Decision Trees for Function Evaluation
Simultaneous Optimization of Worst and Expected Cost[*]

Ferdinando Cicalese
University of Salerno, Italy
cicalese@dia.unisa.it

Eduardo Laber
PUC-Rio, Brazil
laber@inf.puc-rio.br

Aline Medeiros Saettler
PUC-Rio, Brazil
alinemsaettler@gmail.com

October 2, 2018

Abstract

In several applications of automatic diagnosis and active learning a central problem is the evaluation of a discrete function by adaptively querying the values of its variables until the values read uniquely determine the value of the function. In general, the process of reading the value of a variable might involve some cost, computational or even a fee to be paid for the experiment required for obtaining the value. This cost should be taken into account when deciding the next variable to read. The goal is to design a strategy for evaluating the function incurring little cost (in the worst case or in expectation according to a prior distribution on the possible variables' assignments).

Our algorithm builds a strategy (decision tree) which attains a logarithmic approximation simultaneously for the expected and worst cost spent. This is best possible under the assumption that $\mathcal{P} \neq \mathcal{NP}$.

1 Introduction

In order to introduce the problem we analyze in the paper, let us start with some motivating examples.

In high frequency trading, an automatic agent decides the next action to be performed as sending or canceling a buy/sell order, on the basis of some market variables as well as private variables (e.g., stock price, traded volume, volatility, order books distributions as well as complex relations among these variables). For instance in [32] the trading strategy is learned in the form of a discrete function, described as a table, that has to be evaluated whenever a new scenario is faced and an action (sell/buy) has to be taken. The rows of the table represent the possible scenarios of the market and the columns represent the variables taken into account by the agent to distinguish among the different scenarios. For each scenario, there is an associated action. Every time an action need to be taken, the agent can identify the scenario by computing the value of each single variable and proceed with the associated action. However, recomputing

---

all the variable every time might be very expensive. By taking into account the structure of the function/table together with information on the probability distribution on the scenarios of the market and also the fact that some variables are more expensive (or time consuming) to calculate than others, the algorithm could limit itself to recalculate only some variables whose values determine the action to be taken. Such an approach can significantly speed up the evaluation of the function. Since market conditions change on a millisecond basis, being able to react very quickly to a new scenario is the key to a profitable strategy.

In a classical Bayesian active learning problem, the task is to select the right hypothesis from a possibly very large set $\mathcal{H} = \{h_1, \ldots, h_n\}$. Each $h \in \mathcal{H}$ is a mapping from a set $\mathcal{X}$ called the query/test space to the set (of labels) $\{1, \ldots, \ell\}$. It is assumed that the functions in $\mathcal{H}$ are unique, i.e., for each pair of them there is at least one point in $\mathcal{X}$ where they differ. There is one function $h^* \in \mathcal{H}$ which provides the correct labeling of the space $\mathcal{X}$ and the task is to identify it through queries/tests. A query/test coincides with an element $x \in \mathcal{X}$ and the result is the value $h^*(x)$. Each test x has an associated cost $c(x)$ that must be paid in order to acquire the response $h^*(x)$, since the process of labeling an example may be expensive either in terms of time or money (e.g. annotating a document). The goal is to identify the correct hypothesis spending as little as possible. For instance, in automatic diagnosis, $\mathcal{H}$ represents the set of possible diagnoses and $\mathcal{X}$ the set of symptoms or medical tests, with h^* being the exact diagnosis that has to be achieved by reducing the cost of the examinations.

In [4], a more general variant of the problem was considered where rather than the diagnosis it is important to identify the therapy (e.g., for cases of poisoning it is important to quickly understand which antidote to administer rather than identifying the exact poisoning). This problem can be modeled by defining a partition $\mathcal{P}$ on $\mathcal{H}$ with each class of $\mathcal{P}$ representing the subset of diagnoses which requires the same therapy. The problem is then how to identify the class of the exact h^* rather than h^* itself. This model has also been studied by Golovin et al. [15] to tackle the problem of erroneous tests' responses in Bayesian active learning.

The above examples can all be cast into the following general problem.

The Discrete Function Evaluation Problem (DFEP). An instance of the problem is defined by a quintuple $(S, C, T, \mathbf{p}, \mathbf{c})$, where $S = \{s_1, \ldots, s_n\}$ is a set of objects, $C = \{C_1, \ldots, C_m\}$ is a partition of S into m classes, T is a set of tests, $\mathbf{p}$ is a probability distribution on S, and $\mathbf{c}$ is a cost function assigning to each test t a cost $c(t) \in \mathbb{N}^+$. A test $t \in T$, when applied to an object $s \in S$, incurs a cost $c(t)$ and outputs a number $t(s)$ in the set $\{1, \ldots, \ell\}$. It is assumed that the set of tests is complete, in the sense that for any distinct $s_1, s_2 \in S$ there exists a test t such that $t(s_1) \neq t(s_2)$. The goal is to define a testing procedure which uses tests from T and minimizes the testing cost (in expectation and/or in the worst case) for identifying the class of an unknown object s^* chosen according to the distribution $\mathbf{p}$.

The DFEP can be rephrased in terms of minimizing the cost of evaluating a discrete function that maps points (corresponding to objects) from some finite subset of $\{1, \ldots, \ell\}^{|T|}$ into values (corresponding to classes), where an object $s \in S$ corresponds to the point $(t_1(s), \ldots, t_{|T|}(s))$ obtained by applying each test of T to s. This perspective motivates the name we chose for the problem. However, for the sake of uniformity with more recent work [15, 4] we employ the definition of the problem in terms of objects/tests/classes.

Decision Tree Optimization. Any testing procedure can be represented by a *decision tree*, which is a tree where every internal node is associated with a test and every leaf is associated with a set of objects that belong to the same class. More formally, a decision tree D for

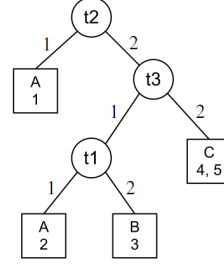| Object	t1	t2	t3	Class	Probability
1	1	1	2	A	0.1
2	1	2	1	A	0.2
3	2	2	1	B	0.4
4	1	2	2	C	0.25
5	2	2	2	C	0.05

Figure 1: Decision tree for 5 objects presented in the table in the left, with $c(t1) = 2$, $c(t2) = 1$ and $c(t3) = 3$. Letters and numbers in the leaves indicate, respectively, classes and objects.

$(S, C, T, \mathbf{p}, \mathbf{c})$ is a leaf associated with class i if every object of S belongs to the same class i. Otherwise, the root r of D is associated with some test $t \in T$ and the children of r are decision trees for the sets $\{S_t^1, ..., S_t^\ell\}$, where S_t^i, for $i = 1, \ldots, \ell$, is the subset of S that outputs i for test t.

Given a decision tree D, rooted at r, we can identify the class of an unknown object s^* by following a path from r to a leaf as follows: first, we ask for the result of the test associated with r when performed on s^*; then, we follow the branch of r associated with the result of the test to reach a child r_i of r; next, we apply the same steps recursively for the decision tree rooted at r_i. The procedure ends when a leaf is reached, which determines the class of s^*.

We define $cost(D, s)$ as the sum of the tests' cost on the root-to-leaf path from the root of D to the leaf associated with object s. Then, the *worst testing cost* and the *expected testing cost* of D are, respectively, defined as

$$cost_W(D) = \max_{s \in S}\{cost(D, s)\} \tag{1}$$

$$cost_E(D) = \sum_{s \in S} cost(D, s)p(s) \tag{2}$$

Figure 1 shows an instance of the DFEP and a decision tree for it. The tree has worst testing cost $1+3+2 = 6$ and expected testing cost $(1 \times 0.1) + (6 \times 0.2) + (6 \times 0.4) + (4 \times 0.3) = 4.9$.

Our Results. Our main result is an algorithm that builds a decision tree whose expected testing cost and worst testing cost are at most $O(\log n)$ times the minimum possible expected testing cost and the minimum possible worst testing cost, respectively. In other words, the decision tree built by our algorithm achieves simultaneously the best possible approximation achievable with respect to both the expected testing cost and the worst testing cost. In fact, for the special case where each object defines a distinct class—known as the *identification problem*— both the minimization of the expected testing cost and the minimization of the worst testing cost do not admit a sub-logarithmic approximation unless $P = NP$, as shown in [5] and in [26], respectively. In addition, in Section 4, we show that the same inapproximability results holds in general for the case of exactly m classes for any $m \geq 2$.

It should be noted that in general there are instances for which the decision tree that minimizes the expected testing cost has worst testing cost much larger than that achieved by the decision tree with minimum worst testing cost. Also there are instances where the converse happens. Therefore, it is reasonable to ask whether it is possible to construct decision trees that

3

are efficient with respect to both performance criteria. This might be important in practical applications where only an estimate of the probability distribution is available which is not very accurate. Also, in medical applications like the one depicted in [4], very high cost (or equivalently significantly time consuming therapy identification) might have disastrous/deadly consequences. In such cases, besides being able to minimize the expected testing cost, it is important to guarantee that the worst testing cost also is not large (compared with the optimal worst testing cost).

With respect to the minimization of the expected testing cost, our result improves upon the previous $O(\log 1/p_{\min})$ approximation shown in [15] and [4], where p_{min} is the minimum positive probability among the objects in S. From the result in these papers an $O(\log n)$ approximation could be attained only for the particular case of uniform costs via a technique used in [25].

From a high-level perspective, our method closely follows the one used by Gupta *et al.* [21] for obtaining the $O(\log n)$ approximation for the expected testing cost in the identification problem. Both constructions of the decision tree consist of building a path (backbone) that splits the input instance into smaller ones, for which decision trees are recursively constructed and attached as children of the nodes in the path.

A closer look, however, reveals that our algorithm is much simpler than the one presented in [21]. First, it is more transparently linked to the structure of the problem, which remained somehow hidden in [21] where the result was obtained via an involved mapping from adaptive TSP. Second, our algorithm avoids expensive computational steps as the Sviridenko procedure [35] and some non-intuitive/redundant steps that are used to select the tests for the backbone of the tree. In fact, we believe that providing an algorithm that is much simpler to implement and an alternative proof of the result in [21] is an additional contribution of this paper.

State of the art. The DFEP has been recently studied under the names of class equivalence problem [15] and group identification problem [4] and long before it had been described in the excellent survey by Moret [28]. Both [15] and [4] give $O(\log(1/p_{min}))$ approximation algorithms for the version of the DFEP where the expected testing cost has to be minimized and both the probabilities and the testing costs are non-uniform. In addition, when the testing costs are uniform both algorithms can be converted into a $O(\log n)$ approximation algorithm via Kosaraju approach [25]. The algorithm in [15] is more general because it addresses multiway tests rather than binary ones. For the minimization of the worst testing cost, Moshkov has studied the problem in the general case of multiway tests and non-uniform costs and provided an $O(\log n)$-approximation in [30]. In the same paper it is also proved that no $o(\log n)$-approximation algorithm is possible under standard the complexity assumption $NP \not\subseteq DTIME(n^{O(\log \log n)})$. The minimization of the worst testing cost is also investigated in [20] under the framework of covering and learning.

The particular case of the DFEP where each object belongs to a different class—known as the *identification problem*—has been more extensively investigated [11, 1, 5, 6]. Both the minimization of the worst and the expected testing cost do not admit a sublogarithmic approximation unless $P = NP$ as proved by [26] and [5]. For the expected testing cost, in the variant with multiway tests, non uniform probabilities and non uniform testing costs, an $O(\log(1/p_{min}))$ approximation is given by Guillory and Blimes in [18]. Gupta *et al.* [21] improved this result to $O(\log n)$ employing new techniques not relying on the Generalized Binary Search (GBS)—the basis of all the previous strategies.

An $O(\log n)$ approximation algorithm for the minimization of the worst testing cost for the

identification problem has been given by Arkin et. al. [3] for binary tests and uniform cost and by Hanneke [22] for case with mutiway tests and non-uniform testing costs.

In the case of Boolean functions, the DFEP is also known as Stochastic Boolean Function Evaluation (SBFE), where the distribution over the possible assignments is a product distribution defined by assuming that variable x_i has a given probability $p(x_i)$ of being one independently of the value of the other variables. Another difference with respect to the DFEP as it is presented here, is that in Stochastic Boolean Function Evaluation the common assumption is that the complete set of associations between the assignments of the variables and the value of the function is provided, directly or via a representation of the function, e.g., in terms of its DNF or CNF. The present definition of DFEP considers the more general problem where only a sample of the Boolean function is given and from this we want to construct a decision tree with minimum expected costs and that exactly fits the sample.

Results on the exact solution of the SBFE for different classes of Boolean functions can be found in the survey paper [37]. In a recent paper Deshpande et al. [12], provide a 3-approximation algorithm for evaluating Boolean linear threshold formulas and an $O(\log kd)$ approximation algorithm for the evaluation of CDNF formulas, where k and d is the number of clauses of the input CNF and d is the number of terms of the input DNF. The same result had been previously obtained by Kaplan et al. [24] for the case of monotone formulas and uniform distribution (in a slightly different setting). Both algorithms of [12] are based on reducing the problem to Stochastic Submodular Set Cover introduced by Golovin and Krause [16] and providing a new algorithm for this latter problem.

Other special cases of the DFEP like the evaluation of AND/OR trees (a.k.a. read-once formulas) and the evaluation of Game Trees (a central task in the design of game procedures) are discussed in [36, 34, 17]. In [7], Charikar *et al.* considered discrete function evaluation from the perspective of competitive analysis; results in this alternative setting are also given in [24, 8].

2 Preliminaries

Given an instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, we will denote by $OPT_E(I)$ ($OPT_W(I)$) the expected testing cost (worst testing cost) of a decision tree with minimum possible expected testing cost (worst testing cost) over the instance I. When the instance I is clear from the context, we will also use the notation $OPT_W(S)$ ($OPT_E(S)$) for the above quantity, referring only to the set of objects involved. We use p_{min} to denote the smallest non-zero probability among the objects in S.

Let $(S, T, C, \mathbf{p}, \mathbf{c})$ be an instance of DFEP and let S' be a subset of S. In addition, let C', $\mathbf{p}'$ and $\mathbf{c}'$ be, respectively, the restrictions of C, $\mathbf{p}$ and $\mathbf{c}$ to the set S'. Our first observation is that every decision tree D for $(S, C, T, \mathbf{p}, \mathbf{c})$ is also a decision tree for the instance $I' = (S', C', T, \mathbf{p}', \mathbf{c}')$. The following proposition immediately follows.

Proposition 1. *Let $I = (S, C, T, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP and let S' be a subset of S. Then, $OPT_E(I') \leq OPT_E(I)$ and $OPT_W(I') \leq OPT_W(I)$, where $I' = (S', C', T, \mathbf{p}', \mathbf{c}')$ is the restriction of I to S'.*

One of the measures of progress of our strategy is expressed in terms of the number of pairs of objects belonging to different classes which are present in the set of objects satisfying the tests already performed. The following definition formalizes this concept of pairs for a given set of objects.

Definition 1 (Pairs). *Let $I = (S, T, C, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP and $G \subseteq S$. We say that two objects $x, y \in S$ constitute a pair of G if they both belong to G but come from different classes. We denote by $P(G)$ the number of pairs of G. In formulae, we have*

$$P(G) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_i(G) n_j(G)$$

where for $1 \le i \le m$ and $A \subseteq S$, $n_i(A)$ denotes the number of objects in A belonging to class C_i.

As an example, for the set of objects S in Figure 1 we have $P(S) = 8$ and the following set of pairs $\{(1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5)\}$.

We will use s^* to denote the initially unknown object whose class we want to identify. Let $\mathbf{t}$ be a sequence of tests applied to identify the class of s^* (it corresponds to a path in the decision tree) and let G be the set of objects that agree with the outcomes of all tests in $\mathbf{t}$. If $P(G) = 0$, then all objects in G belong to the same class, which must coincide with the class of the selected object s^*. Hence, $P(G) = 0$ indicates the identification of the class of the object s^*. Notice that s^* might still be unknown when the condition $P(G) = 0$ is reached.

For each test $t \in T$ and for each $i = 1, \ldots, \ell$, let $S_t^i \subseteq S$ be the set of objects for which the outcome of test t is i. For a test t, the outcome resulting in the largest number of pairs is of special interest for our strategy. We denote with S_t^* the set among $S_t^1, \ldots, S_t^\ell$ such that $P(S_t^*) = \max\{P(S_t^1), \ldots, P(S_t^\ell)\}$ (ties are broken arbitrarily). We denote with $\sigma_S(t)$ the set of objects not included in S_t^*, i.e., we define $\sigma_S(t) = S \setminus S_t^*$. Whenever S is clear from the context we use $\sigma(t)$ instead of $\sigma_S(t)$.

Given a set of objects S, each test produces a tripartition of the pairs in S: the ones with both objects in $\sigma(t)$, those with both objects in S_t^* and those with one object in $\sigma(t)$ and one object in S_t^*. We say that the pairs in $\sigma(t)$ are *kept* by t and the pairs with one object from $\sigma(t)$ and one object from S_t^* are *separated* by t. We also say that a pair is *covered* by the test t if it is either kept or separated by t. Analogously, we say that a test t covers an object s if $s \in \sigma(t)$.

For any set of objects $Q \subseteq S$ the probability of Q is $p(Q) = \sum_{s \in Q} p(s)$.

3 Logarithmic approximation for the Expected Testing Cost and the Worst Case Testing Cost

In this section, we describe our algorithm `DecTree` and analyze its performance. The concept of the separation cost of a sequence of tests will turn useful for defining and analyzing our algorithm.

The separation cost of a sequence of tests. Given an instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, for a sequence of tests $\mathbf{t} = t_1, t_2, \ldots, t_q$, we define the separation cost of $\mathbf{t}$ in the instance I, denoted by $sepcost(I, \mathbf{t})$, as follows: Fix an object x. If there exists $j < q$ such that $x \in \sigma(t_j)$ then we set $i(x) = \min\{j \mid x \in \sigma(t_j)\}$. If $x \notin \sigma(t_j)$ for each $j = 1, \ldots, q-1$, then we set $i(x) = q$. Let $sepcost(I, \mathbf{t}, x) = \sum_{j=1}^{i(x)} c(t_j)$ denote the *cost of separating x in the instance I by means of the sequence $\mathbf{t}$*. Then, the *separation cost of $\mathbf{t}$* (in the instance I) is defined by

$$sepcost(I, \mathbf{t}) = \sum_{s \in S} p(s) sepcost(I, \mathbf{t}, s). \tag{3}$$

In addition, we define $totcost(I, \mathbf{t})$ as the total cost of the sequence $\mathbf{t}$, i.e.,

$$totcost(I, \mathbf{t}) = \sum_{j=1}^{q} c(t_j).$$

Lower bounds on the cost of an optimal decision tree for the DFEP. We denote by $sepcost^*(I)$ the minimum separation cost in I attainable by a sequence of tests in T which covers all the pairs in S and $totcost^*(I)$ as the minimum total cost attainable by a sequence of tests in T which covers all the pairs in S.

The following theorem shows lower bounds on both the expected testing cost and the worst case testing cost of any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP.

Theorem 1. *For any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DEFP, it holds that $sepcost^*(I) \leq OPT_E(I)$ and $totcost^*(I) \leq OPT_W(I)$.*

Proof. Let D be a decision tree for the instance I. Let $t_1, t_2, \ldots, t_q, l$ be the nodes in the root-to-leaf path in D such that for each $i = 2, \ldots, q$, the node t_i is on the branch stemming from t_{i-1} which is associated with $S^*_{t_{i-1}}$, and the leaf node l is the child of t_q associated with the objects in $S^*_{t_q}$.

Let $\mathbf{t} = t_1, t_2, \ldots, t_q$. Abusing notation let us now denote with t_i the test associated with the node t_i so that $\mathbf{t}$ is a sequence of tests. In particular, $\mathbf{t}$ is the sequence of tests performed according to the strategy defined by D when the object s^* whose class we want to identify, is such that $s^* \in S^*_t$ holds for each test t performed in the sequence.

Notice that, by construction, $\mathbf{t}$ is a sequence of tests covering all pairs of S.

Claim. For each object s it holds that $sepcost(I, \mathbf{t}, s) \leq cost(D, s)$.

If for each $i = 1, \ldots, q$, we have that $s \notin \sigma(t_i)$ then it holds that $cost(D, s) = \sum_{j=1}^{q} c(t_j) = sepcost(I, \mathbf{t}, s)$. Conversely, let t_i be the first test in $\mathbf{t}$ for which $s \in \sigma(t_i)$. Therefore, we have that $t_1, t_2, \ldots, t_i$ is a prefix of the root to leaf path followed when s is the object chosen. It follows that $cost(D, s) \geq \sum_{j=1}^{i} c(t_j) = sepcost(I, \mathbf{t}, s)$. The claim is proved.

In order to prove the first statement of the theorem, we let D be a decision tree which achieves the minimum possible expected cost, i.e., $cost_E(D) = OPT_E(I)$. Then, we have

$$sepcost^*(I) \leq sepcost(I, \mathbf{t}) = \sum_{s \in S} p(s) sepcost(I, \mathbf{t}, s) \leq \sum_{s \in S} p(s) cost(D, s) = OPT_E(I). \quad (4)$$

In order to prove the second statement of the theorem, we let D be a decision tree which achieves the minimum possible worst testing cost, i.e., $cost_W(D) = OPT_W(D)$. Let $s \in S$ be such that, for each $j = 1, \ldots, q-1$, it holds that $s \notin \sigma(t_j)$. Then, by the above claim it follows that

$$totcost(I, \mathbf{t}) = sepcost(I, \mathbf{t}, s) \leq cost(D, s) \leq cost_W(D). \quad (5)$$

Using (5), we have

$$totcost^*(I) \leq totcost(I, \mathbf{t}) \leq cost_W(D) = OPT_W(I). \quad (6)$$

The proof is complete. $\qquad\square$

The following subadditivity property will be useful.

Proposition 2 (Subadditivity). *Let $S_1, S_2, \ldots, S_q$ be a partition of the object set S. We have $OPT_E(S) \geq \sum_{j=1}^{q} OPT_E(S_j)$ and $OPT_W(S) \geq \max_{j=1}^{q}\{OPT_W(S_j)\}$, where $OPT_E(S_j)$ and $OPT_W(S_j)$ are, respectively, the minimum expected testing cost and the worst case testing cost when the set of objects is S_j.*

The optimization of submodular functions of sets of tests. Let $I = (S, T, C, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP. A set function $f : 2^T \mapsto \mathbb{R}_+$ is submodular non-decreasing if for every $R \subseteq R' \subseteq T$ and every $t \in T \setminus R'$, it holds that $f(R \cup \{t\}) - f(R) \geq f(R' \cup \{t\}) - f(R')$ (submodularity) and $f(R) \leq f(R')$ (non-decreasing).

It is easy to verify that the functions

$$f_1 : R \subseteq T \mapsto P(S) - P(\bigcap_{t \in R} S_t^*)$$

$$f_2 : R \subseteq T \mapsto p(S) - p(\bigcap_{t \in R} S_t^*)$$

are non-negative non-decreasing submodular set functions. In words, f_1 is the function mapping a set of tests R into the number of pairs covered by the tests in R. The function f_2, instead, maps a set of tests R into the probability of the set of objects covered by the tests in R.

Let B be a positive integer. Consider the following optimization problem defined over a non-negative, non-decreasing, sub modular function f:

$$\mathcal{P}(f, B, T, \mathbf{c}) : \max_{R \subseteq T} \left\{ f(R) : \sum_{t \in R} c(t) \leq B \right\}. \tag{7}$$

In [38], Wolsey studied the solution to the problem $\mathcal{P}$ provided by Algorithm 1 below, called the adapted greedy heuristic.

Algorithm 1 Wolsey greedy algorithm

Procedure `Adapted-Greedy`$(S, T, f, \mathbf{c}, B)$

1: $spent \leftarrow 0, \ A \leftarrow \emptyset, \ k \leftarrow 0$
2: **repeat**
3:    $k \leftarrow k + 1$
4:    let t_k be a test t which maximizes $\frac{f(A \cup \{t\}) - f(A)}{c(t)}$ among all $t \in T$ s.t. $c(t) \leq B$
5:    $T \leftarrow T \setminus \{t_k\}, \ spent \leftarrow spent + c(t_k), \ A \leftarrow A \cup \{t_k\}$
6: **until** $spent > B$ or $T = \emptyset$
7: **if** $f(\{t_k\}) \geq f(A \setminus \{t_k\})$ **then** Return $\{t_k\}$
       else Return $\{t_1 \, t_2 \, \ldots \, t_{k-1}\}$

The following theorem summarizes results from [38][Theorems 2 and 3].

Theorem 2. *[38] Let R^* be the solution of the problem $\mathcal{P}$ and $\overline{R}$ be the set returned by Algorithm 1. Moreover, let e be the base of the natural logarithm and χ be the solution of $e^\chi = 2 - \chi$. Then we have that $f(\overline{R}) \geq (1 - e^{-\chi}) f(R^*) \approx 0.35 f(R^*)$. Moreover, if there exists c such $c(t) = c$ for each $t \in T$ and c divides B, then we have $f(\overline{R}) \geq (1 - 1/e) f(R^*) \approx 0.63 f(R^*)$.*

Corollary 1. *Let $\mathbf{t} = t_1 \ldots t_{k-1} t_k$ be the sequence of all the tests selected by* `Adapted-Greedy`, *i.e., the concatenation of the two possible outputs in line 7. Then, we have that the total cost of the tests in $\mathbf{t}$ is at most $2B$ and $f(\{t_1, \ldots, t_{k-1}, t_k\}) \geq (1 - e^{-\chi}) f(R^*) \approx 0.35 f(R^*)$.*

Our algorithm for building a decision tree will employ this greedy heuristic for finding approximate solutions to the optimization problem $\mathcal{P}$ over the submodular set functions f_1 and f_2 defined in (7).

3.1 Achieving logarithmic approximation

We will show that Algorithm 2 attains a logarithmic approximation for DFEP. The algorithm consists of 4 blocks. The first block (lines 1-2) is the basis of the recursion, which returns a leaf if all objects belong to the same class ($P(S) = 0$). If $P(S) = 1$, we have that $|S| = 2$ and the algorithm returns a tree that consists of a root and two leaves, one for each object, where the root is associated with the cheapest test that separates these two objects. Clearly, this tree is optimal for both the expected testing cost and the worst testing cost.

The second block (line 3) calls procedure `FindBudget` to define the budget B allowed for the tests selected in the third and fourth blocks. `FindBudget` finds the smallest B such that `Adapted-Greedy`$(S, T, f_1, \mathbf{c}, B)$ returns a set of tests R covering at least $\alpha P(S)$ pairs.

Algorithm 2

Procedure `DecTree`$(S, T, C, \mathbf{p}, \mathbf{c})$

1: **If** $P(S) = 0$ **then return** a single leaf l associated with S
2: **If** $P(S) = 1$ **then return** a tree whose root is the cheapest test that separates the two objects in S
3: B=`FindBudget`$(S, T, C, \mathbf{c})$, $\quad spent \leftarrow 0$, $spent_2 \leftarrow 0$, $U \leftarrow S$, $k \leftarrow 1$
4: **while** there is a test in T of cost $\leq B - spent$ **do**
5: $\quad$ let t_k be a test which maximizes $\frac{p(U)-p(U \cap S_t^*)}{c(t)}$ among all tests t s.t. $t \in T$ and $c(t) \leq B - spent$
6: $\quad$ **If** $k = 1$ **then** make t_1 root of D **else** t_k child of t_{k-1}
7: $\quad$ **for** every $i \in \{1, \ldots, \ell\}$ such that $(S_{t_k}^i \cap U) \neq \emptyset$ **and** $S_{t_k}^i \neq S_{t_k}^*$ **do**
8: $\quad\quad$ Make $D^i \leftarrow$ `DecTree`$(S_{t_k}^i \cap U, T, C, \mathbf{p}, \mathbf{c})$ child of t_k
9: $\quad$ $U \leftarrow U \cap S_{t_k}^*$, $spent \leftarrow spent + c(t_k)$, $T \leftarrow T \setminus \{t_k\}$, $\quad k \leftarrow k + 1$
10: **end while**
11: **repeat**
12: $\quad$ let t_k be a test which maximizes $\frac{P(U)-P(U \cap S_t^*)}{c(t)}$ among all tests $t \in T$ s.t. $c(t) \leq B$
13: $\quad$ Set t_k as a child of t_{k-1}
14: $\quad$ **for** every $i \in \{1, \ldots, \ell\}$ such that $(S_{t_k}^i \cap U) \neq \emptyset$ **and** $S_{t_k}^i \neq S_{t_k}^*$ **do**
15: $\quad\quad$ Make $D^i \leftarrow$`DecTree`$(U \cap S_{t_k}^i, T, C, \mathbf{p}, \mathbf{c})$ child of t_k
16: $\quad$ $U \leftarrow U \cap S_{t_k}^*$, $spent_2 \leftarrow spent_2 + c(t_k)$, $T \leftarrow T \setminus \{t_k\}$, $\quad k \leftarrow k + 1$
17: **until** $B - spent_2 < 0$ **or** $T = \emptyset$
18: $D' \leftarrow$ `DecTree`$(U, T, C, \mathbf{p}, \mathbf{c})$; Make D' a child of t_{k-1}
19: Return the decision tree D constructed by the algorithm

Procedure `FindBudget`$(S, T, C, \mathbf{c})$

1: Let $f : R \subseteq T \mapsto P(S) - P(\bigcap_{t \in R} S_t^*)$ and let $\alpha = 1 - e^\chi \approx 0.35$
2: Do a binary search in the interval $[1, \sum_{t \in T} c(t)]$ to find the smallest B such that `Adapted-Greedy`$(S, T, f, \mathbf{c}, B)$ returns a set of tests R covering at least $\alpha P(S)$ pairs
3: Return B

The third (lines 4-10) and the fourth (lines 11-17) blocks are responsible for the construction of the backbone of the decision tree (see Fig. 2) as well as to call `DecTree` recursively to construct the decision trees that are children of the nodes in the backbone.

The third block (the **while** loop in lines 4-10) constructs the first part of the backbone (sequence $\mathbf{t}^A$ in Fig. 2) by iteratively selecting the test that covers the maximum uncovered

mass probability per unit of testing cost (line 5). The selected test t_k induces a partition $(U^1_{t_k}, \ldots, U^\ell_{t_k})$ on the set of objects U, which contains the objects that have not been covered yet. In lines 7 and 8, the procedure is recursively called for each set of this partition but for the one that is contained in the subset $S^*_{t_k}$. With reference to Figure 2, these calls will build the subtrees rooted at nodes not in $\mathbf{t}^A$ which are children of some node in $\mathbf{t}^A$.

Similarly, the fourth block (the **repeat-until** loop) constructs the second part of the backbone (sequence $\mathbf{t}^B$ in Fig. 2) by iteratively selecting the test that covers the maximum number of uncovered pairs per unit of testing cost (line 12). The line 18 is responsible for building a decision tree for the objects that are not covered by the tests in the backbone.

We shall note that both the third and the fourth block of the algorithm are based on the adapted greedy heuristic of Algorithm 1. In fact, $p(U) - p(U \cap S^*_{t_k})$ in line 5 (third block) corresponds to $f_2(A \cup t_k) - f_2(A)$ in Algorithm 1 because, right before the selection of the k-th test, A is the set of tests $\{t_1, \ldots, t_{k-1}\}$ and $U = \cap_{i=1}^{k-1} S^*_{t_i}$. Thus,

$$f_2(A \cup t_k) = p(S) - p(\cap_{i=1}^k S^*_{t_i}) = p(S) - p(U \cap S^*_{t_k})$$

and

$$f_2(A) = p(S) - p(\cap_{i=1}^{k-1} S^*_{t_i}) = p(S) - p(U)$$

so that

$$f_2(A \cup t_k) - f_2(A) = p(U) - p(U \cap S^*_{t_k}).$$

A similar argument shows that $P(U) - P(U \cap S^*_t)$ in line 12 (fourth block) corresponds to $f_1(A \cup t_k) - f_1(A)$ in Algorithm 1. These connections will allow us to apply both Theorem 2 and Corollary 1 to analyze the cost and the coverage of these sequences.
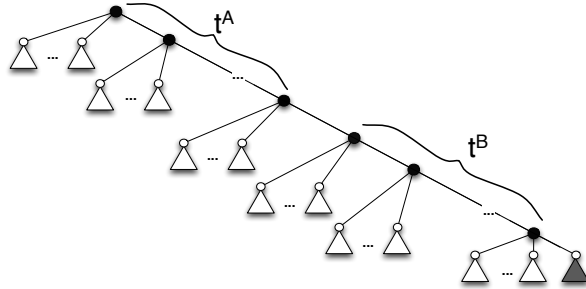

Figure 2: The structure of the decision tree built by `DecTree`: white nodes correspond to recursive calls. In each white subtree, the number of pairs is at most $P(S)/2$, while in the lowest-right gray subtree it is at most $8/9 P(S)$ (see the proof of Theorem 4).

Let $\mathbf{t}_I$ denote the sequence of tests obtained by concatenating the tests selected in the **while** loop and in the **repeat-until** loop of the execution of `DecTree` over instance I. We delay to the next section the proof of the following key result.

Theorem 3. *Let χ be the solution of $e^\chi = 2 - \chi$, and $\alpha = 1 - e^\chi \approx 0.35$. There exists a constant $\delta \geq 1$, such that for any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, the sequence $\mathbf{t}_I$ covers at least $\alpha^2 P(S) > \frac{1}{9} P(S)$ pairs, and it holds that $sepcost(I, \mathbf{t}_I) \leq \delta \cdot sepcost^*(I)$ and $totcost(I, \mathbf{t}_I) \leq 3 totcost^*(I)$.*

Applying Theorem 3 to each recursive call of `DecTree` we can prove the following theorem about the approximation guaranteed by our algorithm both in terms of worst testing cost and expected testing cost.

10

Theorem 4. *For any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP, the algorithm* `DecTree` *outputs a decision tree with expected testing cost at most $O(\log(n)) \cdot OPT_E(I)$ and with worst testing cost at most $O(\log(n)) \cdot OPT_W(I)$.*

Proof. For any instance I, let $D^{\mathbb{A}}(I)$ be the decision tree produced by the algorithm `DecTree`. First, we prove an approximation for the expected testing cost. Let β be such that $\beta \log \frac{9}{8} = \delta$, where δ is the constant given in the statement of Theorem 3. Let us assume by induction that the algorithm guarantees approximation $1 + \beta \log P(G)$, for the expected testing cost, for every instance I' on a set of objects G with $1 \leq P(G) < P(S)$.

Let $\mathcal{I}$ be the set of instances on which the algorithm `DecTree` is recursively called in lines 8,15 and 18. We have that

$$\frac{cost_E(D^{\mathbb{A}}(I))}{OPT_E(I)} = \frac{sepcost(I, \mathbf{t}_I) + \sum_{I' \in \mathcal{I}} cost_E(D^{\mathbb{A}}(I'))}{OPT_E(I)} \tag{8}$$

$$\leq \frac{sepcost(I, \mathbf{t}_I)}{OPT_E(I)} + \max_{I' \in \mathcal{I}} \frac{cost_E(D^{\mathbb{A}}(I'))}{OPT_E(I')} \tag{9}$$

$$\leq \delta + \max_{I' \in \mathcal{I}} \frac{cost_E(D^{\mathbb{A}}(I'))}{OPT_E(I')} \tag{10}$$

$$\leq \delta + \max_{I' \in \mathcal{I}} \left\{ 1 + \beta \log P(I') \right\} \tag{11}$$

$$\leq \delta + 1 + \beta \log 8P(S)/9 = 1 + \beta log(P(S)). \tag{12}$$

The first equality follows by the recursive way the algorithm `DecTree` builds the decision tree. Inequality (9) follows from (8) by the subadditivity property (Proposition 2) and simple algebraic manipulations. The inequality in (10) follows by Theorem 3 together with Theorem 1 yielding $sepcost(I, \mathbf{t}_I) \leq \delta \, OPT_E(I)$. The inequality (11) follows by induction (we are using $P(I')$ to denote the number of pairs of instance I').

To prove that the inequality in (12) holds we have to argue that every instance $I' \in \mathcal{I}$ has at most $\frac{8}{9} P(S)$ pairs. Let $U_{t_k}^i = S_{t_k}^i \cap U$ as in the lines 8 and 15. First we show that the number of pairs of $U_{t_k}^i$ is at most $P(S)/2$. We have $S_{t_k}^i \neq S_{t_k}^*$ and $S_{t_k}^*$ is the set with the maximum number of pairs in the partition $\{S_{t_k}^1, \dots, S_{t_k}^\ell\}$, induced by t_k on the set S. It follows that $P(U_{t_k}^i) \leq P(S_{t_k}^i) \leq P(S)/2$. Now it remains to show that the instance I', recursively called, in line 18 has at most $8/9P(S)$ pairs. This is true because the number of pairs of I' is equal to the number of pairs not covered by $\mathbf{t}_I$ which is bounded by $(1 - \alpha^2)P(S) \leq 8P(S)/9$ by Theorem 3.

Now, we prove an approximation for the worst testing cost of the tree $D^{\mathbb{A}}(I)$. Let ρ be such that $\rho \log \frac{9}{8} = 3$. Let us assume by induction that the worst testing cost of $D^{\mathbb{A}}(I')$ is at most $(1 + \rho \log P(G) \cdot OPT_W(I'))$ for every instance I' on a set of objects G with $1 \leq P(G) < P(S)$. We have that

$$\frac{cost_W(D^{\mathbb{A}}(I))}{OPT_W(I)} \leq \frac{totcost(I, \mathbf{t}_I) + \max_{I' \in \mathcal{I}}\{cost_W(D^{\mathbb{A}}(I'))\}}{OPT_W(I)} \tag{13}$$

$$\leq \frac{totcost(I, \mathbf{t}_I)}{OPT_W(I)} + \max_{I' \in \mathcal{I}} \frac{cost_W(D^{\mathbb{A}}(I'))}{OPT_W(I')} \tag{14}$$

$$\leq \frac{totcost(I, \mathbf{t}_I)}{totcost^*(I)} + \max_{I' \in \mathcal{I}} \frac{cost_W(D^{\mathbb{A}}(I'))}{OPT_W(I')} \tag{15}$$

$$\leq 3 + 1 + \rho \log(8P(S)/9) = 1 + \rho \log(P(S)) \tag{16}$$

11

Inequality (14) follows from the subadditivity property (Proposition 2) for the worst testing cost. The inequality (15) follows by Theorem 1. The inequality (16) follows from Theorem 3, the induction hypothesis (we are using $P(I')$ to denote the number of pairs of instance I') and from the fact mentioned above that every instance in $\mathcal{I}$ has at most $8/9 P(S)$ pairs.

Since $P(S) \leq n^2$ it follows that the algorithm provides an $O(\log n)$ approximation for both the expected testing cost and the worst testing cost.

$\square$

The previous theorem shows that algorithm `DecTree` provides simultaneously logarithmic approximation for the minimization of expected testing cost and worst testing cost. We would like to remark that this is an interesting feature of our algorithm. In this respect, let us consider the following instance of the DFEP[1]: Let $S = \{s_1, \ldots, s_n\}$; $p_i = 2^{-i}$, for $i = 1, \ldots, n-1$ and $p_n = 2^{-(n-1)}$; the set of tests is in one to one correspondence with the set of all binary strings of length n so that the test corresponding to a binary string $\mathbf{b}$ outputs $0(1)$ for object s_i if and only if the ith bit of $\mathbf{b}$ is $0(1)$. Moreover, all tests have unitary costs. This instance is also an instance of the problem of constructing an optimal prefix coding binary tree, which can be solved by the Huffman's algorithm [10]. Let D_E^* and D_W^* be, respectively, the decision trees with minimum expected cost and minimum worst testing cost for this example. Using Huffman's algorithm, it is not difficult to verify that $Cost_E(D_E^*) \leq 3$ and $Cost_W(D_E^*) = n-1$. In addition, we have that $Cost_E(D_W^*) = Cost_W(D_W^*) = \log n$. This example shows that the minimization of the expected testing cost may result in high worst testing cost and vice versa the minimization of the worst testing cost may result in high expected testing cost. Clearly, in real situations presenting such a dichotomy, the ability of our algorithm to optimize simultaneously both measures of cost might provide a significant gain over strategies only guaranteeing competitiveness with respect to one measure.

3.2 The proof of Theorem 3

We now return to the proof of Theorem 3 for which will go through three lemmas.

Lemma 1. *For any instance* $I = (S, C, T, \mathbf{p}, \mathbf{c})$ *of the DFEP, the value* B *returned by the procedure* `FindBudget`$(S, T, C, \mathbf{c})$ *satisfies* $B \leq totcost^*(I)$.

Proof. Let us consider the problem $\mathcal{P}$ in equation (7) with the function f_1 that measures the number of pairs covered by a set of tests. Let $G(x)$ be the number of pairs covered by the solution constructed with `Adapted-Greedy` when the budget—the righthand side of equation (7)—is x. By construction, `FindBudget` finds the smallest B such that $G(B) \geq \alpha P(S)$.

Let $\tilde{\mathbf{t}}$ be a sequence that covers all pairs in S and that satisfies $totcost(\tilde{\mathbf{t}}) = totcost^*(I)$. Arguing by contradiction we can show that $totcost(I, \tilde{\mathbf{t}}) \geq B$. Suppose that this was not the case, then $\tilde{\mathbf{t}}$ would be the sequence which covers $P(S)$ pairs using a sequence of tests of total cost not larger than some $B' < B$. By Theorem 2, the procedure `Adapted-Greedy` provides an α-approximation of the maximum number of pairs covered with a given budget. Therefore, when run with budget B', `Adapted-Greedy` is guaranteed to produce a sequence of total cost $\leq B'$ which covers at least $\alpha P(S)$ pairs. However, by the minimality of B it follows that such a sequence does not exist. Since this contradiction follows by the hypothesis $totcost(I, \tilde{\mathbf{t}}) < B$, it must hold that $totcost^*(I) \geq totcost(I, \tilde{\mathbf{t}}) \geq B$, as desired. $\square$

---

[1] This is also an instance of the identification problem mentioned in the introduction

Given an instance I, for a sequence of tests $\mathbf{t} = t_1, \ldots, t_k$ and a real $K > 0$, let $sepcost_K(I, \mathbf{t})$ be the separation cost of $\mathbf{t}$ when every non-covered object is charged K, that is,

$$sepcost_K(I, \mathbf{t}) = \sum_{\substack{x \in S \\ x \text{ is covered by } \mathbf{t}}} p(x) sepcost(I, \mathbf{t}, x) + \sum_{\substack{x \in S \\ x \text{ is not covered by } \mathbf{t}}} p(x) \cdot K$$

The proofs of the following technical lemma is deferred to the appendix.

Lemma 2. *Let $\mathbf{t}^A$ be the sequence obtained by concatenating the tests selected in the* **while** *loop of Algorithm 2. Then, $totcost(I, \mathbf{t}^A) \leq B$ and $sepcost_B(I, \mathbf{t}^A) \leq \gamma \cdot sepcost^*(I)$, where γ is a positive constant and B is the budget calculated at line 3.*

Lemma 3. *The sequence $\mathbf{t}_I$ covers at least $\alpha^2 P(S)$ pairs and it holds that $totcost(I, \mathbf{t}_I) \leq 3B$.*

Proof. The sequence $\mathbf{t}_I$ can be decomposed into the sequences $\mathbf{t}^A$ and $\mathbf{t}^B$, that are constructed, respectively, in the **while** and **repeat-until** loop of the algorithm `DecTree` (see also Fig. 2).

It follows from the definition of B that there is a sequence of tests, say $\mathbf{t}$, of total cost not larger than B that covers at least $\alpha P(S)$ pairs for instance I. Let z be the number of pairs of instance I covered by the sequence $\mathbf{t}^A$. Thus, the tests in $\mathbf{t}$, that do not belong to $\mathbf{t}^A$, cover at least $\alpha P(S) - z$ pairs in the set $U = \bigcap_{t \in \mathbf{t}^A} S_t^*$ of objects not covered by $\mathbf{t}^A$.

The sequence $\mathbf{t}^B$ coincides with the concatenation of the two possible outputs of the procedure `Adapted-Greedy`$(U, T - \mathbf{t}^A, f', \mathbf{c}, B)$ (Algorithm 1), when it is executed on the instance defined by: the objects in U (those not covered by $\mathbf{t}^A$); the tests that are not in $\mathbf{t}^A$; the submodular set function $f' : R \subseteq T - \mathbf{t}^A \mapsto P(S) - P(U \cap (\bigcap_{t \in R} S_t^*))$ and bound B. By Corollary 1, we have that $totcost(I, \mathbf{t}^B) \leq 2B$ and $\mathbf{t}^B$ covers at least $\alpha(\alpha P(S) - z)$ uncovered pairs.

Therefore, since $totcost(I, \mathbf{t}^A) \leq B$, altogether, we have that $\mathbf{t}_I$ covers at least $z + \alpha(\alpha P(S) - z) \geq \alpha^2 P(S)$ pairs and $totcost(I, \mathbf{t}_I) \leq 3B$. $\qquad\square$

The proof of Theorem 3 will now follow by combining the previous three lemmas.

Proof of Theorem 3. First, it follows from Lemma 3 that $\mathbf{t}_I$ covers at least $\alpha^2 P(S)$ pairs.

To prove that $sepcost(I, \mathbf{t}_I) \leq sepcost^*(I)$, we decompose $\mathbf{t}_I$ into $\mathbf{t}^A = t_1^A, \ldots, t_q^A$ and $\mathbf{t}^B = t_1^B, \ldots, t_r^B$, the sequences of tests selected in the **while** and in the **repeat-until** loop of Algorithm 2, respectively.

For $i = 1, \ldots, q$, let $\pi_i = \sigma(t_i^A) \setminus (\bigcup_{j=1}^{i-1} \sigma(t_j^A))$. In addition, let π_A be the set of objects which are not covered by the tests in $\mathbf{t}^A$. Thus,

$$
\begin{aligned}
sepcost(I, \mathbf{t}_I) &\leq \sum_{i=1}^{q} p(\pi_i) \left(\sum_{j=1}^{i} c(t_j^A) \right) + 3B \cdot p(\pi_A) \\
&\leq 3 sepcost_B(I, \mathbf{t}^A) \leq 3\gamma sepcost^*(I),
\end{aligned}
$$

where the last inequality follows from Lemma 2.

It remains to show that $totcost(I, \mathbf{t}_I) \leq 3 totcost^*(I)$. This inequality holds because Lemma 3 assures that $totcost(I, \mathbf{t}_I) \leq 3B$ and Lemma 1 assures that $totcost^*(I) \geq B$. The proof is complete.

4 $O(\log n)$ is the best possible approximation.

Let $U = \{u_1, \ldots, u_n\}$ be a set of n elements and $\mathcal{F}$ be a family of subsets of U. The minimum set cover problem asks for a family $\mathcal{F}' \subseteq \mathcal{F}$ of minimum cardinality such that $\bigcup_{F \in \mathcal{F}'} F = U$. It is known that no sub logarithmic approximation is achievable for the minimum set cover problem under the standard assumption that $P \neq NP$. More precisely, by the result of Raz and Safra [33] it follows that there exists a constant $\tilde{k} > 0$ such that no $\tilde{k}\, log_2 n$-approximation algorithm for the minimum set cover problem exists unless $P = NP$ [33, 13].

   We will show that an $o(\log n)$ approximation algorithm for the minimization of the expected testing cost DFEP with exact b classes for $b \geq 2$, implies that the same approximation can be achieved for the Minimum Set Cover problem. This implies that one cannot expect to obtain a sublogarithmic approximation for the DFEP unless $P = NP$. The reduction we present can also be used to show the same inapproximability result for the minimization of the worst testing cost version of the DFEP.

   Given an instance $I_{SC} = (U, \mathcal{F})$ for the minimum set cover problem as defined above, we construct an instance $I_{DFEP} = (S, C, T, \mathbf{p}, \mathbf{c})$ for the DFEP as follows: The set of objects is $S = U \cup \{o_1, \ldots, o_{b-1}\}$. The family of classes $C = (C_0, \ldots, C_{b-1})$ is defined as follows: All the objects of U belong to class C_0 while the object o_i, for $i = 1, \ldots, b-1$, belongs to class C_i. Notice that b_1 and the objects of U belong to the same class. In order to define the set of tests T, we proceed as follows: For each set $F \in \mathcal{F}$ we create a test t_F such that t_F has value 0 for the objects in F and value 1 for the remaining objects. In addition, we create a test $\tilde{t}$ which has value 0 for objects in U and value $i - 1$ for object o_i $(1 \leq i \leq b-1)$. For our later purposes, we notice here that the test $\tilde{t}$ cannot distinguish between b_1 and the elements in U.

   Each test has cost 1, i.e., the cost assignment $\mathbf{c}$ is given by $c(t) = 1$ for each $t \in T$. Finally, we set the probability of o_1 to be equal to $1 - (n + b - 2)\eta$ and the probability of the other objects equal to η, for some fixed $\eta < \frac{1}{2(n+b-2)}$.

   Let D^* be the decision tree with minimum expected testing cost for I_{DFEP} and let $\mathcal{F}^* = \{F_1, \ldots, F_h\}$ be a minimum set cover for instance $I_{SC} = (U, \mathcal{F})$, where $h = |\mathcal{F}^*|$.

   We first argue that $cost_E(D^*) \leq h + 1$. In fact, we can construct a decision tree D by putting the test t_{F_1} associated with F_1 in the root of the tree, then the test t_{F_2} associated with F_2 as the child of t_{F_1} and so on. Notice that, for $i = 1, \ldots, h-1$ we have that t_{F_i} has two children, one is $t_{F_{i+1}}$ and the other is a leaf mapping to the class C_0. As for $t_F h$, one of its children in again a leaf mapping to C_0, the other child is set to the test $\tilde{t}$, whose children are all leaves.

   The expected testing cost of D can be upper bounded by

$$OPT_E(I_{DFEP}) = cost_E(D^*) \leq cost_E(D) \leq (h + 1) = OPT(I_{SC}) + 1 \tag{17}$$

since we have $cost(D, s) = (h + 1)$ for any $s \in \{o_1, \ldots, o_{b-1}\}$ and $cost(D, s) \leq h + 1$ for any $s \in U$.

   On the other hand, let D be a decision tree for I_{DFEP} and let P be the path from the root of D to the leaf where the object o_1 lies. It is easy to realize that the subsets associated with the tests on this path cover all the elements in U—in fact these tests separate o_1 from all the other objects from U. Let $\mathcal{T}$ be the solution to the set cover problem provided by the sets associated with the tests on the path P. We have that

$$|\mathcal{T}| \leq cost(D, o_1) \leq \frac{cost_E(D)}{1 - \eta(n + b - 2)} \leq 2cost_E(D), \tag{18}$$

in the last inequality we are using the fact that $\eta \leq \frac{1}{2(n+b-2)}$.

Now assume that there is an algorithm that for any instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP can guarantee a solution with approximation $\alpha \log |S|$ for some $\alpha < \tilde{k}/8$. Therefore, given an instance $I_{SC} = (U, \mathcal{F})$ for set cover we can use this algorithm on the transformed instance I_{DFEP} defined above, where $|S| = |U| + b - 1$. We obtain a decision tree D for I_{DFEP} such that

$$cost_E(D) \leq \alpha \log(n+b-1) OPT_E(I_{DFEP}) \leq \alpha(OPT(I_{SC}) + 1) \log(n+b) \leq 4\alpha \log n OPT(I_{SC})$$

where we upper bound $OPT(I_{SC}) + 1 \leq 2 OPT(I_{SC})$ and $\log(n+b) \leq 2 \log n$.

From D, as seen above we can construct a solution $\mathcal{T}$ for the set cover problem such that $|\mathcal{T}| \leq 2 cost_E(D)$. Hence, it would follow that $\mathcal{T}$ is an approximate solution for the set cover instance satisfying:

$$|\mathcal{T}| \leq 8\alpha \log n OPT(I_{SC}) < (\tilde{k} \log n) OPT(I_{SC})$$

which by the result of [33] is not possible unless $P = NP$.

The same construction can be used for analyzing the case of the worst testing cost, in which case we have that (17) becomes $OPT_W(I_{DFEP}) \leq OPT(I_{SC}) + 1$ and (18) becomes $|\mathcal{T}| \leq cost_W(D)$, leading to the inapproximability of the DFEP w.r.t. the worst testing cost within a factor of $\alpha \log n$ for any $\alpha < \tilde{k}/4$. Notice that an analogous result regarding the worst testing cost had been previously shown by Moshkov [29] based on the result of Feige [13].

Thus, we have the following theorem

Theorem 5. *Both the minimization of the worst case and the expected case of the DFEP don't admit an $o(\log n)$ approximation unless $P = NP$.*

5 Final remarks and Future Directions

We presented a new algorithm for the discrete function evaluation problem, a generalization of the classical Bayesian active learning also studied under the names of Equivalence Class Determination Problem [15] and Group Based Active Query Selection problem of [4]. Our algorithm builds a decision tree which asymptotically matches simultaneously for the expected and the worst testing cost the best possible approximation achievable under standard complexity assumptions $P \neq NP$. This way, we close the gap left open by the previous $O(\log 1/p_{\min})$ approximation for the expected cost shown in[15] and [4], where p_{min} is the minimum positive probability among the objects in S and in addition we show that this can be done with an algorithm that guarantees the best possible approximation also with respect to the worst testing cost.

With regards to the broader context of learning, given a set of samples labeled according to an unknown function, a standard task in machine learning is to find a good approximation of the labeling function (hypothesis). In order to guarantee that the hypothesis chosen has some generalization power w.r.t. to the set of samples, we should avoid overfitting. When the learning is performed via decision tree induction this implies that we shall not have leaves associated with a small number of samples so that we end up with a decision tree that have leaves associated with more than one label. There are many strategies available in the literature to induce such a tree.

In the problem considered in this paper our aim is to over-fit the data because the function is known a priori and we are interested in obtaining a decision tree that allows us to identify the label of a new sample with the minimum possible cost (time/money). The theoretical results we obtain for the "fitting" problem should be generalizable to the problem of approximating the function. To this aim we could employ the framework of covering and learning from [20] along the following lines: we would interrupt the recursive process in Algorithm 2 through which we construct the tree as soon as we reach a certain level of learning (fitting) w.r.t. the set of labeled samples. Then, it remains to show that our decision tree is at logarithmic factor of the optimal one for that level of learning. This is an interesting direction for future research.

References

[1] Adler, M. and Heeringa, B. Approximating optimal binary decision trees. AP-PROX/RANDOM '08, pp. 1–9, 2008.

[2] Allen, S.R., Hellerstein, L., Kletenik, D, and Ünlüyurt, T. Evaluation of DNF Formulas. In *Proc. of ISAIM 2014*.

[3] Arkin, E. M., Meijer, H., Mitchell, J.S.B., Rappaport, D. and Skiena, S.S. Decision trees for geometric models. In *Proc. of SCG '93*, pp. 369–378, 1993.

[4] Bellala, G., Bhavnani, S. K., and Scott, C. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Trs. Inf. Theor.*, 58(1):459–478, 2012.

[5] Chakaravarthy, V. T., Pandit, V., Roy, S., Awasthi, P., and Mohania, M. Decision trees for entity identification: approximation algorithms and hardness results. In *Proc. PODS '07*, pp. 53–62, 2007.

[6] Chakaravarthy, V. T., Pandit, V., Roy, S., and Sabharwal, Y. Approximating decision trees with multiway branches. In *Proc. ICALP '09*, pp. 210–221, 2009.

[7] Charikar, M., Fagin, R., Guruswami, V., Kleinberg, J. M., Raghavan, P., and Sahai, A. Query strategies for priced information. *Journal of Computer and System Sciences*, 64 (4):785–819, 2002.

[8] Cicalese, F. and Laber, E. S. On the competitive ratio of evaluating priced functions. *J. ACM*, 58(3):9, 2011.

[9] Cicalese, F., Jacobs, T., Laber, E., and Molinaro, M. On greedy algorithms for decision trees. In *Proc. of ISAAC'10*, 2010.

[10] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[11] Dasgupta, S. Analysis of a greedy active learning strategy. In *NIPS'04*, 2004.

[12] Deshpande, A., Hellerstein, L., and Kletenik, D.. Approximation Algorithms for Stochastic Boolean Function Evaluation and Stochastic Submodular Set Cover. In *Proc. of SODA 2014*, pp. 1453–1467, 2014.

[13] Feige, U. A threshold of $\ln n$ for approximating set cover. *Journal of ACM* 45 (1998) 634–652.

[14] Garey, M. Optimal binary identification procedures. *SIAM Journal on Applied Mathematics*, 23(2):173–186, 1972.

[15] Golovin, D., Krause, A., and Ray, D. Near-optimal bayesian active learning with noisy observations. In *Proc. of NIPS'10*, pp. 766–774, 2010.

[16] Golovin, D., Krause, A.. Adaptive Submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, Vol. 42, pages 427-486, 2011.

[17] Greiner, R., Howard, R., Jankowska, M., and Malloy, M. Finding optimal satisfiscing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2005.

[18] Guillory, A. and Bilmes, J. Average-case active learning with costs. In *Proc. of ALT'09*, pp. 141–155, 2009.

[19] Guillory, A. and Bilmes, J. Interactive submodular set cover. In *Proc. ICML'10*, pp. 415–422. 2010.

[20] Guillory, A. and Bilmes, J. Simultaneous learning and covering with adversarial noise. *ICML'11*, pp. 369–376. 2011.

[21] Gupta, A., Nagarajan, V., and Ravi, R. Approximation algorithms for optimal decision trees and adaptive tsp problems. In *Proc. ICALP'10*, pp. 690–701, 2010.

[22] Hanneke, S. The cost complexity of interactive learning. *unpublished*, 2006.

[23] Hyafil, L. and Rivest, R. L. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[24] Kaplan, H., Kushilevitz, E., and Mansour, Y.. Learning with attribute costs. In *Proc. of STOC 2005*, pp. 356–365, 2005.

[25] Kosaraju, S. Rao, Przytycka, Teresa M., and Borgstrom, Ryan S. On an optimal split tree problem. In *Proc. of WADS '99*, pp. 157–168, 1999.

[26] Laber, E. S. and Nogueira, L. T. On the hardness of the minimum height decision tree problem. *Discrete Appl. Math.*, 144:209–212, 2004.

[27] Larmore, L. L. and Hirschberg, D. S. A fast algorithm for optimal length-limited huffman codes. *JACM: Journal of the ACM*, 37, 1990.

[28] Moret, B. M. E. Decision Trees and Diagrams. *ACM Computing Surveys*, pp. 593–623, 1982.

[29] Moshkov, J.M. Approximate Algorithm for Minimization of Decision Tree Depth. In *Proc. of 9th Intl. Conf. on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pp. 611-614, 2003.

[30] Moshkov, J.M. Greedy Algorithm with Weights for Decision Tree Construction. *Fundamentae Informaticae*, 104: 285–292, 2010.

[31] Nemhauser, G., Wolsey, L., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions-i. *Math. Programming*, 14:265–294, 1978.

[32] Nevmyvaka, Y., Feng, Y., and Kearns, M.. Reinforcement learning for optimized trade execution. In *Proc. of ICML'06*, pp. 673–680, 2006.

[33] Raz, R. and Safra, S.. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. *Proc. 29th Ann. ACM Symp. on Theory of Comp.*, ACM, 1997, pp. 475–484.

[34] Saks, M. E. and Wigderson, A. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *Proc. of FOCS'86*, pp. 29–38, 1986.

[35] Sviridenko, M. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41 – 43, 2004.

[36] Tarsi, M. Optimal search on some game trees. *Journal of the ACM*, 30(3):389–396, July 1983.

[37] Ünlüyurt, T. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004.

[38] Wolsey, L. Maximising real-valued submodular functions. *Math. of Operation Research*, 7(3):410–425, 1982.

A The proof of Lemma 2

Lemma 2. *Let $\mathbf{t}^A$ be the sequence obtained by concatenating the tests selected in the* **while** *loop of Algorithm 2. Then, $totcost(I, \mathbf{t}^A) \leq B$ and $sepcost_B(I, \mathbf{t}^A) \leq \gamma \cdot sepcost^*(I)$, where γ is a positive constant and B is the budget calculated at line 3.*

Proof. Clearly, the Algorithm 2 in the **while** loop constructs a sequence $\mathbf{t}^A$ such that

$$totcost(I, \mathbf{t}^A) \leq B.$$

In order to prove the second inequality in the statement of the lemma, it will be convenient to perform the analysis in terms of a variant of our problem which is explicitly defined with respect to the separation cost of a sequence of tests. We call this new problem the *Pair Separation Problem* (PSP): The input to the PSP, as in the DFEP, is a 5-uple $(S, C, \mathcal{X}, \mathbf{p}, \mathbf{c})$, where $S = \{s_1, \ldots, s_n\}$ is a set of objects, $C = \{C_1, \ldots, C_m\}$ is a partition of S into m classes, $\mathcal{X}$ is a family of subsets of S, $\mathbf{p}$ is a probability distribution on S, and $\mathbf{c}$ is a cost function assigning to each $X \in \mathcal{X}$ a cost $c(X) \in \mathbb{Q}^+$. The only difference between the input of these problems is that the set of tests T in the input of DFEP is replaced with a family $\mathcal{X}$ of subsets of S. We say that $X \in \mathcal{X}$ covers an object s iff $s \in X$. Moreover, we say that $X \in \mathcal{X}$ covers a pair of objects (s, s') if at least one of the conditions hold: (i) $s \in X$ or (ii) $s' \in X$. We say that a pair (s, s') is covered by a sequence of tests if some test in the sequence covers (s, s'). The separation cost of a sequence $\mathbf{X} = X_1 X_2 \ldots X_q$ in the instance I_P of PSP is given by:

$$sepcost(I, \mathbf{X}) = \sum_{i=1}^{q} p\left(X_i \setminus \bigcup_{j=1}^{i-1} X_j\right)\left(\sum_{j=1}^{i} c(X_j)\right) + p\left(S - \bigcup_{j=1}^{q} X_j\right)\sum_{j=1}^{q} c(X_j). \quad (19)$$

The *Pair Separation Problem* consists of finding a sequence of subsets of $\mathcal{X}$ with minimum separation cost, $sepcost^*(I_P)$, among those sequences that cover all pairs in S.

An instance $I = (S, C, T, \mathbf{p}, \mathbf{c})$ of the DFEP induces an instance $I_P = (S, C, \mathcal{X}, \mathbf{p}, \mathbf{c})$ of the PSP where $|T| = |\mathcal{X}|$ and for every test $t \in T$ we have a corresponding subset $X(t) \in \mathcal{X}$ such that $X(t) = \sigma_S(t)$. Thus, in our discussion we will use the term test X to refer to a subset $X \in \mathcal{X}$. In the body of this paper we implicitly work with the instance of the PSP induced by the input instance of the DFEP. It is easy to realize that $sepcost^*(I) = sepcost^*(I_P)$. In addition, $sepcost_B(I, \mathbf{t}^A) = sepcost_B(I_P, \mathbf{X}^A)$, where $\mathbf{X}^A$ is the sequence obtained from $\mathbf{t}^A$ when every $t \in \mathbf{t}^A$ is replaced with $X(t)$. Thus, in order to establish the lemma it suffices to prove that

$$sepcost_B(I_P, \mathbf{X}^A) \leq \gamma \cdot sepcost^*(I_P).$$

It is useful to observe that $\mathbf{X}^A$ is equal to the sequence *seq* returned by procedure `GreedyPSP` in Algorithm 3 when it is executed on the instance (I_P, B). This algorithm corresponds to lines 4,5,9 and 10 of the **While** loop of Algorithm 2. In Algorithm 3, the greedy criteria consists of choosing the test X that maximizes the ratio $p(U \cap X)/c(X)$. This is equivalent to the maximization of $(p(U) - p(U \cap S_t^*))/c(t) = (p(U \cap \sigma_S(t))/c(t)$ defining the greedy choice in Algorithm 2.

The proof consists of the following steps:

   i We construct an instance $I' = (S', C', \mathcal{X}', \mathbf{p}', \mathbf{c}')$ of the PSP from I_P

Algorithm 3

Procedure `GreedyPSP` ($I_P = (S, C, \mathcal{X}, \mathbf{p})$: instance of PSP, B:Budget))

1: $seq \leftarrow \emptyset, \quad U \leftarrow S, \; k \leftarrow 1$
2: **while** there is a test in $\mathcal{X}$ of cost $\leq B$ **do**
3:     let X_k be a test which maximizes $\frac{p(U \cap X)}{c(X)}$ among all tests $X \in \mathcal{X}$ s.t. $c(X) \leq B$ (*)
4:     Append X_k to $seq, \quad U \leftarrow U \setminus X_k, \; B \leftarrow B - c(X_k) \;, \mathcal{X} \leftarrow \mathcal{X} \setminus \{X_k\}, \quad k \leftarrow k + 1$
5: **end while**

---

  ii We prove that the optimal separation cost for I' is no larger than the optimal one for I_P, that is, $sepcost^*(I') \leq sepcost^*(I_P)$.

  iii we prove that separation cost $sepcost(I', \mathbf{X}')$ of any sequence of tests $\mathbf{X}'$ returned by the above pseudo-code on the instance (I', B) is at a constant factor of $sepcost^*(I')$, that is, $sepcost(I', \mathbf{X}')$ is $O(sepcost^*(I'))$.

  iv we prove that there exists a sequence of tests $\mathbf{Z}$ possibly returned by `GreedyPSP` when executed on the instance (I', B) such that $sepcost_B(I_P, \mathbf{X}^A) \leq 2sepcost(I', \mathbf{Z})$.

By chaining these inequalities, we conclude that $sepcost_B(I_P, \mathbf{X}^A)$ is $O(sepcost^*(I_P))$. The steps (ii), (iii) and (iv) are proved in Claims 1,2 and 3, respectively. We start with the construction of instance I'.

Construction of instance I'. For every test $X \in \mathcal{X}$, we define $n(X) = 2c(X)$.

The instance $I' = (S', C', \mathcal{X}', \mathbf{p}', \mathbf{c}')$ is constructed from $I_P = (S, C, \mathcal{X}, \mathbf{p}, \mathbf{c})$ as follows. Let $N = \prod_{X \in \mathcal{X}} n(X)$. For each $s \in S$ we add N objects to S', each of them with probability $p(s)/N$ and with class equal to that of s. If an object s' is added to set S' due to s, we say that s' is generated from s.

For every test $X \in \mathcal{X}$ we add $n(X)$ tests to the set $\mathcal{X}'$, each of them with cost $1/2$. If a test X' is added to set $\mathcal{X}'$ due to X, we say that X' is generated from X.

It remains to define to which subset of S' each test $X' \in \mathcal{X}'$ corresponds to. If $s \notin X$ then $s' \notin X'$ for every s' generated from s and every X' generated from X. Let $\mathcal{X}_s = \{X^1, \ldots, X^{|\mathcal{X}_s|}\}$ be the set of tests that contains the object $s \in S$. Note that the number of tuples $(\theta^1, \ldots, \theta^{|\mathcal{X}_s|})$, where $\theta^i \in \mathcal{X}'$ is a test generated from $X^i \in \mathcal{X}_s$ is $\prod_{X \in \mathcal{X}_s} n(X)$. Thus, we create a one to one correspondence between these tuples and the numbers in the set $Poss(s) = \{1, \ldots, \prod_{X \in \mathcal{X}_s} n(X)\}$. For a test $\theta \in \mathcal{X}'$, generated from $X \in \mathcal{X}_s$, let $F(\theta) \subset Poss(s)$ be the set of numbers that correspond to the tuples that includes θ. Note that

$$|F(\theta)| = \left(\prod_{Y \in \mathcal{X}_s} n(Y) \right) / n(X). \tag{20}$$

In addition, we associate each object $s' \in S'$, generated from s, with a number $f(s') \in Poss(s)$ in a balanced way so that each number in $Poss(s)$ is associated with $N / \prod_{X \in \mathcal{X}_s} n(X)$ objects. Thus, a test $\theta \in \mathcal{X}'$, generated from $X \in \mathcal{X}_s$, covers an object s' generated from s if and only if $f(s') \in F(\theta)$.

For the instance I' we have the following useful properties:

  a if $X \in \mathcal{X}$ covers object $s \in S$ then each test $\theta \in \mathcal{X}'$, generated from X, covers exactly $N/n(X)$ objects generated from s. Moreover, each object generated from s is covered by exactly one test generated from X.

b If a set of tests $G' \subseteq \mathcal{X}'$ covers all pairs of I' then the set $G = \{X \in \mathcal{X}|$ all tests generated from X belong to $G'\}$ covers all pairs of I_P.

Property (a) holds because a test θ generated by X is associated with $|F(\theta)| = \prod_{Y \in \mathcal{X}_s} n(Y)/n(X)$ numbers in $Poss(s)$ and to each number in $Poss(s)$ we have $N/\prod_{Y \in \mathcal{X}_s} n(Y)$ objects associated with.

To see that property (b) holds, let us assume that G' covers all pairs of the instance I' and G does not cover a pair (s_1, s_2). Let $\mathcal{X}_{s_1} = \{X_1^1, \ldots, X_1^x\}$ and $\mathcal{X}_{s_2} = \{X_2^1, \ldots, X_2^y\}$ be the set of tests that covers s_1 and s_2, respectively. The fact that G does not cover (s_1, s_2) implies that $(\mathcal{X}_{s_1} \cup \mathcal{X}_{s_2}) \cap G = \emptyset$ so that for each $X_1^i \in \mathcal{X}_{s_1}$, there is a test θ_1^i, generated from X_1^i, that does not belong to G'. Similarly, for each $X_2^i \in \mathcal{X}_{s_2}$, there is a test θ_2^i, generated from X_2^i, that does not belong to G'. Let s_1' be an object, generated from s_1, that is mapped, via function $f(\cdot)$, into the number in $Poss(s_1)$ that corresponds to the tuple $(\theta_1^1, \ldots, \theta_1^x)$. Moreover, let s_2' be an object, generated from s_2, that is mapped, via function $f(\cdot)$, into the number in $Poss(s_2)$ that corresponds to the tuple $(\theta_2^1, \ldots, \theta_2^y)$. The pair (s_1', s_2') is not covered by G', which is a contradiction.

Claim 1. The optimal separation cost for I' is no larger than the optimal separation cost for I_P, i.e., $sepcost^*(I') \leq sepcost^*(I_P)$.

Given a sequence $\mathbf{X}_P$ for I_P that covers all $P(S)$ pairs we can obtain a sequence $\mathbf{X}$ for I' by replacing each test $X_P \in \mathbf{X}_P$ with the $n(X_P) = 2c(X_P)$ tests in $\mathcal{X}'$ that were generated from X_P, each of which has cost $1/2$. It is easy to see that $\mathbf{X}$ covers all the pairs in I' and the separation cost of $\mathbf{X}$ is not larger than that of $\mathbf{X}_P$. This establishes our claim.

Now let $\mathbf{X}'$ be a sequence of tests returned by procedure `GreedyPSP` in Algorithm 3 when it is executed on the instance (I', B).

Claim 2. The separation cost of the sequence $\mathbf{X}'$ is at most a constant factor of that of $\mathbf{X}^* = X_1^*, \ldots, X_{q^*}^*$, which is the sequence of tests with minimum separation cost among all sequences of tests covering all the pairs, for the instance I', i.e., $sepcost(I', \mathbf{X}') \leq \beta sepcost^*(I')$, for some constant β.

Let p_j (resp. p_j^*) be the sum of the probabilities of the objects covered by the first j tests in $\mathbf{X}'$ (resp. $\mathbf{X}^*$). In particular, we have $p_0 = p_0^* = 0$. In addition, let Q be the sum of the probabilities of all objects in S'. Notice that, with the above notation, we can rewrite the separation cost of the sequence $\mathbf{X}' = X_1', \ldots, X_q'$ as

$$sepcost(I', \mathbf{X}') = \sum_{j=1}^{q} c(X_j')(Q - p_{j-1}) = \sum_{j=1}^{q} (1/2) \cdot (Q - p_{j-1}).$$

Let ℓ be such that $2^{\ell-1} \leq B \leq 2^\ell - 1$, where B is the budget in the statement of the lemma. For $j = 0, \ldots, \ell$, let $i_j = 2^{j+2} - 2$ and $i_j^* = 2^{j+1}$. Furthermore, let $P^{[j]}$ be the sum of the probabilities of the objects covered by the first i_j tests of $\mathbf{X}'$. In formulae, $P^{[j]} = p\left(\bigcup_{k=1}^{i_j} X_k'\right)$. Analogously, let $P_*^{[j]}$ be the sum of the probabilities of the objects covered by the first i_j^* tests in $\mathbf{X}^*$. In formulae, $P_*^{[j]} = p\left(\bigcup_{k=1}^{i_j^*} X_k^*\right)$. For the sake of definiteness, we set $i_{-1} = i_{-1}^* = 0$ and $P^{[-1]} = P_*^{[-1]} = 0$

Then, we have

$$
\begin{aligned}
sepcost(I', \mathbf{X}') \;=\; & \sum_{i=1}^{q}(1/2)\cdot(Q-p_{i-1}) \le \sum_{j=0}^{\ell-1}\sum_{k=i_{j-1}+1}^{i_j}(1/2)\cdot(Q-p_{k-1}) \\
\le\; & \sum_{j=0}^{\ell-1}\sum_{k=i_{j-1}+1}^{i_j}(1/2)(Q-P^{[j-1]}) \le \sum_{j=0}^{\ell-1}2^j(Q-P^{[j-1]}) \\
\le\; & Q+\sum_{j=1}^{\ell-1}2^j(Q-P^{[j-1]}) \le Q+\sum_{j=0}^{\ell-2}2^{j+1}(Q-P^{[j]}),
\end{aligned}
$$

where the first inequality holds because $q \le 2B \le 2^{\ell+1}-2 = i_{\ell-1}$ and the second one holds because $p_{k-1} \ge P^{[j-1]} = p_{i_{j-1}}$ for $k \ge i_{j-1}+1$.

We now devise a lower bound on the separation cost of $\mathbf{X}^*$. For this, we first note that the length q^* of $\mathbf{X}^*$ is at least $2B \ge 2^\ell = i^*_{\ell-1}$, for otherwise the property (b) of instance I' would guarantee the existence of a sequence of tests of total cost smaller than B that covers all pairs for instance I_P (and for the instance I of the DFEP as well), which contradicts Lemma 1. Therefore, we can lower bound the the separation cost of the sequence $\mathbf{X}^*$ as follows:

$$
sepcost(I',\mathbf{X}^*) \;=\; \sum_{i=1}^{q^*} c(X^*_i)(Q-p^*_{i-1}) \tag{21}
$$

$$
\ge\; \sum_{i=1}^{i^*_0}(1/2)\cdot(Q-p^*_{i-1}) + \sum_{j=1}^{\ell-1}\sum_{k=i^*_{j-1}+1}^{i^*_j}(1/2)\cdot(Q-p^*_{k-1}) \tag{22}
$$

$$
\ge\; \frac{2Q-p^*_1}{2} + \sum_{j=1}^{\ell-1}(2^j-2^{j-1})(Q-P^{[j]}_*) \ge \frac{3Q}{4}+\frac12\sum_{j=1}^{\ell-1}2^j(Q-P^{[j]}_*) \tag{23}
$$

The inequality in (22) follows from (21) by considering in the summation on the right hand side of (22) only the first $i^*_{\ell-1} = 2^\ell \le B \le q^*$ tests.

The term $(2Q-p^*_1)/2$ in the first inequality (23) is the contribution of the first two tests of the sequence $\mathbf{X}^*$ to the separation cost. To prove that $\frac{2Q-p^*_1}{2} \ge \frac{3Q}{4}$ in the last inequality, we note that that $p^*_1 \le Q/2$ because the probability covered by the first test X^*_1 of sequence $\mathbf{X}^*$ is $p(X)/n(X) \le Q/2c(X) \le Q/2$, where X is the test that generates X^*_1. In the last inequality we used the fact that $c(X) \ge 1$ for all $X \in \mathcal{X}$.

Let $S'_k \subseteq S'$ be the set of objects covered by the sequence of tests $X'_1, X'_2, \ldots, X'_k$, which is the prefix of length k of the sequence of tests $\mathbf{X}'$. We shall note that for $l \ge k+1$, the subsequence $X'_{k+1}, \ldots, X'_l$ of $\mathbf{X}'$ coincides with the sequence of tests constructed through the execution of `Adapted-Greedy` over the instance $(S' \setminus S'_k, \tilde{T}, f_2, \mathbf{c}', B')$, where

- $\tilde{T} = \mathcal{X}' \setminus \{X'_1, X'_2, \ldots, X'_k\}$ is a set of tests, all of them with cost $1/2$;

- the function f_2 maps a set of tests into the probability of the objects in $S' \setminus S'_k$ that are covered by the tests in the set;

- $B' = \frac{(l-k)}{2}$.

Since the set $\{X_1^*, X_2^*, \ldots, X_{l-k}^*\} - \{X_1', X_2', \ldots, X_k'\}$ is a feasible solution for this instance, it follows from Theorem 2 that $p_l - p_k \geq \hat{\alpha}(p_{l-k}^* - p_k)$, where $\hat{\alpha} = 1 - \frac{1}{e}$. By setting $l = i_j$ and $k = i_{j-1}$ we get that

$$P^{[j]} - P^{[j-1]} \geq \hat{\alpha}(P_*^{[j-1]} - P^{[j-1]}).$$

It follows that

$$Q - P^{[j]} \leq \hat{\alpha}(Q - P_*^{[j-1]}) + (1 - \hat{\alpha})(Q - P^{[j-1]}).$$

Thus, setting

$$U = Q + \sum_{j=0}^{\ell-2} 2^{j+1}(Q - P^{[j]}),$$

which is the upper bound we derived on the separation cost of the sequence $\mathbf{X}'$, we have

$$
\begin{aligned}
U \;=\;& Q + \sum_{j=0}^{\ell-2} 2^{j+1}(Q - P^{[j]}) \\[4pt]
\leq\;& Q + \hat{\alpha}\sum_{j=0}^{\ell-2} 2^{j+1}(Q - P_*^{[j-1]}) + (1 - \hat{\alpha})\sum_{j=0}^{\ell-2} 2^{j+1}(Q - P^{[j-1]}) \\[4pt]
=\;& Q + 2\hat{\alpha}Q + \hat{\alpha}\sum_{j=1}^{\ell-2} 2^{j+1}(Q - P_*^{[j-1]}) + 2(1 - \hat{\alpha})Q + (1 - \hat{\alpha})\sum_{j=1}^{\ell-2} 2^{j+1}(Q - P^{[j-1]}) \\[4pt]
=\;& Q + 2Q + 2\hat{\alpha}\sum_{j=0}^{\ell-3} 2^{j+1}(Q - P_*^{[j]}) + 2(1 - \hat{\alpha})\sum_{j=0}^{\ell-3} 2^{j+1}(Q - P^{[j]}) \\[4pt]
\leq\;& Q + 2Q + 4\hat{\alpha}Q + 2\hat{\alpha}\sum_{j=1}^{\ell-3} 2^{j+1}(Q - P_*^{[j]}) + 2(1 - \hat{\alpha})\sum_{j=0}^{\ell-3} 2^{j+1}(Q - P^{[j]}) \\[4pt]
=\;& (1 - 2(1 - \hat{\alpha}) + 2 + 4\hat{\alpha})Q + 4\hat{\alpha}\sum_{j=1}^{\ell-3} 2^{j}(Q - P_*^{[j]}) + 2(1 - \hat{\alpha})\left(Q + \sum_{j=0}^{\ell-3} 2^{j+1}(Q - P^{[j]})\right) \\[4pt]
\leq\;& (1 + 6\hat{\alpha})Q + 4\hat{\alpha}\sum_{j=1}^{\ell-1} 2^{j}(Q - P_*^{[j]}) + 2(1 - \hat{\alpha})U \\[4pt]
\leq\;& (8\hat{\alpha} + 4/3)\, sepcost(I', \mathbf{X}^*) + 2(1 - \hat{\alpha})U,
\end{aligned}
$$

where the last inequality follows from equation 23. Thus, we obtain

$$sepcost(I', \mathbf{X}') \leq U \leq \frac{(8\hat{\alpha} + 4/3)}{2\hat{\alpha} - 1} sepcost(I', \mathbf{X}^*).$$

For the last claim let $\mathbf{X}^A$ be the sequence obtained by `GreedyPSP` (Algorithm 3) when it is executed on instance (I_P, B).

Claim 3. There exists an execution of procedure `GreedyPSP` (Algorithm 3) on instance (I', B) which returns a sequence $\mathbf{Z}$ satisfying $sepcost_B(I_P, \mathbf{X}^A) \leq 2sepcost(I', \mathbf{Z})$.

Let X_i^A be the i-th test of sequence $\mathbf{X}^A$ and let X_{r+1}^A be the first test of $\mathbf{X^A}$ that is not the test which maximizes $p(U \cap X)/c(X)$ among all the tests in $\mathcal{X}$ in line (*) of Algorithm 3. Note

that X_{r+1}^A is chosen by Algorithm 3 rather than Y, the test which maximizes $p(U \cap X)/c(X)$, because Y has cost larger than remaining budget $z = B - \sum_{j=1}^{r} c(X_j^A)$. The case where X_{r+1}^A does not exist is easier to handle and will be discussed at the end of the proof. Because $X_1^A, \ldots, X_r^A$ is a prefix of $\mathbf{X}^A$ we have

$$sepcost_B(I_P, \mathbf{X}^A) \le sepcost_B(I_P, \langle X_1^A, \ldots, X_r^A \rangle).$$

Thus, to establish the claim it suffices to show that

$$sepcost_B(I_P, \langle X_1^A, \ldots, X_r^A \rangle) \le 2sepcost(I', \mathbf{Z}),$$

where $\mathbf{Z}$ is a possible output of `GreedyPSP` (Algorithm 3) on instance (I', B).

For $j = 1, \ldots, r$, let $\mathbf{Z}^{[j]} = \langle Z_1^{[j]}, \ldots, Z_{n(X_j^A)}^{[j]} \rangle$ be a sequence of tests defined by some permutation of the $n(X_j^A)$ tests in $\mathcal{X}'$, generated from X_j^A.

Let $z = B - \sum_{j=1}^{r} c(X_j^A)$ and $\mathbf{Z}^{[r+1]} = \langle Z_1^{[r+1]}, \ldots, Z_{2z}^{[r+1]} \rangle$ be a sequence of $2z$ of the $n(Y) = 2c(Y) > 2z$ tests in $\mathcal{X}'$, generated from Y. The proof of the following proposition is deferred to section B.

Proposition 3. *Let $\mathbf{Z} = \mathbf{Z}^{[1]} \mathbf{Z}^{[2]} \ldots \mathbf{Z}^{[r]} \mathbf{Z}^{[r+1]}$ be the sequence obtained by the juxtaposition of the sequences $\mathbf{Z}^{[1]}, \ldots, \mathbf{Z}^{[r+1]}$. Then, for $\mathbf{Z}$ the following conditions hold:*

(i) for each $j = 1, \ldots, r+1$ and $\kappa \ne \kappa' \in \{1, \ldots, n(X_j^A) = 2c(X_j^A)\}$, it holds that

$$Z_\kappa^{[j]} \cap Z_{\kappa'}^{[j]} = \emptyset$$

(ii) For each $j = 1, \ldots, r+1$ and each test Q in $\mathcal{X}'$, with X being the test in $\mathcal{X}$ from which Q is generated, it holds that

$$\frac{p(Q - \bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]})}{c(Q)} = \frac{p(X - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)}$$

(iii) $\mathbf{Z}$ is a feasible output for `GreedyPSP` (Algorithm 3) on instance (I', B).

First note that the sequence $\mathbf{Z}$ has length $2B$ and total cost B. This is easily verified by recalling that: (i) each test in the sequence $\mathbf{Z}$ has cost $1/2$; (ii) for each $j = 1, \ldots, r$, the subsequence $\mathbf{Z}^{[j]}$ has length $n(X_j^A)$, hence $totcost(I', \mathbf{Z}^{[j]}) = n(X_j^A)/2 = c(X_j^A)$; (iii) the subsequence $\mathbf{Z}^{[r+1]}$ has length $2z = 2B - 2\sum_{j=1}^{r} c(X_j^A) = 2(B - \sum_{j=1}^{r} |\mathbf{Z}^{[j]}|)$ hence $totcost(I', \mathbf{Z}^{[r+1]}) = z = B - \sum_{j=1}^{r} totcost(I', \mathbf{Z}^{[j]})$.

Let $C_0 = 0$, and for $j = 1, \ldots, r$, let $C_j = \sum_{i=1}^{j} c(X_i^A)$. By the observations in the previous paragraph, we also have $C_j = totcost(I', < \mathbf{Z}^{[1]} \ldots \mathbf{Z}^{[j]} >) = \sum_{i=1}^{j} \sum_{\kappa=1}^{n(X_i^A)} c(Z_\kappa^{[i]})$.

By grouping objects which incur the same cost in $\mathbf{X}$, we can write $sepcost_B(I_P, X_1^A, \ldots, X_r^A)$ as follows

$$sepcost_B(I_P, \langle X_1^A, \ldots, X_r^A \rangle) = \sum_{j=1}^{r} C_j \cdot p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right) + B \cdot p\left(S - \bigcup_{j=1}^{r} X_j^A\right) \quad (24)$$

Analogously, we can compute $sepcost(I', \mathbf{Z})$ as follows:

$$
\begin{aligned}
sepcost(I', \mathbf{Z}) \;=\; & \sum_{j=1}^{r} \sum_{\kappa=1}^{n(X_j^A)} p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]} \right) \right) \left(C_{j-1} + \kappa \cdot \frac{1}{2} \right) \\
& + \sum_{\kappa=1}^{2z} p\left(Z_\kappa^{[r+1]} - \left(\bigcup_{i=1}^{r} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[r+1]} \right) \right) \left(C_r + \kappa \cdot \frac{1}{2} \right) \\
& + B \cdot p\left(S' - \left(\bigcup_{i=1}^{r} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) - \left(\bigcup_{\kappa=1}^{2z} Z_\kappa^{[r+1]} \right) \right),
\end{aligned}
\tag{25}
$$

where we have splitted $\mathbf{Z}$ into the objects covered by the subsequences $\mathbf{Z}^{[1]}, \ldots, \mathbf{Z}^{[r]}$, the objects covered by the subsequence $\mathbf{Z}^{[r+1]}$ and the remaining objects. In the above expressions, the term $Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]} \right)$ represents the set of objects covered by $Z_\kappa^{[j]}$ and not covered by any of the preceding tests in $\mathbf{Z}$. The cost of separating each of these objects is the sum of the costs of all the tests performed up to $Z_\kappa^{[j]}$, i.e., $\sum_{i=1}^{j-1} totcost(I', \mathbf{Z}^{[i]}) + \kappa/2 = C_{j-1} + \kappa/2$.

Now, we notice that for each $j = 1, \ldots, r+1$ and $1 \leq \kappa \leq \min\{n(X_j^A), 2z\}$ the set of objects covered by $Z_\kappa^{[j]}$ and not covered by the previous tests are

$$
Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]} \right) = Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right)
$$

where the equality follows because by Proposition 3 (i) we have that $Z_\kappa^{[j]} \cap \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]} \right) = \emptyset$.

Moreover, by Proposition 3 (ii) we have that

$$
p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) \right) = \frac{p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A \right)}{n(X_j^A)}.
$$

Finally, the set

$$
R = S' - \left(\bigcup_{i=1}^{r} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]} \right) - \left(\bigcup_{\kappa=1}^{2z} Z_\kappa^{[r+1]} \right)
$$

that appears in the third term of the righthand side of (25) can be spilt into the objects covered by the tests generated from Y and the remaining ones. By using arguments similar to those employed above one can realize that the objects covered by the tests generated from Y are exactly those generated by the objects in $Y - \bigcup_{i=1}^{r} X_i^A$ that are not covered by the tests in $\left(\bigcup_{\kappa=1}^{2z} Z_\kappa^{[r+1]} \right)$. Thus, their contribution to $p(R)$ is $\left(\frac{n(Y)-2z}{n(Y)} \right) p(Y - \bigcup_{i=1}^{r} X_i^A)$. On the other hand, the contribution to $p(R)$ of the remaining objects is $p(S - Y - \bigcup_{j=1}^{r} X_j^A)$.

Therefore, we can rewrite (25) as follows

$$sepcost(I', \mathbf{Z}) \;=\; \sum_{j=1}^{r} \sum_{\kappa=1}^{n(X_j^A)} \frac{p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right)}{n(X_j^A)} \left(C_{j-1} + \kappa \cdot \frac{1}{2}\right)$$

$$+ \sum_{\kappa=1}^{2z} \frac{p\left(Y - \bigcup_{i=1}^{r} X_i^A\right)}{n(Y)} \left(C_r + \kappa \cdot \frac{1}{2}\right)$$

$$+ \frac{n(Y) - 2z}{n(Y)} p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \cdot B + p\left(S - Y - \bigcup_{j=1}^{r} X_j^A\right) B \quad (26)$$

Via simple algebraic manipulation on the first term in the right hand side of (26) we have that

$$\sum_{\kappa=1}^{n(X_j^A)} \frac{1}{n(X_j^A)} \left(C_{j-1} + \kappa \cdot \frac{1}{2}\right) = C_{j-1} + \frac{n(X_j^A) + 1}{4} \geq C_{j-1} + \frac{c(X_j^A)}{2},$$

and, analogously, for the second term in the right hand side of (26) we have

$$\frac{1}{n(Y)} \sum_{\kappa=1}^{2z} \left(C_r + \kappa \cdot \frac{1}{2}\right) = \frac{2z}{n(Y)} \left(C_r + \frac{2z + 1}{4}\right) \geq \frac{2z}{n(Y)} \frac{B + C_r}{2}.$$

Hence, we have

$$sepcost(I', \mathbf{Z}) \;\geq\; \sum_{j=1}^{r} p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right) \left(C_{j-1} + \frac{c(X_j^A)}{2}\right) + p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \frac{2z}{n(Y)} \frac{B + C_r}{2}$$

$$+ \frac{n(Y) - 2z}{n(Y)} p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \cdot B + p\left(S - Y - \bigcup_{j=1}^{r} X_j^A\right) B \quad (27)$$

Finally, we observe that $C_{j-1} + \frac{c(X_j^A)}{2} \geq C_j/2$ and $(B + C_r)/2 \geq B/2$. Then, the sum of the second and third term in the right hand side of (27) can be lower bounded with $p(Y - \bigcup_{j=1}^{r} X_j^A) \cdot B/2$ and we get

$$sepcost(I', \mathbf{Z}) \geq \sum_{j=1}^{r} p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right) \frac{C_j}{2} + p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \frac{B}{2} + p(S - Y - \bigcup_{j=1}^{r} X_j^A) B \quad (28)$$

Putting together (28) and (24) we have the desired result

$$sepcost(I', \mathbf{Z}) \geq 2 sepcost_B(I_P, < X_1^A, \ldots, X_r^A >).$$

It remains to argue about the case where X_{r+1}^A does not exist, which means that all tests that maximize the greedy criteria in Algorithm 3 have cost smaller than the current budget B. In this case, the analysis becomes simpler and be easily handled in the same way as above. In fact, the only difference is that the last term in (24) disappears, as well as all the terms referring to Y and $\mathbf{Z}^{[r+1]}$.

The lemma follows from the correctness of the three claims. $\qquad\square$

B   The Proof of Proposition 3

Proposition 3. *Let $\mathbf{Z} = \mathbf{Z}^{[1]}\,\mathbf{Z}^{[2]}\,\ldots\,\mathbf{Z}^{[r]}\,\mathbf{Z}^{[r+1]}$ be the sequence obtained by the juxtaposition of the sequences $\mathbf{Z}^{[1]}, \ldots, \mathbf{Z}^{[r+1]}$. Then, for $\mathbf{Z}$ the following conditions hold:*

(i) for each $j = 1, \ldots, r+1$ and $\kappa \neq \kappa' \in \{1, \ldots, n(X_j^A) = 2c(X_j^A)\}$, it holds that

$$Z_\kappa^{[j]} \cap Z_{\kappa'}^{[j]} = \emptyset$$

(ii) For each $j = 1, \ldots, r+1$ and each test Q in $\mathcal{X}'$, with X being the test in $\mathcal{X}$ from which Q is generated, it holds that

$$\frac{p(Q - \bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]})}{c(Q)} = \frac{p(X - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)}$$

(iii) $\mathbf{Z}$ is a feasible output for `GreedyPSP` (Algorithm 3) on instance (I', B).

Proof. Item (i) is a direct consequence of property (a) of the instance I'.

In order to prove (ii), we observe that, from the definition of the sequences $\mathbf{Z}^{[i]}$ ($i = 1, \ldots, r$), it follows that the elements of $W = \bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}$ are all the elements in S' which are generated from $\bigcup_{i=1}^{j-1} X_i^A$. Therefore, the elements of $Q - W$ are precisely the elements of Q which are generated from $X - \bigcup_{i=1}^{j-1} X_i^A$.

For each $s \in X - \bigcup_{i=1}^{j-1} X_i^A$, there are precisely $\frac{N}{n(X)}$ elements in Q that are generated from s, and each one of them has probability $p(s)/N$. Hence we have

$$\frac{p\left(Q - \bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)}{c(Q)} = \frac{1}{c(Q)} \sum_{s \in X - \bigcup_{i=1}^{j-1} X_i^A} \frac{N}{n(X)} \frac{p(s)}{N} = \frac{1}{c(Q)n(X)} \sum_{s \in X - \bigcup_{i=1}^{j-1} X_i^A} p(s),$$

from which we have (ii), since $1/c(Q)n(X) = 2/n(X) = c(X)$.

In order to prove (iii) it is enough to show that the following claim holds.

Claim. For each $j = 1, \ldots, r+1$ and $\kappa = 1, \ldots, \min\{2z, n(X_j^A)\}$ we have that

$$\frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)}{c(Z_\kappa^{[j]})} \geq \frac{p\left(Q - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)}{c(Q)}$$

$$\tag{29}$$

for any $Q \in \mathcal{X}'$.

This claim says that, for each $j = 1, \ldots, r+1$ and $1 \leq \kappa \leq \min\{2z, n(X_j^A)\}$, if $\mathbf{Z}$ has been constructed up to the test preceding $Z_\kappa^{[j]}$ then with respect to the tests already chosen, the test $Z_\kappa^{[j]}$ satisfies the greedy criterium of procedure `GreedyPSP`. This implies that $\mathbf{Z}$ is a feasible output for `GreedyPSP`, as desired.

Proof of the Claim. Let R be the quantity on the right hand side of (29), and X be the test in $\mathcal{X}$ from which Q is generated. Then we have

$$R \leq \frac{p\left(Q - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)}{c(Q)} \tag{30}$$

$$= \frac{p\left(X - \bigcup_{i=1}^{j-1} X_i^A\right)}{c(X)} \tag{31}$$

$$\leq \frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)}{c(Z_\kappa^{[j]})} \tag{32}$$

$$= \frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)}{c(Z_\kappa^{[j]})} \tag{33}$$

Inequality (30) holds since the set whose probability is considered at the numerator of the right hand side of (30) is a superset of the set whose probability is considered at the numerator of the right hand side of (29).

Inequality (31) follows from (30) by property (ii) above.

In order to prove (32) we consider two cases, according to whether $j = r + 1$ or $j < r + 1$. If $j < r + 1$, the first inequality below follows from the greedy choice

$$\frac{p(X - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)} \leq \frac{p(X_j^A - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)} = \frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)}{c(Z_\kappa^{[j]})}$$

and the last equality follows from property (ii) of the proposition under analysis.

If $j = r + 1$ we have that, by definition[2] of Y and the sequence $\mathbf{Z}^{[r+1]}$, it holds that

$$\frac{p(X - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)} \leq \frac{p(Y - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)} = \frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)}{c(Z_\kappa^{[j]})}$$

where the last equality follows from property (ii) above.

Finally, (33) follows from (32) because of property (i) above, from which we have that $Z_\kappa^{[j]} \cap \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right) = \emptyset$ hence,

$p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right) = p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right).$

$\square$

---

[2]Recall that Y is the test in $\mathcal{X}$ which maximizes the greedy criterium, but is not chosen because it exceeds the available budget