf{tar}(o) = u$);
- ("None of the above" option) The object does not belong to $\mathcal{T}_u$.

Algorithm 1 Framework($\mathcal{O}, T$)

```
 1: Input: $\mathcal{O}, \mathcal{T}$
 2: $\mathcal{L} \leftarrow \emptyset$
 3: Set the initial category distribution
 4: Generate a plan $\mathcal{P}$ of asking questions
 5: while there is an uncategorized object $o$ in $\mathcal{O}$ do
 6:     while $o$ is not categorized do
 7:         Ask a question to the crowd following $\mathcal{P}$
 8:         Collect answers from the crowd
 9:     end while
10:     $\mathcal{L} \leftarrow \mathcal{L} \cup \{(o, \mathsf{tar}(o))\}$
11:     Re-estimate the category distribution
12:     Reconstruct the plan $\mathcal{P}$
13: end while
14: Output: $\mathcal{L}$
```

For the second type of options, the object is labeled by u as the most suitable category. For the first or third type, we may have to proceed with further questions in order to find the most suitable category for it.

Remark: Some nodes in the hierarchy may have a large number of children (for instance, the "Books" category in Amazon has more than 30 subcategories). Therefore, it is impossible to list all of its children as options in one human intelligence task (HIT). However, in practice, options can often be grouped together into a cluster (for instance, books about "Military history", "United States History" and "History" can be grouped into "History"). In this manner, if we decide to ask a question at this node, we could first list the clusters as options, which essentially reduces the number of options for this node. Based on the answer from the crowd, we could ask a question at the resulting cluster, and list the original options that are grouped into the cluster.

Example 1. *The question corresponding to node "China" from Figure 1 is shown in Figure 2. A photo of* Tsinghua University *should lead to the answer (a) Beijing, i.e., the most suitable category is "Beijing" for the photo.*

If Alice posts a photo of Terracotta Warriors and Horses, *she would receive the answer (b), which means that the most suitable category for this photo is "China". In fact, the photo was taken in Xi'an, which is a city in China, but not included in the given hierarchy. Thus "China" should be the most suitable category for this photo.*

Finally, if the photo is about Mount Fuji, *the right option would be (d) "None of the above". Then a couple more* *questions are needed to finally categorize that picture.*

Our framework works in the online setting, in the sense that the question-asking strategy for the current object should only depend on the objects we have processed so far, but not those unprocessed ones. We do not even assume that all objects are given before we run our crowd categorization algorithm. They may arrive in an online streaming fashion.

For each object, our question-asking strategy can be *adaptive*. Namely, the next question we choose to ask may depend on the outcomes of the previous questions. In next section, we show this is in fact equivalent to constructing a new decision tree based on the original given hierarchy.

Assume that each question costs a fixed price (for instance, \$0.02 per question), the total *monetary* cost is proportional to the total number of questions in need for categorizing all objects in $\mathcal{O}$. Hence, our goal is to minimize the total number of questions.

Problem 1 (Cost Minimization Problem)**.** *Find an online adaptive question-asking strategy to categorize all objects in $\mathcal{O}$ such that the total number of questions is minimized.*

2.2. Adaptive Strategies and Decision Trees

To categorize an object, we first select a node from the given hierarchy $\mathcal{T}$ as the question. Based on different outcomes, the node partitions $\mathcal{T}$ into several subtrees, one of which includes the most appropriate category for the object. Let us examine the effect of one question closely, in graph theoretic terms. Suppose the question corresponds to the internal node $u \in \mathcal{T}$. Removing all edges incident on u in $\mathcal{T}$, we obtain several subtrees:

1. One subtree for each child of u, which corresponds to the first type of options in Definition 1.
2. Node u itsef, which corresponds to the second option.
3. One subtree containing the parent of u, which corresponds to the "None of the above" option.

We use $\Phi_{\mathcal{T}}(u)$ to denote the set of subtrees obtained above. Hence, the answer to the question can help us to identify which subtree in $\Phi_{\mathcal{T}}(u)$ contains the target category. Any reasonable adaptive strategy should ask the next question corresponding to a node from the resulting subtree. [3] Repeating the process, we can identify the best category until only one node is left.

Such adaptive strategy can be modeled as a *decision tree*, in which each internal node is a (multiple-choice) question (say it corresponds to $u \in \mathcal{T}$), whose ancestors are the previous questions, and each of its children corresponds to a question in the set $\Phi_{\mathcal{T}}(u)$ of subtrees. All leaves are target categories. Given a leaf, the root-to-leaf path in the tree represents the series of questions we ask if the target category corresponds to that leaf, and the cost spent for this object is the depth of the leaf (i.e., the length of the path minus 1). Please refer to Figure 4 for an example.

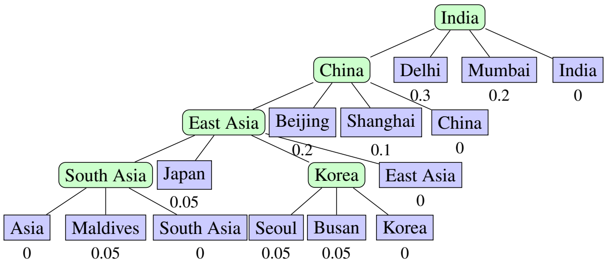


Figure 4. Optimal decision tree to the category tree in Figure 1. Question nodes are represented by green round rectangles and leaves (resulting categories) are represented by purple rectangles. Category probabilities are labeled beside the corresponding leaves.

Now, we are ready to formally define the decision trees corresponding to adaptive strategies.

Definition 2 ((Decision Tree))**.** *Given the initial hierarchy $\mathcal{T}$, a* decision tree w.r.t. $\mathcal{T}$ *is a decision tree $\mathcal{D}$, which satisfies the following properties:*

- *Each leaf of $\mathcal{D}$ is a category in $\mathcal{T}$;*
- *Each internal node v of $\mathcal{D}$ corresponds to an internal node $u \in \mathcal{T}$. v partitions T into $\Phi_{\mathcal{T}}(u)$, and thus it has $|\Phi_{\mathcal{T}}(u)|$ children. Each such child w corresponds to a subtree in $\Phi_{\mathcal{T}}(u)$, and $\mathcal{D}_w$ (i.e., the subtree of $\mathcal{D}$ rooted at w) represents the decision tree for the corresponding subtree.*

For ease of presentation, for each internal node $u \in \mathcal{D}$, we fix its *leftmost* subtree as the decision tree for the subtree corresponding to the "None of the above" option in Definition 1. We fix its *rightmost* child as the category corresponding to the question (i.e., the second option). For instance, in Figure 4, we place questions at "China" as the leftmost child of the question at "India" and we place the category "India" as the rightmost child of it.

Let $\mathsf{leftpath}(\mathcal{D})$ denote the leftmost path of $\mathcal{D}$. Define $\mathsf{Lvs}(\mathcal{D})$ ($\mathsf{Int}(\mathcal{D})$) as the set of leaves (internal nodes) of $\mathcal{D}$. With slight abuse of definition, for any $u \in \mathcal{T}$, we also use u to denote the question (internal) node in D asked at u [4]. For any category u, let $\mathsf{leaf}_{\mathcal{D}}(u)$ denote the leaf in $\mathcal{D}$ that corresponds to u. For any $v \in \mathcal{D}$, denote by $\mathsf{dep}_{\mathcal{D}}(v)$ the depth of v in $\mathcal{D}$.

To define the cost of a decision tree, it is convenient to define the category distribution, as follows: Let $\Pr$ be the

---

[3] Assuming the crowd users do not make mistakes, it is easy to see that asking a question in any other subtree does not help, since we can only obtain the "None of the above" answer from that question.

[4] In the paper, for any category u, we use u as its corresponding node in $\mathcal{T}$, and a question node asked at u in $\mathcal{D}$.

probability distribution over category set V, defined by

$$\Pr(u) \triangleq |\{\mathsf{tar}(o) = u \mid o \in \mathcal{O}\}|/|\mathcal{O}|.$$

We call Pr the the category distribution. For $\mathcal{T}' \subseteq \mathcal{T}$, we define $\Pr(\mathcal{T}') \triangleq \sum_{u \in \mathcal{T}'} \Pr(u)$. Now, we can define our cost metric.

Definition 3. *(Cost Metric) Given a decision tree $\mathcal{D}$, the expected cost, denoted by* $\mathsf{cost}(\mathcal{D})$*, is*

$$\mathsf{cost}(\mathcal{D}) \triangleq \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \mathsf{dep}_{\mathcal{D}}(\mathsf{tar}(o)) = \sum_{u \in \mathsf{Lvs}(\mathcal{D})} \Pr(u)\, \mathsf{dep}_{\mathcal{D}}(u) \tag{1}$$

Problem 2. *(Min-cost Decision Tree (*MDT*) Problem) Given $(\mathcal{T}, \Pr)$, construct a decision tree $\mathcal{D}^*$ w.r.t. $\mathcal{T}$ with the minimum expected cost.*

Remark: By the definition of a decision tree, we can treat the initial category hierarchy $\mathcal{T}$ as a slightly "incomplete" decision tree. The only difference is that any internal node in a decision tree has a child leaf corresponding to the second type of options (in Definition 1). Therefore, we can easily extend $\mathcal{T}$ to a decision tree. In this decision tree, we always start from the root of the hierarchy. Therefore, the "None of the Above" answer to any question is unnecessary since it would always lead to an empty set. For instance, consider a photo of the Great Wall and the hierarchy in Figure 1. Alice can consecutively ask questions at "Asia", "East Asia", "China" and finally obtain "Beijing" as its most suitable category.

Example 2. *For an object set with the category distribution shown in Figure 4, the decision tree directly obtained from Figure 1 incurs an expected cost of 2.9, while the expected cost of the optimal decision tree $\mathcal{D}^*$ for this hierarchy is 1.85, depicted in Figure 4.*

3. Decision Tree Construction

In this section, we focus on the MDT problem. We assume the category probability Pr is known to us throughout this section. We first prove that constructing the optimal decision tree is computationally intractable. Then we propose a natural greedy approach and analyze the approximation ratio for the algorithm. Finally, we propose a non-trivial FPTAS for the problem.

3.1. NP-Hardness

To prove that constructing an optimal decision tree is NP-complete, we utilize a non-trivial polynomial time reduction from the Exact Cover by 3-Sets with multiplicity 3 problem (X3C-3), defined as follows:

Definition 4 (X3C-3)**.** *An instance of the X3C-3 problem $\mathcal{I} = (X, \mathcal{Y})$ contains the following:*

(a) a finite set X with $|X| = 3q$ for $q \in \mathbb{N}^+$;

(b) a collection $\mathcal{Y}$ of 3-element subsets of X, i.e., $\mathcal{Y} = \{Y_1, \ldots, Y_{|\mathcal{Y}|}\}$, and for each $i = 1, \ldots, |\mathcal{Y}|$, it holds that $Y_i \subseteq X$ and $|Y_i| = 3$.

(c) each element $x \in X$ appears in at most 3 sets of $\mathcal{Y}$.

Definition 5 (X3C-3 Problem)**.** *Given an instance $\mathcal{I} = (X, \mathcal{Y})$, the X3C-3 problem decides whether $\mathcal{Y}$ contains an exact cover for X, i.e., to find a subcollection $\mathcal{Y}' \subseteq \mathcal{Y}$ such that members of $\mathcal{Y}'$ form a disjoint partition of X.*

The X3C-3 problem is known to be NP-complete (Cicalese et al., 2011). We now state the following theorem. Please refer to Appendix A for the complete proof.

Theorem 1. MDT *problem is* NP*-complete.*

3.2. Greedy Strategy

To minimize the expected cost, a "good" strategy should first ask questions that are capable of partitioning the category hierarchy fairly evenly (w.r.t. the distribution Pr). Intuitively, this could avoid the case where a high probability node has a large depth (which would incur a large cost). For example, if a node u has probability $\Pr(u) > 0.9$, we probably want to first ask the question that can directly lead us to u (i.e., the question at node u if u is an internal node of $\mathcal{T}$, or $\mathsf{father}_{\mathcal{T}}(u)$ if u is a leaf).

A heuristic approach is to choose a question for some category which makes the heaviest resulting subtree as light as possible. Formally, Given an instance $(\mathcal{T}, \Pr)$ to the problem, we choose an internal node u_0 such that

$$u_0 = \arg \min_{u \in \mathsf{Int}(\mathcal{T})} \max\{Pr(\mathcal{T}') | \mathcal{T}' \in \Phi_{\mathcal{T}}(u)\} \tag{2}$$

For ease of analysis, we assume that distinct subtrees of $\mathcal{T}$ have different probabilities, to guarantee the uniqueness of each question.

Remarks: (1) This natural greedy algorithm is much similar to the classic Huffman Algorithm, not only in the problem formulation, but in the sense that they both try to *balance* the decision tree. The main difference is that our approach is to construct a tree from top to down while the Huffman tree is constructed in a bottom-up manner.

(2) Prior to our work, Parameswaran *et al.* (Parameswaran et al., 2011) proposed a graph-based categorization problem, in which they asked *yes-or-no* questions and developed algorithms to rule out as many candidates as possible in each phase *locally* in order to save money. However, in this paper, we focus on minimizing the overall cost, which can be seen as a *global* objective. Moreover, they assumed that the category distribution was *uniformly* distributed, while we do not require any knowledge of the category distribution.

The approximation ratio of the greedy algorithm is guaranteed by the following theorem:

Theorem 2. *Given any instance* $(\mathcal{T}, \Pr)$ *to the* MDT *problem, let* $\mathcal{D}$ *denote the decision tree constructed by* Greedy. *It holds that*

$$cost(\mathcal{D}) \leq 2cost(\mathcal{D}^*) \tag{3}$$

where $\mathcal{D}^*$ *is the optimal decision tree to the problem.*

3.3. FPTAS

In this section, we present an FPTAS for the problem, which can achieve $(1+\epsilon)$ approximation ratio for any $\epsilon > 0$. For this purpose, we first develop an exact pseudo-polynomial time algorithm based on dynamic programming (DP). Then we show that by rounding the probability of each node in the tree, we can transform the DP into an FPTAS.

Due to space constraints, we only state the key results in the section. Please refer to Appendix C for full details.

Theorem 3. *Given any instance* $(\mathcal{T}, \Pr)$*, there exists a pseudo-polynomial time dynamic programming-based algorithm that constructs the optimal decision tree.*

Theorem 4. *Given any instance* $(\mathcal{T}, \Pr)$*, there exists an FPTAS for the MDT problem.*

4. Learning A-Priori Probabilities

In this section, we show how the learning component in Figure 3 works. Our goal is to develop an online learning algorithm which can learn the category distribution along the way so that the overall long-term cost is close to the offline optimal cost. By combining the learning component with the decision tree component, each time we obtain new categorized objects from the decision tree component, we re-estimate the category distribution in the learning component. We can *adaptively* modify the decision tree, and possibly propose a new decision tree for further object inputs, without knowing their actual distribution in advance.

We adopt Follow the Perturbed Leader(FPL) strategy: we maintain the fraction of each node in the hierarchy. Each time we estimate the probability as the fraction plus an exponentially distributed perturbation (Kalai & Vempala, 2005).

The workflow of the algorithm is depicted in Algorithm 2. In each loop, we take in an uncategorized object, we modify the weights of each node by adding a random variable $\sim \mathsf{exp}(\lambda)$ (Lines 3-5) and normalize the weights as probabilities in Line 6. Then we generate the decision tree using the greedy algorithm or the FPTAS in Section 3 (Line 7). We iteratively ask a question and collect an answer following the decision tree, until we finally obtain the target category (Lines 8-9). Finally, we update the corresponding weight in Line 10.

Algorithm 2 PerturbedLeader$(T, \mathcal{O}, \lambda)$

```
1:  Input: $T, \mathcal{O}, \epsilon$
2:  while there is an uncategorized object in $o \in \mathcal{O}$ do
3:      for $u \in T$ do
4:          choose $q(u) \sim \mathsf{exp}(\lambda)$
5:          $w'(u) \leftarrow w(u) + q(u)$
6:      end for
7:      Normalize the probability of each node by $w'$.
8:      Construct the decision tree using Greedy (or the FPTAS).
9:      Ask questions on the crowdsourcing platform based
        on the decision tree
10:     Collect the target category $\mathsf{tar}(o)$
11:     $w(\mathsf{tar}(o)) \leftarrow w(\mathsf{tar}(o)) + 1$
12: end while
```

In fact, Algorithm 2 realizes the hybrid machine-crowd framework in Section 2. The time cost for categorizing the object set consists of two parts: (a) the total latency incurred on the crowdsourcing platform; (b) the running time of the decision tree construction algorithm and FPL, $O(n^2)$ per object for Greedy and $poly(n/\epsilon)$ for the FPTAS given $\epsilon > 0$.

Theorem 5. *Given any object set* $\mathcal{O}$ *and any constant* $\lambda > 0$*,* PerturbedLeader *(using* FPTAS *in step 7) achieves the following:*

$$\mathbb{E}\{cost\} \leq (1+\lambda)\mathsf{offline}^*(\mathcal{O}) + O\left(\frac{n^2 \ln n}{\lambda}\right) \tag{4}$$

where $\mathbb{E}\{\mathsf{cost}\}$ *is defined as the expectation of our online cost and* $\mathsf{offline}^*(\mathcal{O})$ *is the offline optimal total cost.*

5. Experimental Evaluation

In this section, we perform experimental study on our techniques. Due to limited space, the synthetic data evaluation is deferred to Appendix F.1.

5.1. Real Data Evaluation

The settings of our experiments on AMT are detailed as follows:

- We pruned the "artifact, artefact" sub-hierarchy of ImageNet [5] (Deng et al., 2009), which is a large-scale hierarchical image dataset of over 15 million labeled images.. There are 687 nodes in the resulting hierarchy. The height of the hierarchy is 9. The maximum degree is 21 and the average degree is 2.9.

---

[5] `http://image-net.org/`

- 200 pictures were generated from ImageNet. 80% of them belong to the sub-hierarchy of “conveyance, transport”, and the others (20%) are artifacts belonging to other categories.
- We asked the crowd to categorize 50 photos in parallel and re-estimated the distribution after all the 50 photos were categorized.
- Four questions were bathced into one HIT. One HIT was replicated three times and we paid workers five cents for completing each replication. We adopted majority voting to decide the final answers to each HIT.
- Workers, who had at least 200 HITs approved with at least 90% accuracy in history, were qualified for completing our tasks.
- We implemented the following algorithms:
    - Breadth-first This algorithm asks questions for the first k internal nodes encountered in a *weighted breadth-first* traversal starting from the root. By “weighted”, we mean that after visiting a node u, we add all its internal children v into the queue by the descending order of the total weight of $\mathcal{T}_v$.
    - Greedy: The natural greedy algorithms we proposed in this paper, of which the strategy is to choose k questions to minimize the heaviest resulting group.

In ImageNet, each image belongs to exactly one category. However, we found in many cases that an image may as well belong to some other categories (in particular, the parent category, sibling categories or the child categories). For instance, there are a number of images in ImageNet under “car, automobile”, which should be best categorized by “sports car”. Therefore, we define the following quality measure, based on the similarity among categories: We use $\mathsf{sim}(u, v)$ to denote the similarity score between two nodes in $\mathcal{T}$. $\mathsf{sim}(u, v) = 1$ for all $u \in \mathcal{T}$ and its descendant v. $\mathsf{sim}(u, v) = 0.8(\text{or } 0.4, 0.1)$ if u is the direct parent (2-hop, 3-hop parent) of v, and $\mathsf{sim}(u, v) = \mathsf{sim}(w, u) \times \mathsf{sim}(w, v)$ where $w = \mathrm{LCA}(u, v)$ is the least common ancestor of u and v. This means that, for example, if an object is associated with category $u \in \mathcal{T}$ and a worker labels it as the parent of u, we get a score of 0.8. [6]

Greedy results significant cost savings while maintains the same quality, compared with Breadth-first.

We first constructed Greedy and Breadth-first given the *a-priori* probability distribution of the object set. From Table 1, we can see that the number of questions asked using Greedy is 728, while Breadth-first asked 1240 questions. That amounts to a 41.3% saving, without sacrificing the quality.

Algorithm	Number of Questions	Quality Score
Greedy	728	0.78
Breadth-first	1240	0.79
FPL with Greedy	854	0.82

Table 1. Comparisons of different algorithms

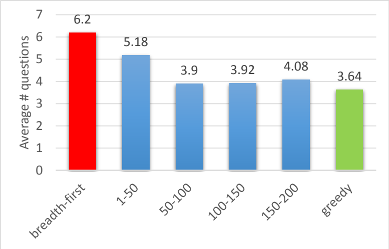


Figure 5. Average number of questions using FPL (for different batches of images) (blue), Breadth-first (red) and Greedy (green).

(1) Our framework can quickly learn the category distribution and construct an efficient strategy; (2) Our framework can achieve a high categorization quality.

We next examined the effectiveness of the framework. As depicted in Figure 5, when the framework took in the first batch of images, for each image, it took 5.18 questions for the crowd to find the most suitable category. After the first batch of images were categorized, FPL tried to learn the *a-priori* distribution and the framework adaptively constructed an efficient strategy by Greedy for the subsequent batches of images. The average number of questions for each image remains around 4. The overall questions asked by the framework is comparable to that of Greedy with the exact knowledge of the *a-priori* distribution, and outperforms Breadth-first.

Somewhat surprisingly, from Table 1, we see that by using the framework, the quality of the categorization is pretty high (0.85), beating the quality of Greedy (0.78) and Breadth-first (0.79).

6. Related Work

For crowdsourcing applications, there are three main issues: how to guarantee the quality of the answers, how to reduce the monetary cost and how to reduce the latency. Gao *et al.* (Gao & Parameswaran, 2014) developed pricing algorithms that could dynamically adjust prices for each task to meet the deadline while minimizing the cost. Parameswaran *et al.* (Parameswaran et al., 2012; 2014) proposed a filtering framework in order to minimize cost,

---

[6] There are a few pairs of categories that are quite similar but far away in the given ImageNet hierarchy. For instance, the “cap” and “baseball cap” synsets. We plan to address such issues systematically in the future work.

while to control the latency as well as the accuracy. Karger *et al.* (Karger et al., 2011) proposed iterative algorithms for efficiently allocating resources on noisy data gathered from the crowd. Tran-Thanh *et al.* (Tran-Thanh et al., 2013) later derived an algorithm that significantly improved their theoretical guarantee.

Researchers have proposed several methods for improving the efficiency of the crowdsourcing systems, by assigning only "necessary" questions to human, and leaving the mechanical jobs to computers (Das Sarma et al., 2014; Boim et al., 2012; Parameswaran et al., 2011; Wang et al., 2013; Zhang et al., 2014b). In real world crowdsourced applications where multiple rounds/batches of questions are asked, there exist two approaches. One is to *maximize the information gain in each round* (Parameswaran et al., 2011; Whang et al., 2013; Zhang et al., 2014b;a). The approach works in Bayesian settings and greedily selects the most "desirable" questions. For instance, Zhang *et al.* (Zhang et al., 2014a) models the question selection problem as maximizing information gain using *entropy*. They (Zhang et al., 2014b) later explored the submodularity of the entropy functions when generating multiple questions for one batch. The other approach tries to directly minimize the *overall* cost (Kaplan et al., 2013; Vesdapunt et al., 2014; Wang et al., 2013). The resulting combinatorial optimization problems are often computationally intractable. In addition to the two approaches, some machine-crowd hybrid approaches have been proposed to further reduce the amount of tasks assigned to human (Whang et al., 2013; Zhang et al., 2014a).

Comprehending images and natural languages are fairly easy for human beings, but in contrast hard for computers. Recently Parameswaran *et al.* (Parameswaran et al., 2011) proposed a model of utilizing human power to categorize object in a graph. Their methods could be used for collecting large-scaled data, which is of crucial importance for machine learning algorithms. Singla *et al.* (Singla et al., 2014) tried to teach the crowdsourcing workers to categorize objects in order to improve the effectiveness and accuracy. Sun *et al.* (Sun et al., 2015) utilized human power to build the hierarchy of concepts which used to be expensive and time-consuming even for experts to construct.

Decision Trees: Decision trees have been used to model adaptive question-asking strategies (Kaplan et al., 2013). There are also several work on constructing decision trees with various objectives (Cicalese et al., 2010; 2011; cic, 2014; Gupta et al., 2010). In particular, the algorithm in (cic, 2014) could be used directly for approximating the optimal plan for yes-or-no questions in (Parameswaran et al., 2011), in which the categories form a hierarchy and the problem is to construct a cost-saving decision tree by asking yes-or-no questions. Cicalese *et al.* (Cicalese et al., 2011) first proved that it is NP-hard to construct an optimal decision strategy. Later, they (cic, 2014) developed an FPTAS for trees with bounded degree, based on dynamic programming. They also analyzed a natural greedy algorithm, and proved the approximation ratio is $\frac{1+\sqrt{5}}{2}$. Our MDT problem is different from theirs, and their results cannot be used directly. Nevertheless, our NP-hardness reduction and approximation algorithms are heavily inspired by the above line of work and we adopt several proof ideas there.

Learning and Prediction: For the learning component of our framework, we directly adopt the results of Kalai *et al.* (Kalai & Vempala, 2005), in which they analyzed the performance of FPL in problems with linear generalization. Herbster *et al.* (Herbster & Warmuth, 1998) developed strategies for *tracking* the best linear predictor.

7. Future Work

There are several interesting future directions we would like to pursue. One important direction is to incorporate various machine learning techniques into our framework. Machine learning algorithms can be used to classify certain instances that are amenable to algorithms, or to make preliminary classification to reduce the search space for human workers, which could potentially bring further savings. On the theoretical side, the approximability for constructing the optimal decision tree in the multiple-question setting remains an open problem . We suspect that the natural greedy algorithms can still achieve a constant approximation ratio.

References

Improved approximation algorithms for the average-case tree searching problem. *Algorithmica*, 2014.

Boim, Rubi, Greenshpan, Ohad, Milo, Tova, Novgorodov, Slava, Polyzotis, Neoklis, and Tan, Wang Chiew. Asking the right questions in crowd data sourcing. In *ICDE*, 2012.

Cicalese, Ferdinando, Jacobs, Tobias, Laber, Eduardo Sany, and Molinaro, Marco. On greedy algorithms for decision trees. In *ISAAC*, 2010.

Cicalese, Ferdinando, Jacobs, Tobias, Laber, Eduardo, and Molinaro, Marco. On the complexity of searching in trees and partially ordered structures. *Theor. Comput. Sci.*, 2011.

Das Sarma, A., Parameswaran, A., Garcia-Molina, H., and Halevy, A. Crowd-powered find algorithms. In *ICDE*, 2014.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

Gao, Yihan and Parameswaran, Aditya. Finish them!: Pricing algorithms for human computation. *PVLDB*, 2014.

Gupta, Anupam, Nagarajan, Viswanath, and Ravi, R. Approximation algorithms for optimal decision trees and adaptive tsp problems. In *Automata, Languages and Programming*, pp. 690–701. 2010.

Herbster, Mark and Warmuth, Manfred K. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

Kalai, Adam and Vempala, Santosh. Efficient algorithms for online decision problems. *J. Comput. System Sci.*, 2005.

Kaplan, Haim, Lotosh, Ilia, Milo, Tova, and Novgorodov, Slava. Answering planning queries with the crowd. *PVLDB*, 2013.

Karger, David R, Oh, Sewoong, and Shah, Devavrat. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 2011.

Kundu, Sukhamay and Misra, Jayadev. A linear tree partitioning algorithm. *SIAM J. Comput.*, 1977.

Parameswaran, Aditya, Sarma, Anish Das, Garcia-Molina, Hector, Polyzotis, Neoklis, and Widom, Jennifer. Human-assisted graph search: It's okay to ask questions. *PVLDB*, 2011.

Parameswaran, Aditya, Boyd, Stephen, Garcia-Molina, Hector, Gupta, Ashish, Polyzotis, Neoklis, and Widom, Jennifer. Optimal crowd-powered rating and filtering algorithms. *PVLDB*, 2014.

Parameswaran, Aditya G., Garcia-Molina, Hector, Park, Hyunjung, Polyzotis, Neoklis, Ramesh, Aditya, and Widom, Jennifer. Crowdscreen: Algorithms for filtering data with humans. In *SIGMOD*, 2012.

Simpson, Edwin, Venanzi, Matteo, Reece, Steven, Kohli, Pushmeet, Guiver, John, Roberts, Stephen, and Jennings, Nick. Language understanding in the wild: Combining crowdsourcing and machine learning. In *WWW*, 2015.

Singla, Adish, Bogunovic, Ilija, Bartók, Gábor, Karbasi, Amin, and Krause, Andreas. Near-optimally teaching the crowd to classify. In *ICML*, 2014.

Sun, Yuyin, Singla, Adish, Fox, Dieter, and Krause, Andreas. Building hierarchies of concepts via crowdsourcing. In *IJCAI*, 2015.

Tran-Thanh, Long, Venanzi, Matteo, Rogers, Alex, and Jennings, Nicholas R. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *AAMAS*, pp. 901–908, 2013.

Vesdapunt, Norases, Bellare, Kedar, and Dalvi, Nilesh. Crowdsourcing algorithms for entity resolution. *PVLDB*, 2014.

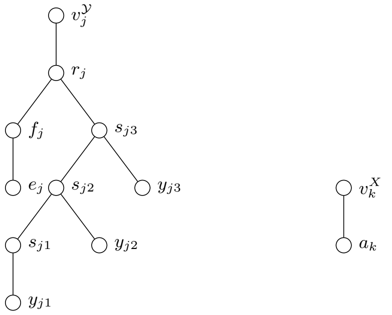
Wang, Jiannan, Kraska, Tim, Franklin, Michael J, and Feng, Jianhua. Crowder: Crowdsourcing entity resolution. *PVLDB*, 2012.

Wang, Jiannan, Li, Guoliang, Kraska, Tim, Franklin, Michael J., and Feng, Jianhua. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.

Whang, Steven Euijong, Lofgren, Peter, and Garcia-Molina, Hector. Question selection for crowd entity resolution. *PVLDB*, 2013.

Zhang, Chen Jason, Chen, Lei, and Tong, Yongxin. Mac: A probabilistic framework for query answering with machine-crowd collaboration. In *CIKM*, 2014a.

Zhang, Chen Jason, Tong, Yongxin, and Chen, Lei. Where to: Crowd-aided path selection. *PVLDB*, 2014b.



(a) Component for $\tilde{Y_j}$ ($T_{v_j^{\mathcal{Y}}}$) (b) Component for X_k ($T_{v_k^X}$)

Figure 6. Input tree components.

A. Hardness

A.1. Preliminaries

Recall that in the X3C-3 problem $(X, \mathcal{Y})$, we would like to investigate whether $\mathcal{Y}$ includes a partition of X, i.e., whether there exists a sub-collection $\mathcal{C} \subseteq \mathcal{Y}$, such that $3|\mathcal{C}| = |\mathcal{X}|$ and $\cup_{Y \in \mathcal{C}} Y = X$.

Order of X3C-3 Instance: To begin, we first define an order $<$ among the elements of X: $x_1 < x_2 < \ldots < x_{|X|}$. We then extend this and define an order $\prec$ on $X \cup \mathcal{Y}$ (Cicalese et al., 2011) as follows:

- for any two sets $A = \{a_1, a_2, a_3\} \in \mathcal{Y}$ ($a_1 < a_2 < a_3$), $B = \{b_1, b_2, b_3\} \in \mathcal{Y}$ ($b_1 < b_2 < b_3$), it holds that $A \prec B$ *iff.* $a_3a_2a_1 < b_3b_2b_1$ in the lexicographical order;
- for any $x \in X$, $x \prec A$ *iff.* xx_1x_1 is lexicographically smaller than $a_3a_2a_1$.

Given $(X, \mathcal{Y})$, we order $\mathcal{Y} = \{Y_1, \ldots, Y_{|\mathcal{Y}|}\}$ such that $Y_1 \prec \ldots \prec Y_{|\mathcal{Y}|}$. Denote by $\Pi = (\pi_1, \pi_2, \ldots, \pi_{|X|+|\mathcal{Y}|})$ the ascending order of elements of $X \cup \mathcal{Y}$. For any $j \in [|\mathcal{Y}|]$, define $Y_j \triangleq \{y_{j1}, y_{j2}, y_{j3}\}$ such that $y_{j1} < y_{j2} < y_{j3}$.

Remark: Given any instance $(X, \mathcal{Y})$ to the X3C-3 problem, each $x \in X$ appears in at most 3 set of $\mathcal{Y}$. It must be true that in Π, there are at most 3 sets of $\mathcal{Y}$ between any adjacent elements $x, x' \in X$.

Example 3. *Let* $X = \{a, b, c, d, e, f\}$ $(a < b < c < d < e < f)$ *and* $\mathcal{Y} = \{\{a, b, c\}, \{b, d, e\}, \{c, e, f\}, \{d, e, f\}\}$. *The order of* $\mathcal{Y}$ *is* $Y_1 = \{a, b, c\}, Y_2 = \{b, d, e\}, Y_3 = \{c, e, f\}, Y_4 = \{d, e, f\}$. *The order of* $X \cup \mathcal{Y}$ *is as follows:*

$$\Pi = \{a, b, c, Y_1, d, e, Y_2, f, Y_3, Y_4\}$$

The solution to the X3C-3 problem $(X, \mathcal{Y})$ *is* $\{Y_1, Y_4\}$.

A.2. Input Tree Construction

Given $(X, \mathcal{Y})$, for any $j = 1, \ldots, |\mathcal{Y}|$, based on $Y_j = \{y_{j1}, y_{j2}, y_{j3}\}$, we define $\tilde{Y_j} \triangleq \{e_j, y_{j1}, y_{j2}, y_{j3}\}$. For any $x_k \in X$ $(k = 1, \ldots, |X|)$, we define a related variable a_k for future use.

Structure: The input category tree is constructed for $(X, \mathcal{Y})$. For any $j = 1, \ldots, |\mathcal{Y}|$, the component for $\tilde{Y_j}$ is depicted by $T_{v_j^{\mathcal{Y}}}$ in Figure 6a. Please refer to Figure 6b for the component for X_k (i.e., $T_{v_k^X}$), for any $k = 1, \ldots, |X|$.

Given $(X, \mathcal{Y})$, we now recursively define the structure of the hierarchy $\mathcal{T}$ based on the ascending order Π. We start from $i = 1$, and each time we consider the following two cases:

- If $\pi_i = x_k$, for some $k \in [|X|]$, we create a node $u_i = u_k^X$, which takes u_{i-1} ($u_0 \triangleq \emptyset$) as its left child and $\mathcal{T}_{v_k^X}$ (Figure 6b) as its right subtree;
- If $\pi_i = Y_j$, for some $j \in [|\mathcal{Y}|]$, we create a node $u_i = u_j^{\mathcal{Y}}$, which takes u_{i-1} as its left child and $\mathcal{T}_{v_j^{\mathcal{Y}}}$ (Figure 6a) as its right subtree.

Please refer to Figure 7 for the input tree corresponding to the instance in Example 3.

Weight: Define the weight assignment $w : \mathcal{T} \to \mathbb{R}^+ \cup \{0\}$ [7]. All internal nodes of the tree weigh 0, i.e., $w(u) = 0, \forall u \in \mathsf{Int}(\mathcal{T})$. For any $x \in X$, $w(x) > 0$. Moreover, for any category u that corresponds to x, it holds that $w(u) = w(x)$. For instance, in Example 3, if we set $w(a) = 1$, then $w(y_{11}) = 1$ since y_{11} refers to category a.

For any $k = 1, \ldots, |X|$, define W_{x_k} as follows:

$$W_{x_k} \triangleq \sum_{Y_j \prec x_k} w(\tilde{Y_j}) + \sum_{k'=1}^{k-1} w(x_{k'})$$

We now assign weights to the leaves based on the order Π:

To begin, $w(x_1) = 1$. Starting from $i = 2$, we iteratively assign weights to leaves that are associated with π_i.

- If $\pi_i = x_k$, for some $k = 1, \ldots, |X|$ we set $w(a_k)$ and $w(x_k)$ as the following:

  $$\begin{aligned} w(x_k) &= 1 + 8 \max\{|n|^3 w(x_{k-1}), W_{x_{k-1}}\} \\ w(a_k) &= 1 + \max_{x_k < Y_j < x_{k+1}} \{Y_j\} \end{aligned}$$

  where $n \triangleq |\mathcal{T}|$.

---

[7] Here we only assign non-negative weights to the nodes, which could be normalized to obtain the probabilities.

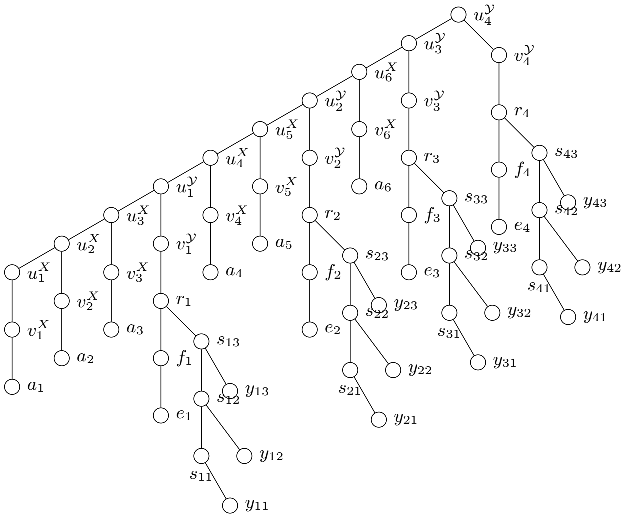


Figure 7. Tree structure for the instance in Example 3.

- If $\pi_i = Y_j$, for some $j = 1, \ldots, |\mathcal{Y}|$,

$$w(e_j) = W_{y_{j3}} + \alpha(Y_j)w(Y_j) + w(Y_j)/2$$

where $\alpha(Y_j) \triangleq |\{Y_{j'} | y_{j3} \prec Y_{j'} \preceq Y_j\}|$.

It can be calculated by induction that $w(\tilde{Y_j}) = O(n^{3j})$ and the total weight of the tree is $w(\mathcal{T}) = O(n^{3(|X|+|\mathcal{Y}|)})$. Thus the encoding of the weights needs $O((|X| + |\mathcal{Y}|)n \log n)$ bits. Therefore, the size of the instance $(\mathcal{T}, \Pr)$ of the MDT problem is polynomial in the size of the instance $(X, \mathcal{Y})$ of the X3C-3 problem.

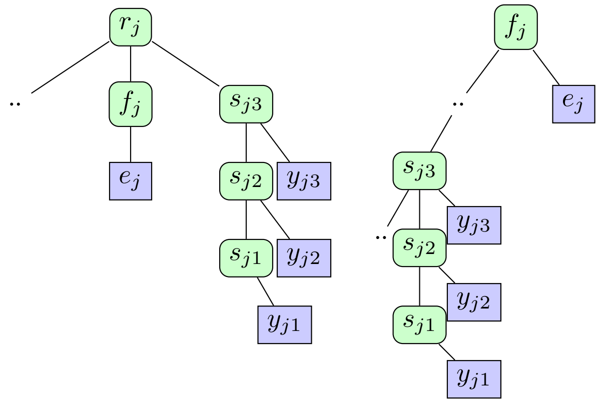
It is worth noting that the weight assignment is quite similar to the assignment in (Cicalese et al., 2011). In addition, we are highly inspired by the construction of input trees and decision trees, as well as the proofs ideas in (Cicalese et al., 2011).

A.3. Potential Optimal Decision Trees

Given $(X, \mathcal{Y})$ and the corresponding $(\mathcal{T}, w)$, we first define two potential structures for $Y \in \mathcal{Y}$ in the optimal decision tree.

For any $j = 1, \ldots, |\mathcal{Y}|$, Let $\mathcal{D}_j^A$ be the decision tree with root r_j, with one subtree a question at f_i, another subtree sequentially asking questions at s_{j3}, s_{j2}, s_{j1}. Please refer to Figure 8a [8] for this.

Let $\mathcal{D}_j^B$ be the decision tree with root f_j. The decision tree for $\mathcal{T}_{s_{j3}}$, denoted by $\mathcal{D}_j^{\text{aux}}$, would be sequentially asking questions at s_{j3}, s_{j2}, s_{j1}. $\mathcal{D}_j^{\text{aux}}$ is placed down on the leftmost path of $\mathcal{D}_j^B$. Figure 8b illustrates this.



(a)$\mathcal{D}_j^A$ (b)$\mathcal{D}_j^B$

Figure 8. Two possible decision tree structures for $\tilde{Y}_j$.

Definition 6 (Extension)**.** *Given decision tree $\mathcal{D}_1$, in which the answer to the "None of the above" option of* $\mathsf{root}(\mathcal{D}_1)$ *is $\emptyset$, the emphextension of $\mathcal{D}_1$ with another decision tree $\mathcal{D}_2$ is constructed by assigning* $\mathsf{root}(\mathcal{D}_2)$ *as the leftmost child of* $\mathsf{root}(\mathcal{D}_1)$.

By appending $\mathcal{D}_2$ to $\mathcal{D}_1$, $\mathcal{D}_2$ simply corresponds to the decision tree for the "None of the above" answer of $\mathsf{root}(\mathcal{D}_1)$.

Definition 7 (Realization (Cicalese et al., 2011))**.** *Given $S \subseteq \mathcal{Y}$, for any $i \in [|X| + |\mathcal{Y}|]$, a* realization *of* $\pi_i, \ldots, \pi_{|X|+|\mathcal{Y}|}$ w.r.t. *S, denoted by $\mathcal{D}_i^S$, is defined recursively as an extension of $\mathcal{D}_{i+1}^S$,*

- *If $\pi_i = Y_j$, for some $j = 1, \ldots, |\mathcal{Y}|$, then we simply append $\mathcal{D}_j^A$ to $\mathcal{D}_{i+1}^S$;*
- *If $\pi_i = x_k$ for some $k = 1, \ldots, |X|$, we first extend $\mathcal{D}_{i+1}^S$ with a subtree with root f_j. We traverse $\{k' : x_k \prec Y_{k'} \prec x_{k+1}, \}$ (may be empty) in the descending order and iteratively append $\mathcal{D}_{k'}^{aux}$* iff *$\mathcal{D}_{k'}^{aux}$ does not appear in $\mathcal{D}_{i+1}^S$.*

Please refer to Figure 9a and b for the realization of Π *w.r.t.* $\emptyset$ and $\{Y_1, Y_4\}$, respectively, for the instance in Example 3.

The above definition introduces a specific type of decision trees, which is crucial to the optimal decision tree. Given $(X, \mathcal{Y})$ and $(\mathcal{T}, w)$, correspondingly, we will prove that the optimal decision tree for $(\mathcal{T}, w)$ is a realization of Π *w.r.t.* the exact cover for $(X, \mathcal{Y})$.

Notation: Given $\mathcal{S} \subseteq \mathcal{Y}$, denote by $\mathcal{D}^{\mathcal{S}}$ the realization of Π *w.r.t.* $\mathcal{S}$. For any $Y_j \in \mathcal{S}$, denote by $\Delta(Y_j, \mathcal{S})$ the difference between the depth of s_{j3} in $\mathcal{D}^{\mathcal{S}}$ and that of s_{j3} in $\mathcal{D}^{\mathcal{S}-\{Y_j\}}$, i.e.,

$$\Delta(Y_j, \mathcal{S}) \triangleq \mathsf{dep}_{\mathcal{D}^{\mathcal{S}}}(s_{j3}) - \mathsf{dep}_{\mathcal{D}^{\mathcal{S}-\{Y_j\}}}(s_{j3})$$

---

[8] In the decision trees, we omit the leaves with zero weight, since they do not contribute to the costs.

In the remainder of the section, we propose several properties of the realization of Π.

Proposition 1. *Given $\mathcal{S} \subseteq \mathcal{Y}$, for any $j \in [|\mathcal{Y}|]$ such that $Y_j \in \mathcal{S}$,*

$$\Delta(Y_j, \mathcal{S}) = \alpha(Y_j) + \beta(Y_j, \mathcal{S}) \tag{5}$$

where $\alpha(Y_j) \triangleq |\{Y_{j'} | y_{j3} \prec Y_{j'} \preceq Y_j\}|$ and $\beta(Y_j, \mathcal{S}) \triangleq |\{Y_{j'} \in \mathcal{S} | Y_j \prec Y_{j'}, y_{j'3} = y_{j3}\}|$.

Proof. By definition of $\mathcal{D}^S$, for some $k \in [|X|]$ such that $y_{j3} = x_k$, it holds that the depth of f_j in $\mathcal{D}^{\mathcal{S}}$ remains unchanged w.r.t. that of $\mathcal{D}^{\mathcal{S}-\{Y_j\}}$, since the change from $\mathcal{D}^A_j$ to $\mathcal{D}^B_j$ only affects nodes that place below it (see Figure 8 and 9). Moreover, for any $j' = 1, \dots, |\mathcal{Y}|$ such that $y_{j'3} \geq x_k$, the depths of $r_{j'}(f_{j'})$ in $\mathcal{D}^{\mathcal{S}}$ and $\mathcal{D}^{\mathcal{S}-\{Y_j\}}$ are the same.

Therefore, to convert $\mathcal{D}^{\mathcal{S}-\{Y_j\}}$ into $\mathcal{D}^{\mathcal{S}}$, it takes $\alpha(Y_j)$ steps to place $\mathsf{root}(\mathcal{D}^{\text{aux}}_j)$ one stair below v^X_k (see Figure 9). We need $|\{Y_j | Y_j \prec Y_{j'}, y_{j'3} = y_{j3}\}|$ more steps to finally place $\mathsf{root}(\mathcal{D}^{\text{aux}}_j)$ in the appropriate position. ☐

Proposition 2. *Given $\mathcal{S} \subseteq \mathcal{Y}$, it holds that*

$$\begin{aligned} & \mathsf{cost}(\mathcal{D}^{\emptyset}) - \mathsf{cost}(\mathcal{D}^{\mathcal{S}}) \\ = & \sum_{j : Y_j \in \mathcal{S}} \left(w(e_j) - W_{y_{j3}} - \Delta(Y_j, \mathcal{S}) w(Y_j) \right) \end{aligned} \tag{6}$$

Proof. W.l.o.g., we assume that $\mathcal{S} = \{Y_{j_1}, Y_{j_2}, \dots, Y_{j_{|S|}}\}$ ($Y_{j_1} \prec Y_{j_2} \prec \dots \prec Y_{j_{|S|}}$). We fix a sequence $\mathcal{S}_{|\mathcal{S}|} = \emptyset, \mathcal{S}_{|\mathcal{S}|-1} = \{Y_{j_{|S|}}\}, \dots, \mathcal{S}_0 = \mathcal{S}$. It suffices to show that for any $p = 1, \dots, |S|$, we have

$$\mathsf{cost}(\mathcal{D}^{\mathcal{S}_{p-1}}) - \mathsf{cost}(\mathcal{D}^{\mathcal{S}_p}) = w(e_{j_p}) - W_{y_{j_p3}} - \Delta(Y_{j_p}, \mathcal{S}) w(Y_{j_p})$$

Observe that in $\mathcal{D}^{\mathcal{S}_p}$, the decision tree part for $\tilde{Y_{j_p}}$ is $\mathcal{D}^A_{j_p}$ while that in $\mathcal{D}^{\mathcal{S}_{p-1}}$ is $\mathcal{D}^B_{j_p}$. To turn $\mathcal{D}^{\mathcal{S}_p}$ into $\mathcal{D}^{\mathcal{S}_{p-1}}$, we first need to move f_{j_p} and e_{j_p} one level up, which constitutes the positive term in the above equation.

For the negative terms, $\mathcal{D}^{\text{aux}}_{j_p}$ is moved down by $\Delta(Y_{j_p}, \mathcal{S})$ steps. Moreover, all the decision subtrees that correspond to $\pi_q (1 \leq q < p)$ have to be moved down one level, which constitutes the $W_{y_{j_p3}}$ part. ☐

In the remainder of the section, to prove Theorem 1, we propose and prove two key lemmas that could bind the MDT problem and X3C-3 problem together.

A.4. Key Lemma 1

Lemma 1. *Given any instance $(\mathcal{T}, w)$, the optimal decision tree is a realization of* Π w.r.t. *some $\mathcal{S} \subseteq \mathcal{Y}$.*

To prove the lemma, we first state the following propositions:

Proposition 3. *for any $1 \leq j' < j \leq |\mathcal{Y}|$, it holds that*

$$w(e_j) > w(e_{j'}) + w(y_{j'1}) + w(y_{j'2}) + w(y_{j'3}) \tag{7}$$

Proof. We prove this by breaking j' into two cases:

(a) $y_{j3} = y_{j'3}$. By definition of $\alpha(Y_j)$, it holds that $\alpha(Y_j) \geq \alpha(Y_{j'}) + 1$. Thus we have the following:

$$\begin{aligned} & w(e_j) - (w(e_{j'}) + w(y_{j'1}) + w(y_{j'2}) + w(y_{j'3})) \\ = & \left(\alpha(Y_j) + \frac{1}{2}\right) w(Y_j) - \left(\alpha(Y_{j'}) + \frac{3}{2}\right) w(Y_{j'}) \\ \geq & \frac{3}{2} w(Y_j) - \frac{3}{2} w(Y_{j'}) \\ > & 0 \end{aligned}$$

where the inequality holds since $\alpha(Y_j) \geq \alpha(Y_{j'}) + 1$ and $w(Y_j) > w(Y_{j'})$.

(b) $y_{j3} > y_{j'3}$. (7) holds since

$$w(e_j) > W_{y_{j3}} > w(e_{j'}) + w(y_{j'1}) + w(y_{j'2}) + w(y_{j'3})$$

☐

Proposition 4. *For any $j \in |\mathcal{Y}|$ and $k = 1, \dots, |X|$ such that $y_{j3} = x_k$, it holds that*

$$w(Y_j) < w(a_k) < w(e_j) \tag{8}$$

Proof. The left part of the inequality holds by definition of $w(a_k)$. For the right part, it holds that

$$w(e_j) > W_{x_k} + \frac{3}{2} w(x_k) > w(a_k)$$

☐

Now we prove Lemma 1 as follows:

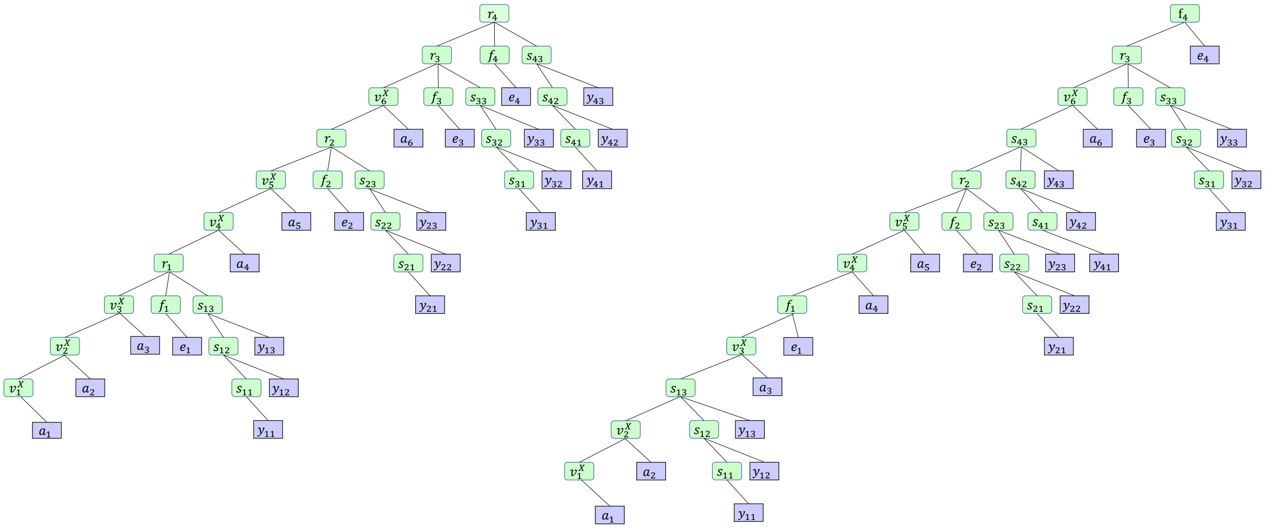
Proof of Lemma 1. Let $\mathcal{D}^*$ be the optimal decision tree for the instance $(\mathcal{T}, \Pr)$. Denote by p the node with the lowest level on the left most path such that $\mathcal{D}^* - \mathcal{D}^*_p$ is a realization of $\pi_{i+1}, \dots, \pi_{|X|+|\mathcal{Y}|}$ for some $i \geq 0$. If $i = 0$, the lemma directly follows.

Assume by contradiction that $i > 0$. We construct $\mathcal{D}'$ from $\mathcal{D}^*$ by modifying $\mathcal{D}^*_p$ such that $\mathcal{D}'$ is a realization of $\pi_i, \dots, \pi_{|X|+|\mathcal{Y}|}$, and we will prove that $\mathsf{cost}(\mathcal{D}') < \mathsf{cost}(\mathcal{D}^*)$, which leads to a contradiction.

After this, we proceed by induction in the decreasing order i, until finally we end up with a decision tree which realizes Π, and thus conclude its optimality.

We consider the following two cases:

Case 1: $\pi_i = Y_j$, for some $j = 1, \dots, |\mathcal{Y}|$.



(a) *Realization w.r.t.* $\emptyset$ (b) *Realization w.r.t.* $\{Y_1, Y_4\}$

Figure 9. Realizations of Π for the instances in Example 3.

First we argue that $p \notin \{u_j^{\mathcal{Y}}, v_j^{\mathcal{Y}}\}$ in $\mathcal{D}_p^*$: Assume by contradiction that $p = u_j^{\mathcal{Y}}$. In this case, we construct $\mathcal{D}_p'$ as follows: we replace $u_j^{\mathcal{Y}}$ with r_j; we assign f_j as its second child; and for the third subtree, we remain the order of asking questions at Y_j in $\mathcal{D}_p^*$. It can be easily verified that $\mathcal{D}_p'$ (and the corresponding $\mathcal{D}'$) is a valid decision tree. Moreover, it holds that

$$\begin{aligned} \mathsf{cost}(\mathcal{D}') &\leq \mathsf{cost}(\mathcal{D}^*) \\ &\quad - \min\{w(Y_j), w(y_j1), w(y_j2), w(y_j3)\} \\ &< \mathsf{cost}(\mathcal{D}^*) \end{aligned}$$

which contradicts the optimality of $\mathcal{D}^*$.

By similar analysis, we could construct $\mathcal{D}'$ by replacing $v_j^{\mathcal{Y}}$ with σ_{r_j} and show that $\mathsf{cost}(\mathcal{D}') < \mathsf{cost}(\mathcal{D}^*)$, which means that $p \neq v_j^{\mathcal{Y}}$.

Next we prove that $p \in \{r_j, f_j\}$. Based on the positions of r_j and f_j, we could consider the following two cases:

Subcase 1.1: f_j is an ancestor of r_j in $\mathcal{D}_p^*$. Define q to be the father of f_j. In this case, f_j must be the leftmost child of q. We swap f_j with q. By *swapping*, we mean that the two nodes exchange positions and their subtrees except the leftmost ones are carried along with them.

If $q \in \{u_k^X, v_k^X\}$ for some $k \geq 1$, it holds that $x_k \geq y_{j3}$ by definition of p. Therefore, we have that

$$\mathsf{cost}(\mathcal{D}') \leq \mathsf{cost}(\mathcal{D}^*) - w(e_j) + w(a_k) < \mathsf{cost}(\mathcal{D}^*)$$

where the second inequality is guaranteed by Proposition 4.

If q corresponds to some node in $Y_{j'}$, for some $j' = 1, \ldots, j-1$, then we have

$$\begin{aligned} &\mathsf{cost}(\mathcal{D}') \\ \leq\ &\mathsf{cost}(\mathcal{D}^*) - w(e_j) + w(e_{j'}) + w(y_{j'1}) + w(y_{j'2}) + w(y_{j'3}) \\ <\ &\mathsf{cost}(\mathcal{D}^*) \end{aligned}$$

where the second inequality is stated by Proposition 3.

Subcase 1.2: r_j is an ancestor of f_j. In this case f_j must be a child of r_j. Now denote by q the father of r_j.

If $q \in \{s_{j1}, s_{j2}, s_{j3}\}$, we first replace q with f_j; then we assign q as a child of r_j (all subtrees of q or f_j except the leftmost one are carried along with it), and append the original sub-decision trees of r_j corresponding to Y_j to q. In this case, we have

$$\begin{aligned} &\mathsf{cost}(\mathcal{D}') \\ \leq\ &\mathsf{cost}(\mathcal{D}^*) - w(e_j) + w(y_{j1}) + w(y_{j2}) + w(y_{j3}) \\ <\ &\mathsf{cost}(\mathcal{D}^*) \end{aligned}$$

If $q \in \{u_k^X, v_k^X\}$ for some $k \geq 1$, or q corresponds to nodes in $Y_{j'}$, for some $j' = 1, \ldots, j-1$, we can swap r_j and q. By similar analysis, we show that $\mathsf{cost}(\mathcal{D}') < \mathsf{cost}(\mathcal{D}^*)$, which leads to a contradiction.

Therefore, $p \in \{r_j, f_j\}$.

If $p = f_j$, $\mathcal{D}^*$ would be a decision tree which realizes $\pi_i, \ldots, \pi_{|X|+|\mathcal{Y}|}$, leading to a contradiction.

If $p = r_j$, we must have f_j as one of its children. In addition, the decision tree for Y_j serves as the other subtree (than the leftmost one) of p. Since $w(y_{j3}) \gg w(y_{j2}) \gg w(y_{j1})$, it is not hard to verify that the optimal decision tree for Y_j is to ask questions for s_{j3}, s_{j2}, s_{j1} in order. This makes $\mathcal{D}_p$ a realization of π_i and thus contradicts the property of i.

Case 2: $\pi_i = x_k$ for some $j = 1, \ldots, |X|$.

By similar analysis in case 1, we can argue that $p \neq u_k^X$.

Let U denote the set that includes nodes of $\mathcal{T}$, which are associated with x_k but not queried in $\mathcal{D}^* - \mathcal{D}_p^*$. We also add a_k into U. Let W denote the set of all nodes that queried in $\mathcal{D}_p^*$. $W - U$ contains nodes that are in $\cup_{Y \prec x_k} Y$ and nodes that are associated with $x_{j'} (j' < j)$ but not queried in $\mathcal{D}^* - \mathcal{D}_p^*$. Therefore, $w(W - U)$ can be upper bound by the following:

$$w(W-U) < W_{x_k} + |\mathcal{T}| w(x_{j-1}) < \frac{w(x_k)}{4} \tag{9}$$

We first argue that all nodes in U are placed on the top $|U|$ nodes of $\mathsf{leftpath}(\mathcal{D}_p^*)$ (recall that $\mathsf{leftpath}(\mathcal{D}_p^*)$ is denoted by the leftmost path of $\mathcal{D}_p^*$). Assume by contradiction that the top $|U|$ levels include at least one node that is not associated with x_k or a_k. In this case, we construct $\mathcal{D}_p'$ from $\mathcal{D}_p$ as follows: we place questions at nodes in U (along with their subtrees other than the leftmost one) on the top $|U|$ levels of $\mathsf{leftpath}(\mathcal{D}_p^*)$. On $\mathsf{leftpath}(\mathcal{D}_p^*)$, we append all the other nodes to the first $|U|$ nodes in the original order. We have the following:

$$\begin{aligned} \mathsf{cost}(\mathcal{D}') &\leq \mathsf{cost}(\mathcal{D}^*) - w(x_k) + |U| w(W-U) \\ &\leq \mathsf{cost}(\mathcal{D}^*) - w(x_k) + 4w(W-U) \\ &< \mathsf{cost}(\mathcal{D}^*) \end{aligned}$$

where first inequality holds since $w(a_k) > w(x_k)$ and all nodes in $W - U$ are moved down at most $|U|$ steps, the second one holds since $|U| \leq 4$ by Definition 4 and the last inequality holds because of (9). Therefore, it leads to a contradiction.

Define $U \triangleq \{a_k, y_{j_1 3}, \ldots, y_{j_{|U|-1} 3}\}$, where $y_{j_m 3}$ is associated with x_k for any $m = 1, \ldots, |U| - 1$ and $j_1 > \ldots > j_{|U|-1}$. It follows that

$$w(a_k) > w(Y_{j_1}) > \ldots > w(Y_{j_{|U|-1}})$$

Therefore, the top $|U|$ nodes of $\mathsf{leftpath}(\mathcal{D}_p^*)$ should be $v_k^X, s_{j_1 3}, \ldots, s_{j_{|U|-1} 3}$ from top down, since otherwise we could just swap two nodes that violate the order and achieve a decision tree with less cost.

Finally, for any $m = 1, \ldots, |U| - 1$, since $w(Y_{j_m 2}) \gg w(Y y_{j_m 1})$, we would first ask the question for $s_{j_m 2}$ before one for $s_{j_m 1}$. Therefore, $\mathcal{D}_p^*$ should be a realization of π_i which contradicts the fact that p is the deepest node such that $\mathcal{D}^* - \mathcal{D}_p^*$ is a realization of $\pi_{i+1}, \ldots, \pi_{|X|+|\mathcal{Y}|}$. □

A.5. Key Lemma 2

For any $\mathcal{S} \subseteq \mathcal{Y}$, by plugging (5) and $w(e_i)$ into (6), we have

$$\begin{aligned} & \mathsf{cost}(\mathcal{D}^{\emptyset}) - \mathsf{cost}(\mathcal{D}^{\mathcal{S}}) \\ = & \sum_{j: Y_j \in \mathcal{S}} \big(w(e_j) - W_{y_{j3}} - \Delta(Y_j, \mathcal{S}) w(Y_j) \big) \\ = & \sum_{k=1}^{|X|} \sum_{Y_j \in \mathcal{S}, x_k \in Y_j} \left(\frac{w(x_k)}{2} - \beta(Y_j, \mathcal{S}) w(x_k) \right) \end{aligned} \tag{10}$$

where the last equality holds by changing the order of summation.

Lemma 2. *Let $\mathcal{D}^*$ be an optimal decision tree for $(\mathcal{T}, w)$ and let $\mathcal{S} \subseteq \mathcal{Y}$ be the set such that $\mathcal{D}^*$ is a realization of* Π w.r.t. *$\mathcal{S}^*$. It holds that* $\mathsf{cost}(\mathcal{D}^*) \leq \mathsf{cost}(\mathcal{D}^{\emptyset}) - \frac{1}{2} \sum_{i=1}^{|X|} w(x_k)$ iff. *$\mathcal{S}^*$ is a solution for the corresponding X3C-3 instance $(X, \mathcal{Y})$.*

Proof. $\Longleftarrow$ If $\mathcal{S}^*$ is a solution for the X3C-3 instance $(X, \mathcal{Y})$, for any $Y \in \mathcal{S}^*$, it holds that $\beta(Y, \mathcal{S}^*) = 0$.

$\Longrightarrow$ We prove the sufficient part by induction on $k' = |X|, \ldots, 1$.

We assume that given $k' \in [|X|]$, for any $k > k'$, there exists exactly one $Y \in \mathcal{S}^*$ such that $x_k \in Y$. But for k', there exists either no Y or multiple sets that include $x_{k'}$.

From (10), it holds that

$$\begin{aligned} & \mathsf{cost}(\mathcal{D}^{\emptyset}) - \mathsf{cost}(\mathcal{D}^*) \\ = & \sum_{k=1}^{|X|} \sum_{Y_j \in \mathcal{S}^*, x_k \in Y_j} \left(\frac{w(x_k)}{2} - \beta(Y_j, \mathcal{S}^*) w(x_k) \right) \\ \leq & \sum_{k > k'} \frac{w(x_k)}{2} + (\frac{1}{2} - \beta(Y_j, \mathcal{S}^*)) w(x_{k'}) \\ & + 3(k'-1)(n + \frac{1}{2}) \frac{w(x_k')}{8n^3} \\ < & \sum_{k > k'} \frac{w(x_k)}{2} + (\frac{1}{2} - \beta(Y_j, \mathcal{S}^*)) w(x_{k'}) + \frac{w(x_{k'})}{2} \end{aligned}$$

where the first inequality holds since (a) there are at most 3 sets in $\mathcal{S}^*$ that include x_k; (b) $\beta(Y_j, \mathcal{S}^*)$ can be lower bounded by $-n$; (c) for any $k < k'$, $w(x_k) < \frac{w(x_k')}{8n^3}$.

By the assumption that either no set in $\mathcal{S}^*$ includes $x_{k'}$ or there are multiple sets in $\mathcal{S}^*$ including $x_{k'}$. In both cases, we have $\beta(Y_j, \mathcal{S}^*) >= \frac{1}{2}$. From the above inequality, we have

$$\mathsf{cost}(\mathcal{D}^{\emptyset}) - \mathsf{cost}(\mathcal{D}^*) < \sum_{k > k'} \frac{w(x_k)}{2} + \frac{w(x_{k'})}{2} < \sum_{x \in X} \frac{w(x)}{2}$$

which contradicts that $\mathsf{cost}(\mathcal{D}^*) \leq \mathsf{cost}(\mathcal{D}^{\emptyset}) - \frac{1}{2} \sum_{k=1}^{|X|} w(x_k)$. □

Finally, Theorem 1 directly follows from Lemma 1 and 2.

B. Greedy

B.1. Proof of Theorem 2

Given any instance $(\mathcal{T}, \mathrm{Pr})$, denote by $\mathcal{D}$ and $\mathcal{D}^*$ the greedy decision tree and the optimal decision tree, respectively. Define $\mathcal{D}'$ as the decision tree whose root is $\mathsf{root}(\mathcal{D})$, and all the resulting groups are constructed by the optimal strategy.

We first prove the following lemma, the proof of which is similar to the ideas in (Cicalese et al., 2010):

Lemma 3. $\mathsf{cost}(\mathcal{D}') \leq \mathsf{cost}(\mathcal{D}^*) + \frac{1}{2}$.

Proof. If $\mathsf{root}(\mathcal{D}') = \mathsf{root}(\mathcal{D}^*)$, the lemma directly follows.

Recall that for any $u \in \mathcal{T}$, $\Phi_{\mathcal{T}}(u)$ is denoted by the resulting subtrees partitioned by u. We assume that $\mathsf{root}(\mathcal{D}^*) \in \mathcal{T}_0 \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}'))$.

Let $u_1, u_2, \ldots$ be the path in $\mathcal{D}^*$ from $\mathsf{root}(\mathcal{D}^*)$ to the category that corresponds to $\mathsf{root}(\mathcal{D})$. Let u_m $(m > 1)$ denote the first node on the path such that $u_m \in \mathcal{T} - \mathcal{T}_0$. Recall that for any decision tree $\mathcal{D}_0$ and $u \in \mathcal{T}$, denote by $\mathsf{dep}_{\mathcal{D}_0}(\mathsf{leaf}(u))$ the depth of $\mathsf{leaf}(u)$ in $\mathcal{D}_0$.

For any $\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}'))$, denote by $\mathcal{D}^*_{\mathcal{T}'}$ the optimal decision tree for $\mathcal{T}'$. It is trivial to see the followings:

$$\begin{aligned} & \mathsf{dep}_{\mathcal{D}^*}(u) - \mathsf{dep}_{\mathcal{D}^*_{\mathcal{T}'}}(u) \\ \geq & \begin{cases} m-1 & \text{if } \mathcal{T}' \neq \mathcal{T}_0 \& u \in \mathsf{Lvs}(\mathcal{D}') \cap \mathcal{T}' \\ 1 & \text{if } u \in \mathsf{Lvs}(\mathcal{D}') \cap \left(\mathcal{T}_0 - \cup_{j=1}^{m-1} \mathcal{T}_{u_j}\right) \end{cases} \end{aligned} \tag{11}$$

Therefore, it holds that

$$\begin{aligned} & \mathsf{cost}(\mathcal{D}^*) - \sum_{\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}'))} \mathsf{cost}(\mathcal{D}^*_{\mathcal{T}'}) \\ = & \sum_{\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}'))} \sum_{v \in \mathcal{T}' \cap \mathsf{Lvs}(\mathcal{D}')} \\ & (\mathsf{dep}_{\mathcal{D}^*}(v) - \mathsf{dep}_{\mathcal{D}^*}(v)) \Pr(v) \\ \geq & (m-1) \Pr(\mathcal{T} - \mathcal{T}_0) + \Pr(\mathcal{T}_0 - \cup_{j=1}^{m-1} \mathcal{T}_{u_j}) \\ \geq & (m-1) \Pr(\mathcal{T} - \mathcal{T}_0) + \Pr(\mathcal{T}_0) + \sum_{j=1}^{m-1} \Pr(\mathcal{T}_{u_j}) \end{aligned}$$

where the first inequality holds because of (11).

Moreover, we have the following proposition:

Proposition 5. $\Pr(\mathcal{T}_{u_j}) \leq \max\{\frac{1}{2}, \Pr(\mathcal{T} - \mathcal{T}_0)\}$*, for any* $j = 1, \ldots, m$*.*

Then we obtain the following:

$$\begin{aligned} & \mathsf{cost}(\mathcal{D}^*) - \sum_{\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}'))} \mathsf{cost}(\mathcal{D}^*_{\mathcal{T}'}) \\ \geq & (m-1) \Pr(\mathcal{T} - \mathcal{T}_0) + \Pr(\mathcal{T}_0) + \sum_{j=1}^{m-1} \Pr(\mathcal{T}_{u_j}) \\ \geq & \Pr(\mathcal{T} - \mathcal{T}_0) + \Pr(\mathcal{T}_0) - \Pr(\mathcal{T}_{u_1}) \\ \geq & \frac{1}{2} \end{aligned}$$

where the last two inequalities hold due to Proposition 5 and $m > 1$.

Therefore, it holds that

$$\mathsf{cost}(\mathcal{D}') = \sum_{\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}'))} \mathsf{cost}(\mathcal{D}^*_{\mathcal{T}'}) + 1 \leq \mathsf{cost}(\mathcal{D}^*) + \frac{1}{2}$$

□

Proof of Proposition 5. Assume by contradiction that $\Pr(\mathcal{T}_{u_j}) > \max\{\frac{1}{2},$
$\Pr(\mathcal{T} - \mathcal{T}_0)\}$. Since $\mathcal{T}_{u_j} \subseteq \mathcal{T}_0$, it must hold that $\Pr(\mathcal{T}_0) > 1/2$.

However, we can find a node u of T from bottom up, such that $\Pr(\mathcal{T}_u) > \frac{1}{2}$ and for any internal child v of u, it holds that $\Pr(\mathcal{T}_v) < \frac{1}{2}$. If we choose u to partition $\mathcal{T}$, it must hold that

$$\Pr(\mathcal{T}) \leq \frac{1}{2} \qquad \mathcal{T} \in \Phi_{\mathcal{T}}(u)$$

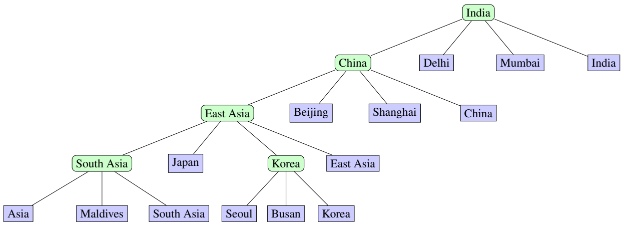
Therefore, by choosing u over $\mathsf{root}(\mathcal{D})$ as the root node of $\mathcal{T}$, the total probability of the heaviest resulting group can be reduce, which contradicts (2). Therefore, the proposition follows. □

We finally prove Theorem 2:

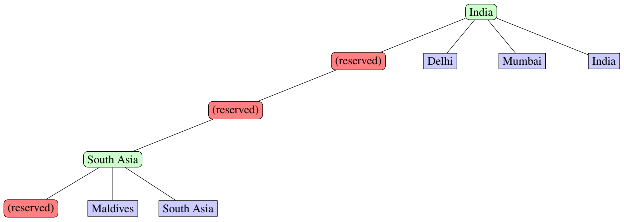
Proof of Theorem 2. For any $\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}))$, denote by $\mathcal{D}_{\mathcal{T}'}$ the decision tree for $\mathcal{T}'$ using Greedy. By induction, assume that $\mathsf{cost}(\mathcal{D}^*_{\mathcal{T}'}) \leq 2\mathsf{cost}(\mathcal{D}^*_{\mathcal{T}'})$. It follows that:

$$\begin{aligned} \mathsf{cost}(D) &= 1 + \sum_{\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}))} \mathsf{cost}(\mathcal{D}_{\mathcal{T}'}) \\ &\leq 1 + 2 \sum_{\mathcal{T}' \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{D}))} \mathsf{cost}(\mathcal{D}^*_{\mathcal{T}'}) \\ &\leq 2\mathsf{cost}(\mathcal{D}') - 1 \\ &\leq 2\mathsf{cost}(\mathcal{D}^*) \end{aligned}$$

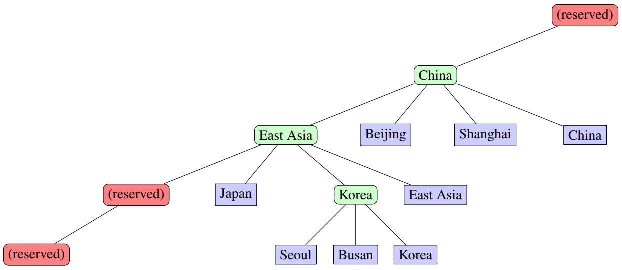
□



(a) Optimal decision tree for “Asia”.



(b) Optimal decision tree for “South Asia”.



(c) Optimal decision tree for “East Asia”.

Figure 10. Subproblems of dynamic programming.

C. FPTAS

In this section, we provide an FPTAS for the problem. We first define a closely related problem. Then we develop a non-trivial dynamic programming algorithm taking the revised problem as subproblems. At last, we transform the dynamic programming into an FPTAS. It is worth noting that in this section, we are highly inspired by the ideas in (cic, 2014).

Recall that for each internal node, its leftmost subtree represents a decision tree to the resulting subtree of the “None of the Above” option.

Intuitions: A natural idea to tackle the problem is to first construct the optimal decision tree for the subtrees of $\mathcal{T}$ and then somehow merge them into the optimal tree for the entire problem. For instance, to generate the optimal decision tree for “Asia” in Figure 1, it would be very tempting to first construct the optimal decision trees for “South Asia” and “East Asia”, and then merge them. However, the merging phase can be quite tricky in the sense that nodes from two different subtrees may interlace quite arbitrarily (see Figure 10(a)). Suppose we ask a question at “India” and receive the “None of the Above” answer, we shall proceed with asking about “China”. It occupies the second position of $\mathsf{leftpath}(\mathcal{D})$. In other words, any internal node in “South Asia” subtree other than “India” has to be placed in nodes of depth at least 2, which essentially destroys the structure of the “South Asia” decision tree.

From this, we could see that the second and third positions of $\mathsf{leftpath}(\mathcal{D})$ are reserved for questions in “East Asia” and by no means could we place nodes “South Asia” in those places. Likewise, we extract a revised decision tree (with reserved positions in the leaf path) for “East Asia” (see Figure 10(c)). The first and fourth positions of $\mathsf{leftpath}(\mathcal{D})$ are reserved for questions in “South Asia”, respectively. Finally, we merge them together, appending the leaf category “Asia”, and thus obtain the optimal decision tree for “Asia” (see Figure 10(a)).

By generalizing the above toy example, we define the subproblems of the dynamic programming to be $(\mathcal{T}, \Pr, \mathsf{leftpath})$, where each leftpath contains two kinds of nodes: *reserved* ones and *freely used* ones. By reserved nodes on leftpath, we mean that these nodes are reserved for questions asked in other subtrees, and questions at nodes in $\mathcal{T}$ can never be placed on reserved positions of leftpath.

For each subtree of $\mathcal{T}$, we consider all possible leftpaths and compute the optimal solution for the subproblems. We merge all decision trees for subtrees and finally output the decision tree with the lowest cost.

Given leftpath, denote by height the number of nodes on leftpath. leftpath can be represented by a bit vector, i.e., $\mathsf{leftpath} \in \{0,1\}^{\mathsf{height}}$, in which $\mathsf{leftpath}_i = 0$ if it is a reserved nodes or 0 otherwise, for any $i = 1, \ldots, \mathsf{height}$. Below we formally define the problem:

Problem 3. *(Restricted Min-cost Decision Tree (RMDT) Problem.) Given any instance* $(\mathcal{T}, \Pr, \mathsf{leftpath})$*, construct a decision tree* $\mathcal{D}$ w.r.t. $\mathcal{T}$ *and* leftpath*, with height at most* height *and incurring the minimum expected cost. the minimum expected cost.*

Given height, the following lemma which states the relationship between the decision tree to RMDT problem $(\mathcal{T}, \Pr, \ \mathsf{leftpath} = \mathbf{1}_{\mathsf{height}})$ and the optimal solution to MDT problem of height at most height.

Lemma 4. *Given the optimal decision tree* $\mathcal{D}^*$ *to RMDT problem* $(\mathcal{T}, \Pr, \mathbf{1}_{\mathsf{height}})$*, there exist a linear time algorithm that converts* $\mathcal{D}^*$ *to an optimal decision tree* $\mathcal{D}_1$ *for MDT problem* $(\mathcal{T}, \Pr)$ *of height at most* height*. Moreover, it holds that*

$$\mathsf{cost}(\mathcal{D}^*) = \mathsf{cost}(\mathcal{D}_1)$$

Proof. Denote by leftpath the leftmost path of D^*. We first argue that for any $i = 1, \ldots, \mathsf{height}$ such that $\mathsf{leftpath}_i$ is not assigned to any question of $\mathcal{T}$, the total probability of categories under $\mathcal{D}^*_{\mathsf{leftpath}_i}$ must be 0.

Assume that this is not true, there must be some $j > i$ such that some leaves of $\mathcal{D}^*_{\mathsf{leftpath}_j}$ are assigned to positive probabilities. Among these nodes, we select the one with the minimum index (denoted by j_0) and denote by v s leaf in $\mathcal{D}^*_{\mathsf{leftpath}_{j_0}}$ such that $\Pr(v) > 0$.

We can construct a tree $\mathcal{D}'$ simply by deleting nodes between i and $j_0 - 1$ from $\mathsf{leftpath}$. It is trivial to see that $\mathcal{D}'$ is a decision tree to RMDT problem with height at most height. It holds that

$$\mathsf{cost}(\mathcal{D}') \le \mathsf{cost}(\mathcal{D}^*) - \Pr(v) < \mathsf{cost}(\mathcal{D}^*)$$

which leads to a contradiction.

Finally to convert $\mathcal{D}^*$ into $\mathcal{D}_1$, we simply delete all nodes and their subtrees in $\mathcal{D}^*$, which are not assigned to questions. This makes the resulting $\mathcal{D}_1$ a feasible decision tree to MDT problem $(\mathcal{T}, \Pr)$ of height at most height.

Moreover, any decision tree $\mathcal{D}$ to MDT problem $(\mathcal{T}, \Pr)$ of height at most height is a already a decision tree to RMDT problem $(\mathcal{T}, \Pr, \mathbf{1}_{\mathsf{height}})$. Therefore, it must hold that

$$\mathsf{cost}(\mathcal{D}^*) = \mathsf{cost}(\mathcal{D}_1) \le \mathsf{cost}(\mathcal{D})$$

$\square$

C.1. Dynamic Programming

We first introduce one more definition which will be used in the algorithm:

Definition 8 (Compatibility)**.** *Given two positive integers* $h, l \le h + 1$ *and a bit vector* $\mathbf{g} \in \{0, 1\}^h$*, a set* $S = \{\mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^{|S|}\}$ *is said to be* compatible w.r.t. $(\mathbf{g}, l)$ iff. *all of the following properties are satisfied:*

(a) if $l \le h$*,* $g_l = 1$*;*

(b) $\forall i = 1, \ldots, h$*, it holds that*

$$s_i^{|S|} + s_i^{|S|} + \ldots + s_i^{|S|} = \begin{cases} g_i & \textit{if } i \le l - 1 \\ 0 & \textit{if } i = l \\ |S| & \textit{otherwise} \end{cases} \tag{12}$$

We next present the dynamic programming algorithm DP. Denote by $\mathrm{OPT}(\mathcal{T}, \mathsf{leftpath})$ DP solution to the subproblem $(\mathcal{T}, \Pr, \mathsf{leftpath})$.

Base Case: $\mathcal{T}$ is a star. In this case, if $\mathsf{leftpath}$ includes at least one node that is not reserved, then we can place question at $\mathsf{root}(\mathcal{T})$ to the top node $\mathsf{leftpath}_{i_0}$ with $\mathsf{leftpath}_{i_0} = 1$. Otherwise, there is no question to ask. In this case, there does not exist a feasible solution and its value is denoted by ∞.

$$\mathrm{OPT}(\mathcal{T}, \mathsf{leftpath}) = \begin{cases} \infty, & \text{if } \mathsf{leftpath} = \mathbf{0}_{\mathsf{height}} \\ \min\{i | \mathsf{leftpath}_i = 1\}, & \text{otherwise} \end{cases} \tag{13}$$

General Case: We have the following

$$\mathrm{OPT}(\mathcal{T}, \mathsf{leftpath}) = \min \left\{ \sum_{i : \mathcal{T}^i \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))} \mathrm{OPT}(\mathcal{T}^i, \mathbf{t}^i) \right.$$

$$\left. \middle| 1 \le l \le h + 1, \{\mathbf{t}^i\} \textit{ compatible w.r.t. } (\mathsf{leftpath}, l) \right\} \tag{14}$$

To put this into words, DP considers all possible positions for the question asked on $\mathsf{root}(\mathcal{T})$, and for each position, the algorithm considers all possible $\mathsf{leftpath}$s for the subtrees, and finally outputs a decision tree with the minimum expected cost:

1. If a question at $\mathsf{root}(\mathcal{T})$ is not asked, $l = h + 1$. We consider all possible compatible sets of t^i *w.r.t.* $(\mathsf{leftpath}, l)$ and obtain the decision tree with the minimum expected cost.
2. If a question at $\mathsf{root}(\mathcal{T})$ is asked, then it must be assigned to some $\mathsf{leftpath}_l$ such that $l \le h$ and $\mathsf{leftpath}_l = 1$. For any subproblem for $\mathcal{T}^i$, its l-th node on the leftmost path should be a *reserved* node (for $\mathsf{root}(\mathcal{T})$). We consider all possible compatible sets of t^i *w.r.t.* $(\mathsf{leftpath}, l)$ and select the minimum one.
3. We finally compare the two results above and output the decision tree with the smaller expected cost.

Time Complexity: There are at most $n2^{\mathsf{height}}$ subproblems in total. Each subproblem can be solved in $O(n2^{\mathsf{height}})$ time. So the algorithm takes $O(n^2 2^{\mathsf{height}})$ time.

Space Complexity: The space complexity is determined by the number of subproblems in total, and is thus $O(n2^{\mathsf{height}})$.

The correctness of the algorithm is guaranteed by the following theorem:

Theorem 6. *DP computes the optimal solution to RMDT problem* $(\mathcal{T}, \Pr, \mathsf{leftpath})$.

To prove the theorem, for the base case, the correctness can be easily verified; for the general case, it suffices to prove the following two lemmas. The proof of the following lemmas can be illustrated by Figure 10.

Lemma 5. *Given any instance* $(\mathcal{T}, \Pr, \mathsf{leftpath})$*, for any* $l = 1, \ldots, \mathsf{height} + 1$ *and any set* S *with*

$|S| = |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|$ *such that* S *is compatible* w.r.t. $(\mathsf{leftpath}, l)$*, it holds that*

$$OPT(\mathcal{T}, \mathsf{leftpath}) \le \sum_{i:\mathcal{T}^i \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))} OPT(\mathcal{T}^i, \mathbf{s}^i)$$

Proof. Denote by $\mathcal{D}^i$ the optimal decision tree for the subproblem $(\mathcal{T}^i, \Pr, \mathbf{s}^i)$. It suffices to show that we can merge $\mathcal{D}^i$s $(i = 1, \ldots, |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|)$ into a feasible solution $\mathcal{D}'$ for the subproblem $(\mathcal{T}, \Pr, \mathsf{leftpath})$.

During the procedure, we try to merge the decision trees by merging their leftpaths from top down. Starting from $j = 1$, each time we consider one of the following cases:

Case 1: $j < l$. If $\sum_{i=1}^{|\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|} s_j^i = 0$, then $\mathsf{leftpath}_j = 0$, i.e., $\mathsf{leftpath}_j$ is a reserved node. In this case, we simply do nothing but increase j.

If $\sum_{i=1}^{|\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|} s_j^i = 1$, we select $i(j)$ such that $s_j^{i(j)} = 1$, assign$s_j^{i(j)}$ to $\mathsf{leftpath}_j$ and assign all the subtrees except the leftmost one of $s_j^{i(j)}$ as subtrees of $\mathsf{leftpath}_j$.

Case 2: $j = l$ and $l \le \mathsf{height}$. It must hold that $s_j^i = 0, \forall i = 1, \ldots, |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|$ but $\mathsf{leftpath}_j = 1$. In this case, we simply ask a question at $\mathsf{root}(\mathcal{T})$ on $\mathsf{leftpath}_j$. Moreover, for any i, we assign $\mathcal{D}^i_{s^i_{j+1}}$ as a subtree of $\mathsf{leftpath}_j$ and we are done.

It can be shown that the constructed $\mathcal{D}'$ is a feasible decision tree to $(\mathcal{T}, \Pr, \mathsf{leftpath})$ of height at most height. In addition, we have the following:

$$\mathsf{cost}(\mathcal{D}') = \sum_{i:\mathcal{T}^i \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))} \mathrm{OPT}(\mathcal{T}^i, \mathbf{s}^i) \tag{15}$$

Therefore, the lemma follows from the above relation and (14). □

Lemma 6. *Given any subproblem* $(\mathcal{T}, \Pr, \mathsf{leftpath})$*, consider the* DP *solution* $OPT(\mathcal{T}, \mathsf{leftpath})$*. Denote by* $l = 1, \ldots, \mathsf{height} + 1$ *the index such that in a question at* $\mathsf{root}(\mathcal{T})$ *is asked on* $\mathsf{leftpath}_l$*. It follows that there exists a set* $\{\mathbf{s}^i, 1 \le i \le |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|\}$ *compatible w.r.t.* $(\mathsf{leftpath}, l)$*, such that*

$$OPT(\mathcal{T}, \mathsf{leftpath}) \ge \sum_{i:\mathcal{T}^i \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))} OPT(\mathcal{T}^i, \mathbf{s}^i)$$

Proof. The proof of this lemma is the reverse of that of Lemma 5. Denote by $\mathcal{D}^*$ the decision tree returned by DP.

It suffices to show that we can transform $\mathcal{D}^*$ into a set of decision trees $\{\mathcal{D}^i, i = 1, \ldots, |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|\}$, such that $\mathcal{T}^i$ is a decision tree for the subproblem $(\mathcal{T}^i, \Pr, \mathsf{s}^\mathsf{i})$, where $\mathcal{D}^i \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))$ and $\{\mathsf{s}^\mathsf{i}\}$ is compatible w.r.t. $(\mathsf{leftpath}, l)$.

By constructing a set of decision trees, we mainly build their leftmost paths from top down and the rest is trivial. Starting from $j = 1$, each time we consider one of the following cases:

Case 1: $j < l$. If $\mathsf{leftpath}_j = 0$, then $s_j^i = 0, \forall i = 1, \ldots, |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))$, i.e., s_j^i is a reserved node for any i if $\mathsf{leftpath}_j$ is a reserved node.

If $\mathsf{leftpath}_j = 1$, denote by $i(j)$ the index such that $\mathsf{leftpath}_j$ is a question asked at some node of $\mathcal{T}^{i(j)} \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))$. We assign $\mathsf{leftpath}_j$ along with all its subtrees except the leftmost one to $s_j^{i(j)}$. For any $i \ne i(j)$, we set $s_j^i = 0$.

Case 2: $j = l$ and $l \le \mathsf{height}$. $\mathsf{leftpath}_j$ must be a question at $\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))$. We set $s_j^i = 0, \forall i = 1, \ldots, |\Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))|$. In addition, for any subtree of $\mathcal{D}'$ of $\mathcal{D}_{\mathsf{leftpath}_j}$, we find some $\mathcal{T}^i$ such that $\mathcal{D}'$ is a decision tree corresponding to $\mathcal{T}^i$, and assign $\mathcal{D}'$ as the only subtree of s_j^i. The set of decision trees are thus constructed.

It is trivial to verify that $\{\mathbf{s}^i\}$ is compatible w.r.t. $(\mathsf{leftpath}, l)$, and the constructed $\mathcal{D}^i$ is a feasible decision tree to the subproblem $(\mathcal{T}^i, \Pr, \mathbf{s}^i)$ of height at most height. Therefore, we have

$$\mathrm{OPT}(\mathcal{T}, \mathsf{leftpath}) \ge \sum_{i:\mathcal{T}^i \in \Phi_{\mathcal{T}}(\mathsf{root}(\mathcal{T}))} \mathrm{OPT}(\mathcal{T}^i, \mathbf{s}^i)$$

□

By Lemma 4, to solve the MDT problem, we can solve the RMDT problem $(\mathcal{T}, \Pr, \mathsf{leftpath} = \mathbf{1}_{\mathsf{height}})$ for sufficiently large height, which could be n and thus incur the running time of $O(n^2 2^n)$ in the worst case. However, the theorem below states that we only need to solve the RMDT problem in which height is much smaller than n:

Theorem 7. *For any* MDT *problem with instance*$(\mathcal{T}, \Pr)$*, there exists an optimal decision tree, of which the height is* $O(\log(\frac{1}{p_{\min}}) + \log n)$*, where* $p_{\min} \triangleq \min\{\Pr(u) | u \in \mathcal{T}, \Pr(u) > 0\}$.

To prove this theorem, we first need the following lemma:

Lemma 7. *Given any* MDT *problem with instance*$(\mathcal{T}, \Pr)$*, let* $\mathcal{D}^*$ *denote an optimal decision solution to the problem. Then* $\forall v \in \mathcal{D}^*$ *such that* $\mathsf{depth}_{\mathcal{D}^*}(v) \le 4$*, it holds that* $Pr(\mathcal{D}^*_v) \le Pr(\mathcal{D}^*)/2$.

Proof. The proof of this lemma is similar to (cic, 2014): Assume by contradiction that there exists $u^* \in \mathcal{D}^*$ such that $\mathsf{dep}_{\mathcal{D}^*} = 4$ but $Pr(\mathcal{D}^*_{u^*}) > Pr(\mathcal{D}^*)/2$. Denote $\mathcal{T}'$ by the category tree that corresponds to all the leaves of $\mathcal{D}^*_{v^*}$, i.e., all the leaves of $\mathcal{D}^*_{v^*}$ compose the subtree $\mathcal{T}'$.

Now consider the path in $\mathcal{D}^*$: $u_1 = \mathsf{root}(\mathcal{D}^*), u_2, u_3, u_4, u_5 = u^*$. Define $X \triangleq \{u_1, u_2, u_3, u_4\}$. Each element of X is a question asked at some node in $(\mathcal{T} - \mathcal{T}')$.

Recall that for any node $u \in \mathcal{T}$, $\Phi_{\mathcal{T}}(u)$ is denoted by all the resulting subtrees that is partitioned by u. It is worth noting that there exists a node $v \in \mathcal{T}$ such that for all any subtree $\mathcal{T}_0 \in \Phi_{\mathcal{T}}(v)$, it holds that $|\mathcal{T}_0 \cap X| \leq 2$. To see this, we can traverse $\mathcal{T}$ starting at $\mathsf{root}(\mathcal{T})$ and proceed as follows: suppose currently we visit u, if $|\mathcal{T}_u \cap X| \leq 2$, we return u as the target node; otherwise we set u as its child u' with largest $|\mathcal{T}_{u'} \cap X|$.

For any $\mathcal{T}_0 \in \Phi_{\mathcal{T}}(v)$, we construct a decision tree $\mathcal{D}_{\mathcal{T}_0}$ from $\mathcal{D}^*$: For any node $u \in \mathcal{D}$ such that u is a question asked at some node in $\mathcal{T} - \mathcal{T}_0$, if u is a question asked at an ancestor of $\mathsf{root}(\mathcal{T}_0)$, then there must be a child v of u such that leaves of $\mathcal{D}^*_v$ include at least one node of $\mathcal{T}_0$. In this case, we simply replace u with v and delete all the subtrees of u other than $\mathcal{D}^*_v$. Otherwise, we simply delete u and all its subtrees except the leftmost one. It is obvious that $\mathcal{D}_0$ is a feasible decision tree to $\mathcal{T}_0$.

In order to prove this lemma, we construct a decision tree $\mathcal{D}'$ as follows: $\mathcal{D}'$ takes v as its root. Below v, we use $\mathcal{D}_0$ to be the decision part for $\mathcal{T}_0 \in \Phi_{\mathcal{T}}(v)$. For any node $u \in \mathcal{T}$, it holds that,

$$\mathsf{dep}_{\mathcal{D}_0}(\mathsf{leaf}_{\mathcal{D}_0}(u)) = \mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) - 1 \tag{16}$$

In the following we compare $\mathcal{D}'$ and $\mathcal{D}^*$ w.r.t. the depth of the leaf associated with u. We consider two cases:

- If $u \in \mathcal{T}'$, since $\mathsf{leaf}_{\mathcal{D}^*}(u)$ is a descendant of u^* in $\mathcal{D}^*$, it follows that the path from $\mathsf{root}(\mathcal{D}^*)$ to $\mathsf{leaf}_{\mathcal{D}^*}(u)$ must contain all 4 nodes of X. On the other hand, from the construction of $\mathcal{D}_0$, we can see that at most 2 questions at elements of X are asked along the path from $\mathsf{root}(\mathcal{D}_0)$ to $\mathsf{leaf}_{\mathcal{D}_i}(u)$. Thus we have

  $$\mathsf{dep}_{\mathcal{D}_0}(\mathsf{leaf}_{\mathcal{D}_0}(u)) \leq \mathsf{dep}_{\mathcal{D}^*}(\mathsf{leaf}_{\mathcal{D}^*}(u)) - 2$$

  By combining this and (16), we obtain:

  $$\mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) \leq \mathsf{dep}_{\mathcal{D}^*}(\mathsf{leaf}_{\mathcal{D}^*}(u)) - 1$$

- If $u \in \mathcal{T} - \mathcal{T}'$, from the construction of $\mathcal{D}_0$, we have $\mathsf{dep}_{\mathcal{D}_0}(\mathsf{leaf}_{\mathcal{D}_0}(u)) \leq \mathsf{dep}_{\mathcal{D}^*}(\mathsf{leaf}_{\mathcal{D}^*}(u))$. Therefore, we have:

  $$\mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) \leq \mathsf{dep}_{\mathcal{D}^*}(\mathsf{leaf}_{\mathcal{D}^*}(u)) + 1$$

By combining the above two cases, we have

$$\begin{aligned}
\mathsf{cost}(\mathcal{D}') &= \sum_{u \in \mathcal{T}'} \mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) \Pr(u) \\
&\quad + \sum_{u \notin \mathcal{T}'} \mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) \Pr(u) \\
&\leq \sum_{u \in \mathcal{T}'} (\mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) - 1) \Pr(u) \\
&\quad + \sum_{u \notin \mathcal{T}'} (\mathsf{dep}_{\mathcal{D}'}(\mathsf{leaf}_{\mathcal{D}'}(u)) + 1) \Pr(u) \\
&= \mathsf{cost}(\mathcal{D}^*) - \Pr(\mathcal{T}') + \Pr(\mathcal{T} - \mathcal{T}') \\
&< \mathsf{cost}(\mathcal{D}^*)
\end{aligned}$$

where the second inequality holds since $\Pr(\mathcal{T}') > \Pr(\mathcal{T})/2$. This contradicts the optimality of $\mathcal{D}^*$ and thus the lemma follows. □

The lemma essentially states the *shrinkage* property of the optimal decision tree $\mathcal{D}^*$. That is, after a constant number of the questions, the resulting subtree weighs only a fraction of that of the original decision tree. This lemma guarantees that the optimal decision tree has the properties that it can distribute category candidates fairly "even" in terms of the probability, which is quite intuitive.

Finally, we can utilize Lemma 7 repeatedly and thus bound the height.

C.2. From DP to FPTAS

After bounding the height of the optimal decision tree, we employ standard scaling and rounding techniques of the probabilities in $\mathcal{T}$ in order to achieve an FPTAS.

The proof of Theorem 4 is much similar to the proof of Theorem 2 in (cic, 2014) and thus omitted.

D. Learning Component

D.1. Proof of Theorem 5

Proof. For any $t = 1, \ldots, |\mathcal{O}|$ (we do not assume that we know $|\mathcal{O}|$ in advance), we define $g(t) \triangleq \lfloor \log_2 t \rfloor$. Suppose that the t-th object arrives in our framework, we run the FPTAS with the approximation ratio of $\lambda/(6 * 2^{2g(t)})$.

From Theorem 1.1 in (Kalai & Vempala, 2005), we can see that in our problem, it holds that $|D| = n^2$ and $|A| = 1$. Denote by $\mathbb{E}\{\mathsf{cost}^*\}$ the expectation of the online cost by the framework for FPL and the optimal decision tree strategy. By substituting $\epsilon = \frac{\lambda}{3}$ into Theorem 1.1 of (Kalai

& Vempala, 2005), we obtain the following:

$$
\begin{aligned}
\mathbb{E}\{\mathsf{cost}\} &\leq \prod_{t=1}^{|\mathcal{O}|}\left(1+\frac{\lambda}{6*2^{2g(t)}}\right)\mathbb{E}\{\mathsf{cost}\} \\
&\leq \prod_{i=1}^{\lceil|\log_2 \mathcal{O}|\rceil+1}\left(1+\frac{\lambda}{6*2^{2i}}\right)^{i}\mathbb{E}\{\mathsf{cost}\} \\
&\leq \prod_{i=1}^{\lceil|\log_2 \mathcal{O}|\rceil+1}\left(1+\frac{\lambda}{6*2^{2i}}\right)^{i} * \\
&\quad \left[\left(1+\frac{\lambda}{3}\right)\mathsf{offline}^*(\mathcal{O})+\frac{12n^2(1+\ln n)}{\lambda}\right] \\
&\leq (1+\lambda)\mathsf{offline}^*(\mathcal{O})+O\left(\frac{n^2\ln n}{\lambda}\right)
\end{aligned}
$$

□

E. Extension to Parallelism

When publishing crowdsourcing tasks online, people often experience high latency since it takes time for workers to find the tasks on the crowdsourcing platform, and people need to spend time reading the instructions as well as completing the tasks. To reduce the latency, instead of asking one question at a time sequentially, one can ask several questions in a single Human Intelligence Task (HIT). In other words, multiple questions can be asked in parallel at the same time (we call it a *phase*), and we can proceed with further questions in next phase based on the answers we obtain.

E.1. Parallelism Workflow

In this section, we explore two dimensions of parallelism: One is that we may ask questions about several objects at the same time (of course, we may have to pay a bit more for such a HIT). Let m denote the number of objects that arrive at each time. The other is to ask multiple questions about the same object in a single HIT. Let k denote the number of questions asked at one object in each phase.

For example, consider the hierarchy in Figure 1. For the first dimension, we could ask three questions in one HIT, one about the Great Wall, one about the Blue House, and the last about the Taj Mahal. For the second dimension, we could ask questions at "India", "China" and "Korea" at the same time for the same photo.

Please refer to Algorithm 3 for the workflow. Each time the framework takes in at most m uncategorized objects in Line 2, it estimates the category distribution using FPL in Line 3, and adaptively constructs the decision tree using Greedy or the FPTAS in Line 4. Afterwards, for each object, it generates at most k questions in each phase until the target category is obtained (Lines 5-9). Finally, it updates the results of the categorized objects in Line 10.

Algorithm 3 ParallelFramework($\mathcal{T}, \mathcal{O}, \lambda, m, k$)

```
1: Input: $\mathcal{T}, \mathcal{O}, \lambda, m, k$
2: while there are uncategorized objects in $\mathcal{O}$ do
3:     Take in at most $m$ objects from $\mathcal{O}$
4:     Estimate the distribution by FPL($\lambda$)
5:     Construct the decision tree by Greedy (or the FPTAS)
6:     while the objects are not categorized do
7:         Generate $k$ questions for each object
8:         Post the questions on the crowdsourcing platform
9:         Collect answers.
10:    end while
11:    Update the results of the categorized objects.
12: end while
```

It is worth noting that an adaptive strategy for asking multiple questions can once again be modeled as a decision tree, in which each internal node represents a set of k questions (each corresponding to a different internal node in $\mathcal{T}$).

By Theorem 3.1, given $k \geq 1$, to construct the optimal decision tree in general is NP-hard. In the remainder of the section, we propose a heuristic algorithm for this problem.

E.2. Greedy Strategy in Parallel Version

We propose a greedy algorithm similar to the one in the previous section: we try to select k questions that makes the heaviest resulting subtree as light as possible. Let $U_0 = \{u_1^0, \ldots, u_k^0\}$ denote the selected questions. The strategy is described as follows:

$$U_0 = \arg\min_{U\subseteq\mathcal{T},|U|=k}\max\{\Pr(v), \forall v \in \Phi_{\mathcal{T}}(U)\} \tag{17}$$

However, it is not trivial to select the k questions in each phase. A naïve strategy would traverse the whole search space, incurring at least $O(\binom{n}{k})$ running time.

To solve the problem, we first solve a closely related problem: to select the minimum number of questions such that the heaviest resulting subtree does not exceed a given constant $w \in (0, 1]$. Using the algorithm for the above problem as a subroutine, we can solve the original problem by simply performing a binary search (identifying the largest w such that the number of required questions is at most k) [9].

Formally, the related problem is defined as follows:

Problem 4 (Partition Problem). *Given* ($\mathcal{T}, \Pr$) *and some*

---

[9] We are inspired by the ideas in (Parameswaran et al., 2011) and (Kundu & Misra, 1977).

constant $w \in (0, 1]$*, select the minimum number of questions such that for each resulting subtree, its total probability does not exceed* w*.*

Please refer to Algorithm 4 for the algorithm to the partition problem. The input to the Partition algorithm is the triple $(\mathcal{T}, \Pr, w)$. The algorithm is a simple greedy algorithm: We traverse the tree from the leaf level to the root (Lines 2-9). Each time we see a node u, we calculate the total probability of the subtree $\mathcal{T}_u$. We choose it as a question node if the total probability exceeds the threshold w (Lines 3-7).

Algorithm 4 Partition$(\mathcal{T}, \Pr, w)$

```
1: Input: $\mathcal{T}, \Pr, w$
2: $U \leftarrow \emptyset$
3: for i := maximum level in $T \rightarrow 1$ do
4:     for node $u$ on the $i$-th level do
5:         if $\Pr(\mathcal{T}_u) > w$ then
6:             $U \leftarrow U \cup \{u\}$
7:             $\mathsf{child}_{\mathcal{T}}(\mathsf{father}_{\mathcal{T}}(u) \leftarrow \mathsf{child}_{\mathcal{T}}(\mathsf{father}_{\mathcal{T}}(u)) - \{u\}$
8:         end if
9:     end for
10: end for
11: Output: $U$
```

The correctness of the algorithm is guaranteed by the following theorem:

Theorem 8. *Given any instance* $(\mathcal{T}, \Pr, w)$*, Partition correctly solves the partition problem.*

To prove the theorem, it suffices to prove the following two lemmas, whose proofs are similar to the ideas in (Kundu & Misra, 1977):

Lemma 8. *For any instance* $(\mathcal{T}, \Pr, w)$*, let* u *be a node such that* $Pr(\mathcal{T}_u) > w$*, then there exists an optimal partition containing* u*.*

Proof. It is trivial to see that any feasible partition U (including the optimal one) must contain some node $v \in \mathcal{T}_u$. From some optimal partition U^*, we construct $U' = U^* - v \cup u$. U' is a feasible partition. Thus the lemma follows from $|U'| = |U^*|$. ☐

Lemma 9. *For any* $v \in \mathcal{T}$*, denote by* $U^*(\mathcal{T}_v)$ *an optimal partition for* $\mathcal{T}_v$ *. Let* u *be a node which is included in some optimal partition. It follows that U is also an optimal partition for* $\mathcal{T}$*, which is constructed as follows:*

$$U = \{u\} \cup \bigcup_{\mathcal{T}' \in \Phi_{\mathcal{T}}(u)} U^*(\mathcal{T}')$$

where $U^*(\mathcal{T}')$ *is the optimal partition for* $\mathcal{T}'$*.*

Proof. Let U_0 be an optimal partition that includes u. It follows that for any $\mathcal{T}' \in \Phi_{\mathcal{T}}(u)$, $U_0(\mathcal{T}')$ is a feasible partition for $\mathcal{T}'$ and thus $|U_0(\mathcal{T}')| \geq |U^*(\mathcal{T}')|$. It follows that $|U| \leq |U_0|$ and therefore, U is an optimal partition for $\mathcal{T}$. ☐

F. Experimental Evaluation Continued

F.1. Synthetic Data Evaluation

Data Set: We took the whole ImageNet hierarchy in the synthetic experiment. Let the *a-priori* probability of the hierarchy be the fraction of images recorded in each synset.

Object Set: We randomly sampled 10,000 images from ImageNet, in which 80% of the images belong to the "artifact, artefact" sub-hierarchy while others (20%) were from other sub-hierarchies. Moreover, 80% images of artifacts are placed under the "instrumentality, instrumentation" sub-hierarchy. We repeated 20 times and recorded the costs as the average values.

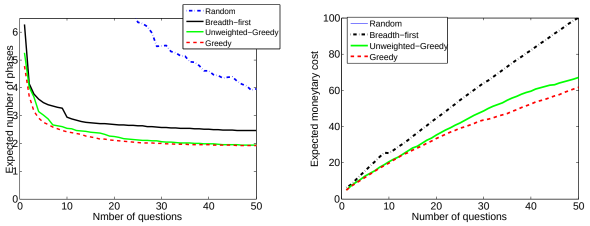
Decision Tree Algorithms: Besides the algorithms mentioned in Section 5, we implemented the following algorithms:

- Random: The algorithm selects k internal nodes as questions by performing a weighted sampling without replacement. For any internal node $u \in \mathcal{T}$, the weight assigned to it is the probability of u plus the total probability of children of u that are also leaves. In other words, the weight of picking a question at u is equal to the total probability of categories that can be directly obtained from u (for instance, in Figure 1, the weight of "Sought Asia" is 0.05 (Maldives) + 0 (South Asia)=0.05). In this way, the total weight of all the internal nodes adds up to 1. Intuitively, this is a reasonable heuristic since it is better to first pick nodes with larger probability so that they are closer to the root. Each time we executed the algorithm for 20 times and recorded the average performance.
- Unweighted-Greedy (Parameswaran et al., 2011): Each time the algorithm chooses k nodes in order to minimize the *size* of the resulting subtree in the worst case. This is equivalent to the greedy strategy on the hierarchy, in which each node is equally distributed.

- **Framework Parameters:** The parameters of our framework we considered in the experiments are as follows:
    - Number of questions k asked for one object at each phase (Section E).
    - Number of arriving objects at each time m (Sec-

tion E).

- Coefficient of the exponential distribution λ: The default value is 0.2. The range is $[0.1, 1]$.



(a) Expected number of phases.

(b) Expected monetary cost.

Figure 11. Expected number of phases and expected monetary cost varying k.

Evaluation on decision tree algorithms:

Expected number of phases: (1) the expected number of phases decreases as k increases. (2) The expected value of Greedy is slightly lower than that of Unweighted-Greedy and both outperform Breadth-first and Random.

Please refer to Figure 11(a) for the experiment that illustrates the results. The difference of the expected # phases between Greedy and the minimum of other algorithms ranges from 0.05 to 0.2.

Expected Monetary Cost: (1) the expected monetary cost is almost a linear function of k. (2) the monetary cost difference between Greedy and other algorithms increases as k increases.

Recall in Section E, that we focus on minimizing the steps or phases of categorization, where in each phase we post k questions on crowdsourcing platforms for each item. However, in practice, the monetary cost for each phase increases as k increases. The more questions we batch into one HIT, the higher reward we should pay to crowdsourcing workers. By assuming that each question costs a unit price (for instance, \$0.02 per question), we studied the expected monetary cost by using the different strategies, varying on k, shown in Figure 11(b).

Recall that asking multiple questions at the same time would reduce the total time we wait for crowdsourcing tasks. However, as depicted in Figure 11(b), we would incur more monetary cost by distributing parallel tasks. Therefore, a time-money tradeoff must be carefully considered by crowdsourcing users, which is beyond the scope of this paper and will be studied in future work.

Evaluation on the framework: Next we examine how our framework for FPL and Greedy captures the true category distribution and approaches the offline near-optimal cost. We evaluate the robustness of the framework with respect to different algorithm parameters.

By Theorem 3.1, the offline optimal cost is NP-hard. Therefore, we select some decision tree algorithm A and use the offline cost incurred by A to approximate the offline optimal cost.

For a fixed input sequence of the object set, after t images are categorized (using the decision tree constructed by A), define $\mathsf{offline}(A,t)$ and $\mathsf{online}(A,t)$ as the average offline cost and average online cost, respectively. Define $\mathsf{ratio}(A,t)$ as follows:

$$\mathsf{ratio}(A,t) \triangleq \frac{\mathsf{online}(A,t)}{\mathsf{offline}(A,t)} \qquad (18)$$

We first compared the online and offline costs of different algorithms:

Online and offline Costs of different algorithms: (1) For all algorithms, the online cost approaches the offline cost as t increases. (2) The ratio of Unweighted-Greedy and Breadth-first is "immune" to the input object, slightly better than that of Greedy, and outperforms Random. (3) Greedy yields less online cost than that of either Breadth-first or Random, and they all outperform Random.

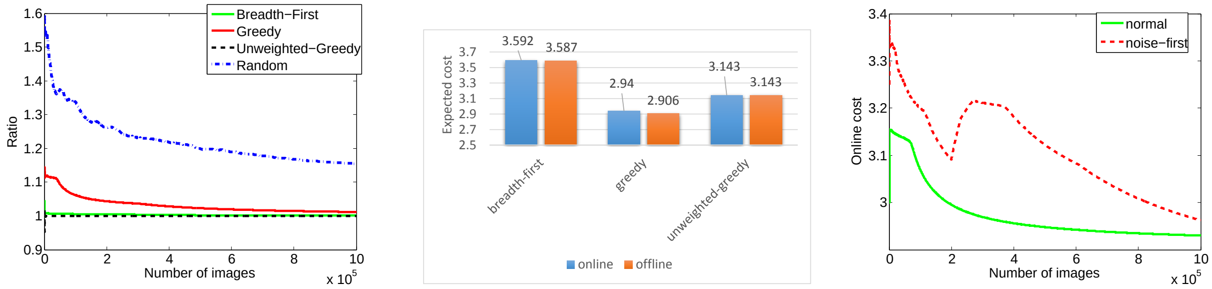
In the experiments, we set $k = 4, \lambda = 0.2, m = 10,000$. Please refer to Figure 12(a) and Figure 12(b) for the ratios and average costs of different algorithms. The online costs of Greedy gradually reaches the online cost as t increases, finally arriving at 1.012 and the overall online cost of Greedy is below the costs of Unweighted-Greedy and Breadth-first, which incurs 7% and 18% savings, respectively.

During our experiments, for each randomly generated image set, the online and offline costs of Greedy, Unweighted-Greedy and Breadth-first remained stable, which somewhat fit the form in Theorem 5. However, for Random, during the executions, the online costs of were relatively stable, the offline costs drastically changed.

In the remainder of the section, we evaluate different parameters of of the framework, in which Greedy was used for constructing the decision trees.

Results on the Input Sequence: The framework can quickly adapt to the change of distribution and finally approaches the offline cost.

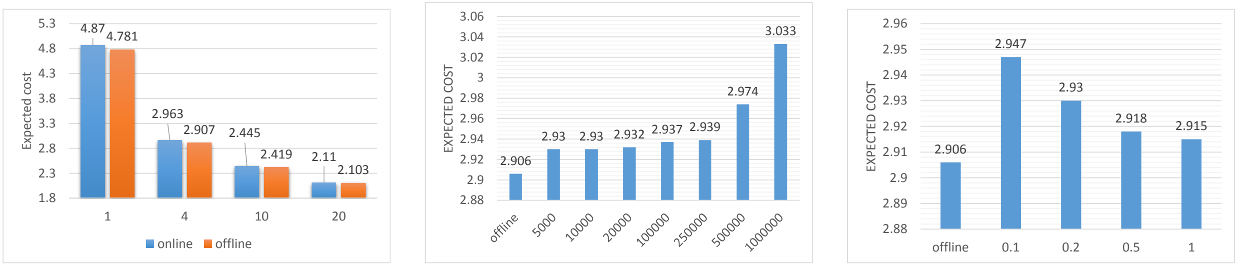
In our second experiment, we changed the sequence of the input to evaluate the robustness of the framework, depicted in Figure 12(c). The offline average cost cost is 2.906. First, we selected the sequence such that the 20% noise images (i.e., images from other sub-hierarchies) of the set



(a) Ratio w.r.t. t.

(b) Online and offline costs of different algorithms.

(c) Varying the input sequence.



(d) Varying k.

(e) Varying m.

(f) Varying λ.

Figure 12. Evaluation on the framework

were randomly distributed in the sequence, and obtained the online cost of 2.93. Then, we changed the sequence such that the first 20% of input were all noise. From Figure 12(c), we could observe that the framework first gradually learned the distribution of the noises. After 20,0000 noisy images, the framework first incurred a lot of additional costs to categorize the informative images (i.e., images from the "artifact" sub-hierarchy). However, it quickly adapted itself to the change of distribution, and finally achieved an online cost of 2.962, which was quite close to both the online costs in the random-noise setting and the offline costs.

Varying k: For different number questions asked in one phase, the online cost approaches the offline cost closely.

Figure 12(d) illustrates the above result.

Varying m: The online cost of the framework increases as m increases.

From Figure 12(e), we can see that the marginal effect of updating the estimated distribution more frequently is, indeed, quite limited. For instance, if we are only allowed to update the distribution 9 times ($m = 100,000$), the online cost (2.937) is quite close to the online cost (2.930) in the case where we update the estimated distribution 199 times ($m = 5,000$).

It is worth noting that even if we are allowed to update the distribution only once ($m = 500,000$), the resulting online cost (2.974) is significantly lower than the cost if we simply use the decision tree based on the initial distribution.

Varying λ: The online cost of the framework slightly decreases as m increases.

Figure 12(f) shows the result of the framework varying λ. Recall that we have proven that the online average cost of the FPTAS can be arbitrarily close to the offline optimal average cost. From Figure 12(f), the online cost varying λ is quite close to the offline cost cost, when using Greedy in the decision tree component.