Pontifícia Universidade Católica
do Rio de Janeiro

Aline Medeiros Saettler

Approximation Algorithms for Decision Trees

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática.

Advisor: Prof. Eduardo Sany Laber

Rio de Janeiro
September 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Aline Medeiros Saettler

Approximation Algorithms for Decision Trees

Thesis presented to the Programa de Pós-Graduação em Informática, of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor.

Prof. Eduardo Sany Laber
Advisor
Departamento de Informática — PUC-Rio

Prof. Thibaut Victor Gaston Vidal
Departamento de Informática — PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática — PUC-Rio

Prof. Celina Miraglia Herrera de Figueiredo
UFRJ

Prof. Cristina Gomes Fernandes
USP

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies Centro Técnico Científico da
PUC-Rio

Rio de Janeiro, September 5^{th}, 2017

Aline Medeiros Saettler

Bachelor's in Computer Science at the Federal University of Espírito Santo (2011). Masters' in Informatics at the Pontifical Catholic University of Rio de Janeiro (2013), with emphasis in Combinatorial Optimization.

Bibliographic data

Saettler, Aline

Approximation Algorithms for Decision Trees / Aline Medeiros Saettler ; advisor: Eduardo Sany Laber. — 2017.
90 f. : il. ; 30 cm

Tese (Doutorado em Informática)-Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2017.
Inclui bibliografia

1. Informática – Teses. 2. Árvores de Decisão; Algoritmos de Aproximação; Optimização Combinatória.. I. Laber, Eduardo. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

To my family, my friends, my advisor and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

Abstract

Saettler, Aline; Laber, Eduardo (advisor). **Approximation Algorithms for Decision Trees**. Rio de Janeiro, 2017. 90p. D.Sc. Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Decision tree construction is a central problem in several areas of computer science, for example, data base theory and computational learning. This problem can be viewed as the problem of evaluating a discrete function, where to check the value of each variable of the function we have to pay a cost, and the points where the function is defined are associated with a probability distribution. The goal of the problem is to evaluate the function minimizing the cost spent (in the worst case or in expectation). In this Thesis, we present four contributions related to this problem. The first one is an algorithm that achieves an $O(log(n))$ approximation with respect to both the expected and the worst costs. The second one is a procedure that combines two trees, one with worst cost W and another with expected cost E, and produces a tree with worst cost at most $(1+\rho)W$ and expected cost at most $(1/(1-e^{-\rho}))E$, where ρ is a given parameter. We also prove that this is a sharp characterization of the best possible trade-off attainable, showing that there are infinitely many instances for which we cannot obtain a decision tree with both worst cost smaller than $(1+\rho)OPT_W(I)$ and expected cost smaller than $(1/(1-e^{-\rho}))OPT_E(I)$, where $OPT_W(I)$ (resp. $OPT_E(I)$) denotes the cost of the decision tree that minimizes the worst cost (resp. expected cost) for an instance I of the problem. The third contribution is an $O(log(n))$ approximation algorithm for the minimization of the worst cost for a variant of the problem where the cost of reading a variable depends on its value. Our final contribution is a randomized rounding algorithm that, given an instance of the problem (with an additional integer $k \geq 0$) and a parameter $0 < \epsilon < 1/2$, builds an oblivious decision tree with cost at most $(3/(1-2\epsilon))ln(n)OPT(I)$ and produces at most (k/ϵ) errors, where $OPT(I)$ denotes the cost of the oblivious decision tree with minimum cost among all oblivious decision trees for instance I that make at most k classification errors.

Keywords

Decision Trees; Approximation Algorithms; Combinatorial Optimization.

Resumo

Saettler, Aline; Laber, Eduardo. **Algoritmos de Aproximação para Árvores de Decisão**. Rio de Janeiro, 2017. 90p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A construção de árvores de decisão é um problema central em diversas áreas da ciência da computação, por exemplo, teoria de banco de dados e aprendizado computacional. Este problema pode ser visto como o problema de avaliar uma função discreta, onde para verificar o valor de cada variável da função temos que pagar um custo, e os pontos onde a função está definida estão associados a uma distribuição de probabilidade. O objetivo do problema é avaliar a função minimizando o custo gasto (no pior caso ou no caso médio). Nesta tese, apresentamos quatro contribuições relacionadas a esse problema. A primeira é um algoritmo que alcança uma aproximação de $O(\log(n))$ em relação a tanto o custo esperado quanto ao pior custo. A segunda é um método que combina duas árvores, uma com pior custo W e outra com custo esperado E, e produz uma árvore com pior custo de no máximo $(1+\rho)W$ e custo esperado no máximo $(1/(1-e^{-\rho}))E$, onde ρ é um parâmetro dado. Nós também provamos que esta é uma caracterização justa do melhor trade-off alcançável, mostrando que existe um número infinito de instâncias para as quais não podemos obter uma árvore de decisão com tanto o pior custo menor que $(1+\rho)OPT_W(I)$ quanto o custo esperado menor que $(1/(1-e^{-\rho}))OPT_E(I)$, onde $OPT_W(I)$ (resp. $OPT_E(I)$) denota o pior custo da árvore de decisão que minimiza o pior custo (resp. custo esperado) para uma instância I do problema. A terceira contribuição é um algoritmo de aproximação de $O(\log(n))$ para a minimização do pior custo para uma variante do problema onde o custo de ler uma variável depende do seu valor. Nossa última contribuição é um algoritmo *randomized rounding* que, dada uma instância do problema (com um inteiro adicional $k \geq 0$) e um parâmetro $0 < \epsilon < 1/2$, produz uma árvore de decisão *oblivious* com custo no máximo $(3/(1-2\epsilon))ln(n)OPT(I)$ e que produz no máximo (k/ϵ) erros, onde $OPT(I)$ denota o custo da árvore de decisão oblivious com o menor custo entre todas as árvores *oblivious* para a instância I que produzem no máximo k erros de classificação.

Palavras-chave

Árvores de Decisão; Algoritmos de Aproximação; Optimização Combinatória.

Contents

List of Figures

List of Tables

1
Introduction

Decision tree construction is a central problem in several areas of computer science, e.g., in data base theory, in computational learning and in artificial intelligence (Sattler & Dunemann (2001)). In a typical scenario, there are several possible hypotheses, which can explain some unknown phenomenon, and we want to decide which of them provides the correct explanation. We have a prior distribution on the hypotheses and we can use tests to discriminate among the hypotheses. Each test's outcome eliminates some of them, and by using a sequence of tests we can significantly reduce the space of hypothesis. Moreover, different tests may have different associated costs. The aim is to define the best testing strategy that reaches the correct decision while spending as little as possible. A strategy is represented by a tree (called decision tree) with each node being a test and each leaf being a hypothesis. In a generalization of this scenario, one is only interested in identifying a class of possible hypothesis explaining the situation. We next give some motivating examples in order to clarify these concepts.

In a typical scenario of automatic medical diagnosis, a person has an unknown disease and there is a set of possible diagnoses that can explain his symptoms. In order to find the correct one, a doctor have to ask for a series of exams. The result of an exam, however, can eliminate the need for further ones. Thus, it makes sense to look for a sequence of exams trying to minimize the cost paid to identify the disease. Another option is to look for a sequence that minimizes the number of exams taken by the person. Figure 1.1 illustrates this situation. Finally, a general variant of this problem asks to identify the therapy rather than the diagnosis. For example, in case of poisoning it is important to quickly understand which antidote to administer rather than identifying the exact poisoning.

In high frequency trading, an automatic agent decides the next action to be performed, such 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). The tests correspond to these variables, and a combination of values to the variables represents a possible scenario of the market. Finally, for

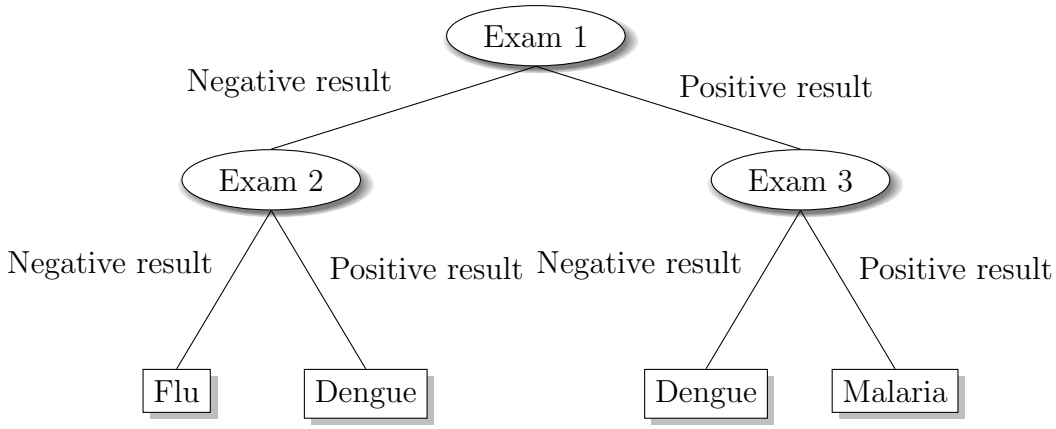

Figure 1.1: A person has to take several exams to get a diagnosis. In the above example, the first exam realized is Exam 1. If its result is negative, then the person has to take Exam 2. However, if the result is positive, he/she has to take Exam 3 to get a diagnosis.

each scenario, there is an associated action to be taken (e.g. buy or sell an item). Every time an action needs to be taken, the agent can identify the scenario by computing the value of each single variable and proceeding with the associated action. However, recomputing all the variables 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 time consuming (expensive) 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.

The above examples can all be cast into one general problem, which we define in the next section.

1.1 Notation and Problem Definition

The Discrete Function Evaluation Problem. 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. A test $t_i \in T$, when applied to an object $s \in S$, outputs a number $t_i(s)$ in the set $\{1, \ldots, \ell\}$. The cost function $\mathbf{c}$ assigns to each pair (test t_i, object s) a cost $c^{t_i(s)}(t) \in \mathbb{Q}^+$.

This definition will be used through the rest of this Thesis, with the exception of Chapter 7. In order to keep the notation simpler, we use $c(t)$

instead of $c^{t(s)}(t)$ whenever the cost is independent of the outcome of the test (i.e., when $c^1(t) = \ldots = c^\ell(t)$).

The $DFEP$, as its name indicates, is often stated in terms of minimizing the cost of evaluating a discrete function, which maps points (objects in S) into function values (classes in C), where the i^{th} coordinate of a point $s \in S$ corresponds to the value $t^i(s)$ output by t when applied to s.

We assume that the set of tests is complete, that is, for every s_i and s_j belonging to different classes, there is at least one test in T that outputs different values for s_i and s_j.

A *decision tree* is a tree where every internal node is associated with a test and every leaf node is associated with a class. The branches leaving an internal node are associated with the possible outcomes of the test associated with the node. More formally, a decision tree D over a set of objects S is either:

- A leaf node associated with some class C_i
- A graph composed by a root node r associated with some test t, and edges from r to decision trees $\{D_1, \ldots, D_\ell\}$, where D_i is defined over the (non-empty) subset of objects in S for which t outputs i.

To classify an object $s \in S$ according to a decision tree D, we start at the root r of D, apply the test t associated with r on s, observe the output $t(s)$ and follow the branch associated with it, paying a cost of $c^{t(s)}(t)$. We repeat this step until we reach a leaf node associated with some class C_i. If $s \in C_j$ and $j \neq i$, s is misclassified and we have a classification error. All problems studied in this Thesis are related to the construction of decision trees that correctly classify all objects in S, with the only exception being the problem studied in Chapter 7. We will assume that this statement is always true, i. e., by a *decision tree* we mean a decision tree *that correctly classify all objects in* S, unless the contrary is explicitly stated.

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

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

$$cost_E(D) = \sum_{s \in S} cost(D, s)p(s) \tag{1-2}$$

Figure 1.2 shows a decision tree D for objects in Table 1.1. Consider that $c(t_1) = 3$, $c(t_2) = 4$, $c^1(t_3) = 5$ and $c^2(t_3) = 6$. We have that:

- $Cost(D, s_1) = c(t_2) = 4$

Object	t_1	t_2	t_3	Class	Probability
s_1	1	1	2	C_1	0.1
s_2	1	2	1	C_1	0.2
s_3	2	2	1	C_2	0.4
s_4	1	2	2	C_3	0.25
s_5	2	2	2	C_3	0.05

Table 1.1: A set S of five objects and their respective classes, probabilities and outputs for each test.

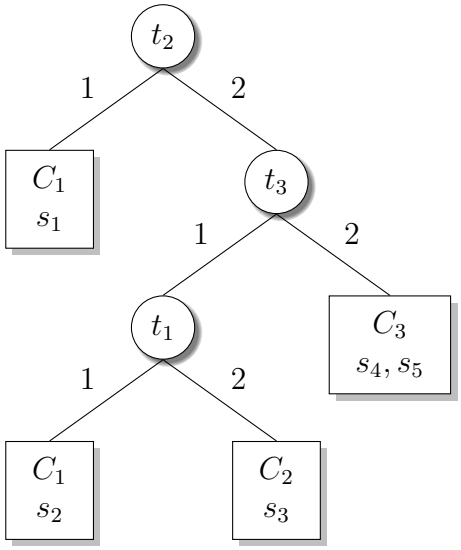

Figure 1.2: Decision Tree for objects of Table 1.1. Circles indicate internal nodes and squares indicate leaf nodes.

- $Cost(D, s_2) = c(t_2) + c^1(t_3) + c(t_1) = 4 + 5 + 3 = 12$
- $Cost(D, s_3) = c(t_2) + c^1(t_3) + c(t_1) = 4 + 5 + 3 = 12$
- $Cost(D, s_4) = c(t_2) + c^2(t_3) = 4 + 6 = 10$
- $Cost(D, s_5) = c(t_2) + c^2(t_3) = 4 + 6 = 10$
- $Cost_W(D) = \max\{4, 12, 10\} = 10$
- $Cost_E(D) = 4 \times 0.1 + 12 \times 0.2 + 12 \times 0.4 + 10 \times 0.05 + 10 \times 0.25 = 10.6$

We 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 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.

An *oblivious decision tree* is a decision tree where nodes at the same level are associated with the same test. Figure 1.3 shows an oblivious decision tree for objects in Table 1.2. Note that the decision tree in Figure 1.2 is also an oblivious decision tree, since there is only one test in each level.

Object	t_1	t_2	Class	Probability
s_1	1	1	C_1	0.25
s_2	1	2	C_2	0.25
s_3	2	1	C_3	0.3
s_4	2	2	C_4	0.2

Table 1.2: A set S of four objects.

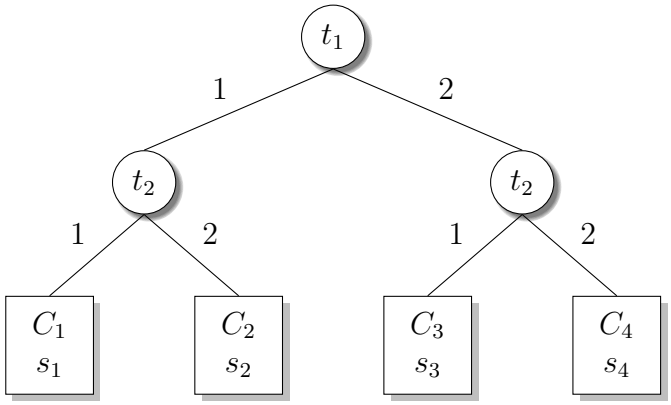

Figure 1.3: Oblivious Decision Tree for objects of Table 1.2

.

Given $G \subseteq S$, 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. Then, we have

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

where for $1 \leq i \leq m$ and $G \subseteq S$, $n_i(G)$ denotes the number of objects in G belonging to class C_i. This is a key concept and will be used in Chapters 3, 5 and 6.

Finally, we briefly present the definition of *approximation algorithms*. Consider a minimization problem where each feasible solution has a nonnegative cost. We say that an algorithm has an approximation $\rho(n)$ for a minimization problem if, for any input of size n, the cost C of the solution produced by the algorithm is within a factor of $\rho(n)$ of the cost C^* of an optimal solution, i. e., $C/C^* \leq \rho(n)$. If an algorithm achieves a $\rho(n)-$approximation, we call it a $\rho(n)$-approximation algorithm (Cormen (2009)).

Throughout this Thesis, we will define several different goals based of the above definitions (for example, to find an approximation algorithm that minimizes simultaneously both the expected and the worst testing costs).

1.2 Research Questions

We now describe the research questions studied in this Thesis. We are interested in answering the following questions:

1. Is it possible to build a polynomial time algorithm that produces a decision tree with a good approximation for both the expected testing cost and worst testing costs? In the positive case, what is the best possible approximation we can get when minimizing these two goals?
2. There exists in general a decision tree with worst testing cost and expected testing cost arbitrarily close, respectively, to the optimal worst testing cost and the optimal expected testing cost? Or, otherwise, what is the threshold for the best trade-off we can hope for? Note that, unlike the first question, in this question we are not concerned with a polynomial time construction.
3. If the cost of each test is not fixed, and depends also on its answers, what is the best possible approximation we can get?
4. Given an integer k, we are interested in finding an oblivious decision tree with minimum cost among all oblivious decision trees that make at most k classification errors. What is the best possible approximation we can get for this problem?

In problems 1, 2 and 4 we assume that $c^1(t) = \ldots = c^\ell(t)$, and in problems 1, 2, and 3 we are interested only in decision trees that correctly classify all objects.

The first two questions we study in this Thesis are regarding to the minimization of two different optimization criteria, which can lead to very different trees. 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 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, 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 is also not large (compared with the optimal worst testing cost).

To illustrate how the goals may differ, let us consider the problem of constructing a prefix code for an alphabet with n symbols (Cormen (2009)). This is a particular case of the decision tree optimization problem given above, where each symbol corresponds to an object, each one of the n symbols (objects) belongs to a different class, the testing costs are uniform and the set of tests is in one to one correspondence with the set of all binary strings of length n. In particular, a test outputs 0 (resp. 1) for an object s_i if the ith bit of the binary string associated to the test is 0 (resp. 1).

Let us consider the case where the probability distribution on the objects is given by $p(s_i) = 2^{-i}$ for each $i = 1, \ldots, n-1$ and $p(s_n) = 2^{-(n-1)}$. Table 1.3 shows an example for $n = 5$. Let D_E^* and D_W^* be decision trees with, respectively, minimum expected cost and minimum worst testing cost for the instance. D_E^* can be constructed by the Huffman's algorithm, and it is not difficult to verify that we have $cost_E(D_E^*) \leq 3$ and $cost_W(D_E^*) = n-1$ (Figure 1.4 shows this construction for the example in Table 1.1). In addition, one possibility for D_W^* is the decision tree that implements a binary search and, in this case, we have that $cost_E(D_W^*) = cost_W(D_W^*) = \Theta(\log n)$. Therefore, we have here an example where the minimization of the expected testing cost produces a decision tree whose worst testing cost is exponentially worse than the cost of the worst-cost-optimal tree and vice versa the minimization of the worst testing cost produces a decision tree whose expected testing cost is $\Theta(\log n)$ larger than the expected testing cost of the decision tree that minimizes this measure. The choice of the "wrong" optimization criterion might have serious consequences in practical applications.

Table 1.3: Example for $n = 5$.

Object	$\mathbf{t_1}$	$\mathbf{t_2}$	$\mathbf{t_3}$	$\cdots$	$\mathbf{t_{30}}$	$\mathbf{t_{31}}$	$\mathbf{t_{32}}$	**Class**	**Probability**
s_1	0	0	0		1	1	1	C_1	0.5
s_2	0	0	0		1	1	1	C_2	0.25
s_3	0	0	0	$\cdots$	1	1	1	C_3	0.125
s_4	0	0	1		0	1	1	C_4	0.0625
s_5	0	1	0		1	0	1	C_5	0.0625

Our third question arises when we analyze scenarios that are not covered by a common assumption made in decision tree problems: that the cost of the tests is fixed in advance and known to the algorithm. In particular, the cost is independent of the outcome of the test. There are several scenarios in medical applications where this assumption does not apply. Many diagnostic tests actually consist of a multi-stage procedure, e.g., in a first stage the

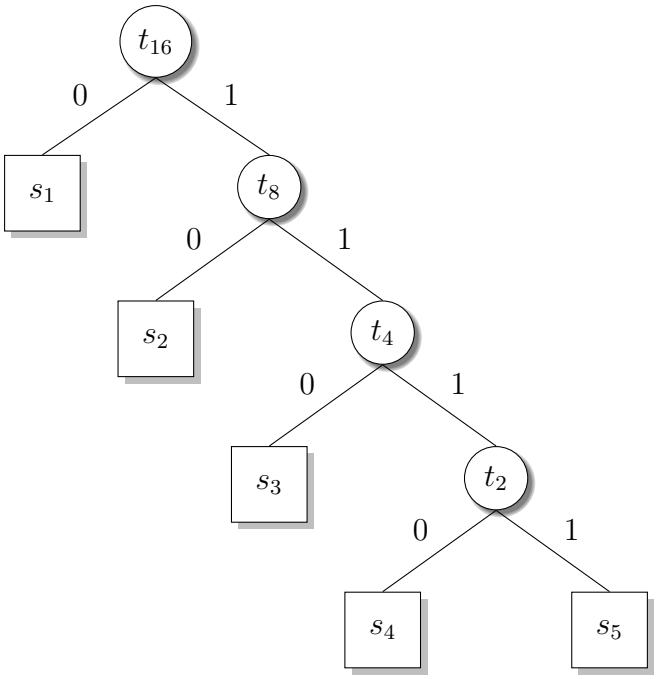

Figure 1.4: Optimal tree D_E^* for objects of Table 1.3, produced by Huffman's algorithm.

sample is tested against some reagent to check for the presence or absence of an antigene. If this appears to be present below a certain level the test is considered to be negative and no further analysis is performed. Otherwise, the test is necessarily followed by a second stage where several new reagents are used with significantly higher final costs. Notice that in such a situation there is no real decision left to the strategy between the first and the second stage, so it is reasonable to consider such a two stage procedure as a single test whose cost depends on the outcome. In particular, we would like to know what is the best approximation we can have for the worst testing cost. We refer to this variant of the $DFEP$ as *value dependent*.

Finally, our last problem is motivated by feature selection, a commonly studied problem in machine learning. The process of selecting subsets of relevant features/attributes of a given dataset may significantly reduce the computational cost of classification algorithms and the chances of data overfitting. The problem of finding a minimal subset of features that is enough to distinguish among all samples from different classes is equivalent to build an oblivious decision tree with minimum height that correctly classifies all samples (which corresponds to select a subset of features of minimum size). When we extend the problem to non-uniform costs, we are interested in finding a subset of features of minimum cost that correctly classifies all samples. We note that since in this particular problem we are not interested in the probability distribution of the samples, and since each level has only one test associated

with it, the cost of the tree is defined simply as the sum of the costs of the tests in the tree, and we refer to the optimal cost an an oblivious decision tree for an instance I of the problem as $OPT(I)$ (note that the definitions of $OPT_W(I)$ and $OPT_E(I)$ do not make sense here). In the last question studied in this Thesis, we relax the assumption that no classification errors are allowed and introduce another parameter in our instance I, an integer $k > 0$. The goal is to compute an oblivious decision tree with minimum cost that incurs in at most k classification errors.

1.3 Our Results

To answer the first question, we present a polynomial time algorithm that achieves an $O(log(n))$ approximation with respect to both the expected testing cost and the worst testing cost, simultaneously. This is the best possible approximation achievable with respect to either optimization measure, under the assumption that $\mathcal{P} \neq \mathcal{NP}$ (Chakaravarthy et al. (2007), Laber & Nogueira (2004)). The main idea of the algorithm is to build a sequence of tests which takes into account both the number of pairs of objects and the probability mass of the objects in S. Previous results achieved $O(log(n))$ approximations for the worst testing cost, but for the expected testing cost this approximation ratio was achieved only in the special cases of the problem where each object belongs to a different class (known as the identification problem) or where the costs of the tests are uniform. For the more general case of the problem, only $O(log(1/p_{min}))$ approximations were known, where p_{min} is the minimum positive probability among the objects in S.

Our method to obtain the first contribution, from a high-level perspective, closely follows the one used by Gupta et al. (2010) 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 Gupta et al. (2010). First, it is more transparently linked to the structure of the problem, which remained somehow hidden in Gupta et al. (2010), where the result was obtained via an involved mapping from adaptive TSP. Second, our algorithm avoids expensive computational steps as the Sviridenko procedure (Sviridenko (2004)) 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 Gupta et al. (2010) is an additional contribution of this Thesis.

For the second question, we first show that for every $\rho > 0$ and every instance I of the problem, there exists a decision tree D with worst testing cost at most $(1 + \rho)OPT_W(I)$ and expected testing cost at most $(1/(1 - e^{-\rho}))OPT_E(I)$, where $OPT_W(I)$ (resp. $OPT_E(I)$) denote the cost of the decision tree with minimum worst testing cost (resp. minimum expected testing cost) for the instance I. We present a procedure that combines two trees, one with worst cost W and another with expected cost E, producing a tree with worst cost at most $(1+\rho)W$ and expected cost at most $(1/(1-e^{-\rho}))E$. We then show that this is a sharp characterization of the best possible trade-off attainable, showing that there are infinitely many instances for which we cannot obtain a decision tree with both worst cost smaller than $(1+\rho)OPT_W(I)$ and expected cost smaller than $(1/(1 - e^{-\rho}))OPT_E(I)$.

For the third question, we present a greedy algorithm for the minimization of the worst testing cost for the value dependent variation of the $DFEP$ and prove that our algorithm is an $O(log(n))$ approximation for the case where all tests have two outcomes of binary tests (i. e., $\ell = 2$). When a test can have more than two different answers, however, we show that a greedy strategy is not sufficient to provide an $O(log(n))$-approximation. In fact, we present an instance of the $DFEP$ for which the algorithm produces a tree with worst cost $\Omega(n)$ times worse than the optimal worst cost. We then present a second greedy algorithm that attains an $O(n)$-approximation for multiway tests.

Finally, our main contribution to answer the last question is a randomized rounding algorithm that, given an instance of the problem (with an additional integer $k \geq 0$) and a parameter $0 < \epsilon < 1/2$, builds an oblivious decision tree with cost at most $(3/(1-2\epsilon))ln(n)OPT(I)$ and produces at most (k/ϵ) errors, where $OPT(I)$ denotes the cost of the oblivious decision tree with minimum cost among all oblivious decision trees for instance I that make at most k classification errors.

1.4 Thesis Organization

This Thesis is organized as follows: In Chapter 2 we present related works. Chapter 3 presents an algorithm that produces a decision tree that attains a logarithmic approximation simultaneously for both worst and expected cost. In Chapter 4 we characterize the best possible trade-off achievable when optimizing the construction of decision trees with respect to both the worst and expected cost. Chapter 5 presents a greedy algorithm to approximate the worst

testing cost that attains a logarithmic approximation for the value dependent variant of the $DFEP$. Chapter 6 presents a randomized rounding algorithm to approximate oblivious decision trees that can make at most k classification errors, for a given $k \geq 0$. Finally, in Chapter 7 we summarize our results.

2
Related Work

In this Chapter, we present previous works related to the problems studied in this Thesis.

We first consider a special version of the DFEP where each object belongs to a different class. In addition, in this version each test has a fixed cost (independent of its outcome), and the goal is to compute a decision tree that correctly classifies all objects, minimizing the worst or the expected testing cost spent. This special case of the $DFEP$ is known as the *identification problem* and was studied by several authors (Adler & Heeringa (2008); Chakaravarthy et al. (2007, 2009); Guillory & Bilmes (2009); Kosaraju et al. (1999); Arkin et al. (1993); Hanneke (2006); Gupta et al. (2010)). Both the minimization of the worst cost and of the expected cost, even with uniform testing costs and binary tests (i. e., when $\ell = 2$) are NP-Complete problems (Hyafil & Rivest (1976)), and none of these goals admit a sublogarithmic approximation unless $\mathcal{P} = \mathcal{NP}$, as shown by Laber & Nogueira (2004) and Chakaravarthy et al. (2007).

Algorithms for the minimization of the worst testing cost of the identification problem were given by Arkin et al. (1993) and Hanneke (2006). Arkin et al. (1993), presents a $\log(n)$ approximation algorithm for the version of identification problem with binary tests and uniform testing costs. For multiway tests and non-uniform testing costs, a logarithmic approximation was presented by Hanneke (2006). The minimization of the expected testing cost was studied by Adler & Heeringa (2008); Chakaravarthy et al. (2007, 2009); Guillory & Bilmes (2009) and Kosaraju et al. (1999). Kosaraju et al. (1999) presented an $O(\log n)$ approximation for uniform costs and binary tests and non-uniform probabilities. The same approximation factor was obtained by Adler & Heeringa (2008), for uniform probabilities, binary tests and non-uniform costs. Chakaravarthy et al. (2007) presented an $O(\log^2 n)$ approximation for uniform costs, multiway tests and non-uniform probabilities. This approximation factor was further improved by Chakaravarthy et al. (2009) to $O(\log n)$ for uniform probabilities. Finally, the general case of multiway tests, non-uniform probabilities and non-uniform testing costs was studied by Guillory & Bilmes (2009), which presents an $O(\log 1/p_{min})$ approximation.

Most of the strategies mentioned above are based in a strategy known as *Generalized Binary Search* (GBS). Algorithms that employ this approach greedily select a test t that minimizes the ratio between the testing cost and the balance of the partition induced by t on the set of objects. However, the best known result to minimize the expected testing cost for the identification problem is due to Gupta et al. (2010) and cannot be seen, at least directly, as a GBS. It achieves, using new techniques, an $O(\log n)$ approximation for the most general case, with multiway tests and non-uniform testing costs and probabilities.

We now move to the case where m can be smaller than n, i. e., we drop the restriction that states that each object belongs to a different class, for fixed testing costs, and the goal is again to minimize the worst or the expected testing cost, considering only decision trees that correctly classify all objects. This problem was studied in the literature under the name of *group identification problem* (Bellala et al. (2012)) and *class equivalence problem* (Golovin et al. (2010)). Both Bellala et al. (2012) and Golovin et al. (2010) provide $O(\log 1/p_{min})$-approximation algorithms for the minimization of the expected cost, for non-uniform testing costs, and both approximation factors can be converted to $O(\log n)$ using a technique presented in Kosaraju et al. (1999) when the testing costs are uniform. The algorithm presented in Bellala et al. (2012) considers only binary tests, while Golovin et al. (2010) addresses multiway tests. For the minimization of the worst testing cost, Saettler (2013) presented a greedy algorithm that achieves a $O(\log n)$ approximation.

Algorithms for the simultaneous minimization of both worst and expected costs of decision trees are known for the prefix code problem (Buro (1993); Garey (1974); Larmore (1987); Larmore & Hirschberg (1990); Milidiú & Laber (2001)) and for the *identification problem* (Cicalese et al. (2010)). The works that consider the prefix code problem provide algorithms to construct L-restricted prefix codes. An L-restricted prefix code is a prefix code where no codeword has length greater than L (i. e., the height of the decision tree associated with the prefix codes cannot exceed L). In particular, Milidiú & Laber (2001) also show that there exists a decision tree that is arbitrarily close to the optimum with respect to both expected and worst cost. More precisely, for every instance I with n objects and any $\rho > 0$, there is a decision tree D such that $cost_W(D)/OPT_W(I) \leq (1+\rho)$ and $cost_E(D)/OPT_E(I) \leq 1+1/\psi^{\rho \log n - 1}$, where ψ is the golden ratio $(1+\sqrt{5})/2$. A previous version of the algorithm presented in Chapter 4 were studied by Saettler (2013). However, that version of the algorithm is much simpler than its actual version, and the upper bound achieved by the algorithm for the expected cost is $(1+1/\rho)E$. Moreover, the

authors do not present lower bounds for their algorithm (in the sense that they do not prove that the analysis is sharp). The techniques used to obtain the upper bound in Chapter 4 are similar to those used in Aslam et al. (1999) and Rasala et al. (2002) to obtain tight trade-offs between the minimization of the expected completion time and the makespan for scheduling problems. However, our upper bound was achieved independently in the sense that we became aware of the scheduling paper when our bound had already been developed. In addition, the upper bounds in the scheduling context were no formally proved (Rasala et al. (2002)). Still with regard to this connection, we shall mention that the scheduling bounds give no clue on how to obtain lower bounds in the decision tree context.

In Cicalese et al. (2010), the authors consider the *identification problem* and provide an algorithm that attains a bicriteria approximation of $O(log(n))$ for non-uniform costs, uniform probabilities and multiway tests. Their algorithm also achieves an $O(log(n))$ approximation for the minimization of the expected testing cost where the costs are uniform and the probabilities are non-uniform. When both costs and probabilities are non-uniform, their algorithm achieves an $O(log(1/p_{min}))$ approximation for the minimization of the expected cost.

Algorithms to build oblivious decision trees are studied in the context of feature selection (Langley & Sage (1994); Schlimmer et al. (1993); Kohavi & Li (1995)) to obtain minimal sets of relevant features of datasets. These works consider fixed costs for each test. Langley & Sage (1994) presented a greedy approach that starts with a full oblivious decision tree and, in each step, removes one attribute (the attribute whose removal produces the most accurate decision tree). This process continues while the accuracy of the resulting tree does not decrease. Schlimmer et al. (1993) present an exponential breadth-first search algorithm to determine all minimal sets of attributes consistent with the training data (i. e., that correctly classify all objects), using a user-defined parameter to reduce the size of the search space. Kohavi & Li (1995) present a top-down greedy procedure to construct oblivious decision trees using the concept of mutual information. The algorithm does not consider only decision trees that correctly classify all objects, since the goal of the authors was to produce a learning algorithm, rather than an approximation algorithm to produce a decision tree that exactly fits the input data.

Finally, we note that the problem of constructing an oblivious decision tree that does not incur in any classification error can be solved by executing an algorithm to solve the Set Cover problem. We do this by constructing an instance of the Set Cover $I' = (U, \mathcal{S})$ from an instance I of the $DFEP$. Each

element in U corresponds to a pair of objects with different classes in I, and to each attribute a in I we have a corresponding set S_a in $\mathcal{S}$ that covers all elements of U associated with the pairs of objects separated by a. It is not obvious how to extend this approach when we allow that a given number of classification errors can be made. For the variation of the problem where we want to guarantee only that at most k pairs of objects from different classes remain together, but no classification errors are allowed, an f-approximation (where f is the highest frequency of any element) can be obtained using the algorithm given in Gandhi et al. (2001) for the the *k-partial set cover problem*. Moreover, this problem optimizes a submodular function, and thus admits a logarithmic approximation using Wolsey's algorithm (Wolsey (1982*a*)). The same situation does not occur when we change the problem by defining k as the maximum number of classification errors allowed, rather than a limit on the the number of pairs of objects from different classes that remain together. Therefore we cannot use an algorithm to approximate submodular set functions to solve the problem.

3
An $O(\log n)$ bicriteria Approximation for the DFEP

In this Chapter, we extend the techniques presented in Gupta et al. (2010) for the identification problem and present an algorithm for the $DFEP$ 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. 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 this Chapter, we assume that the cost function assigns to each test t a number in the set $\mathbb{N}^+$.

3.1
Preliminaries

Let $I = (S, T, C, \mathbf{p}, \mathbf{c})$ 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. We recall that, for any $Q \subseteq S.$, we use $P(Q)$ the number of pairs of Q.

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 $p(Q) = \sum_{s \in Q} p(s)$.

3.2 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 out to be 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

$$sepcost(I, \mathbf{t}) = \sum_{s \in S} p(s) sepcost(I, \mathbf{t}, s). \tag{3-1}$$

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

$$sepcost^*(I) \leq sepcost(I, \mathbf{t}) = \sum_{s \in S} p(s) sepcost(I, \mathbf{t}, s) \tag{3-2}$$

$$\leq \sum_{s \in S} p(s) cost(D, s) \tag{3-3}$$

$$= OPT_E(I). \tag{3-4}$$

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

$$totcost(I, \mathbf{t}) = sepcost(I, \mathbf{t}, s) \leq cost(D, s) \leq cost_W(D). \tag{3-5}$$

Using (3-5), we have

$$totcost^*(I) \leq totcost(I, \mathbf{t}) \leq cost_W(D) = OPT_W(I). \tag{3-6}$$

The proof is complete. ■

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

$$f_1 : R \subseteq T \mapsto P(S) - P(\bigcap_{t \in R} S_t^*) \tag{3-7}$$

$$f_2 : R \subseteq T \mapsto p(S) - p(\bigcap_{t \in R} S_t^*) \tag{3-8}$$

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, submodular function $f : 2^T \mapsto \mathbb{R}_+$

$$\mathcal{P}(f, B, T, \mathbf{c}) : \max_{R \subseteq T} \left\{ f(R) : \sum_{t \in R} c(t) \leq B \right\}. \tag{3-9}$$

Wolsey (1982*b*) studied the solution to the problem $\mathcal{P}$ provided by Algorithm 1 below, called the adapted greedy heuristic.

The following theorem summarizes results from Wolsey (1982*b*) [Theorems 2 and 3].

Theorem 2 *Wolsey (1982*b*) 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 $f(\overline{R}) \geq$*

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: Remove from T all tests with cost larger than B
3: **if** $T \neq \emptyset$ **then**
4: **repeat**
5: $k \leftarrow k + 1$
6: let t_k be a test t which maximizes $\frac{f(A \cup \{t\}) - f(A)}{c(t)}$ among all $t \in T$
7: $T \leftarrow T \setminus \{t_k\},\ spent \leftarrow spent + c(t_k),\ A \leftarrow A \cup \{t_k\}$
8: **until** $spent > B$ or $T = \emptyset$
9: **if** $f(\{t_k\}) \geq f(A \setminus \{t_k\})$ **then** Return $\{t_k\}$
else Return $\{t_1\, t_2\, \ldots\, t_{k-1}\}$

$(1 - e^{-\chi})f(R^*) \approx 0.35 f(R^*)$. *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 9. 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 (3-9).

3.2.1 Achieving logarithmic approximation

Here, we give an overview of how we obtain a logarithmic approximation. First, we focus on the minimization of the expected testing cost, which is the challenging part, and then we discuss the worst testing cost minimization.

The key points for achieving $O(\log n)$-approximation w.r.t. the expected cost. Recall that the first inequalitiy of Theorem 1 guarantees that we can lower bound the optimal expected testing cost for a given instance I by $sepcost^*(I)$, the minimum separation cost achievable by a sequence of tests covering all the pairs of I.

Therefore, the main step consists of showing that we can construct a sequence of tests which covers at least some constant fraction of the total number of pairs and such that the resulting separation cost of such a sequence is within a constant factor of $sepcost^*(I)$ (hence, of the optimal expected testing cost of a decision tree for the same instance). With this result, following a standard approach, we can recursively build a decision tree whose expected cost is at most $O(\log n)$ times the optimal expected testing cost.

The main difficulty in constructing a sequence of tests approximating the optimal separation cost and covering a constant fraction of the total number of pairs is in combining the two goals. In order to cover many pairs we might need a long sequence and we might end up with a lot of probability mass pushed down towards the end of the sequence, hence accounting for a large separation cost. On the other hand, in order to guarantee a small separation cost we might end up with a sequence that does not cover many pairs.

Let's consider these two goals separately. A reasonable approach to obtain a sequence that is effective in covering a large number of pairs at a low cost is to use Wolsey's algorithm with function f_1 defined in (3-7). In fact, by running this algorithm with a budget $\hat{B}$ we end up with a sequence that covers a constant factor of the numbers of pairs covered by the sequence that covers the maximum number of pairs within this budget. If we know that the minimum cost needed to cover all pairs is B^* then we could run Wolsey's algorithm with budget B^* and end up with a sequence, say $\mathbf{t^B}$, that covers a constant fraction of the total number of pairs.

On the other hand, to construct sequences with small separation cost a natural idea is to employ Wolsey's procedure again, but now with function f_2 defined in (3-8), because it greedily selects tests that maximize the mass probability covered per unit of cost. But which budget shall we use? It is possible to prove, though with a considerable effort, that if we run this algorithm with a given budget $\hat{B}$ we end up with a sequence whose separation cost is within a constant factor of that achieved by the sequence with minimum separation cost among those with total cost at least $\hat{B}$. Thus, by running Wolsey's procedure with budget smaller than or equal to B^* (the same B^* of the previous paragraph) we end up with a sequence $\mathbf{t^A}$ whose separation cost is within a constant factor of $sepcost^*(I)$.

Instead of working with B^*, which is NP-Hard to compute, we use the minimum value B such that Wolsey's precedure, with function f_1 and budget B, covers at least $\alpha P(S)$ pairs, where α is the approximation ratio guaranteed by this procedure. We have that B is a lower bound on B^* as proved in Lemma 1.

How can we put together these two goals? If we append $\mathbf{t^B}$ to $\mathbf{t^A}$ (both computed with budget B) we create a new sequence $\mathbf{t_I}$ that covers a constant fraction of the total number of pairs, due to the presence of $\mathbf{t^B}$. In addition, its separation cost is within a constant factor of $sepcost^*(I)$. The last statement holds because the difference between the separation cost of sequence $\mathbf{t_I}$ and that of $\mathbf{t^A}$ is due to the objects that are not covered by $\mathbf{t^A}$. However, these objects have cost (close to) B in $\mathbf{t^A}$ and they cost at most $3B$ in $\mathbf{t_I}$. Thus, they

add at most a constant factor to the separation cost of $\mathbf{t^A}$.

The pseudo-code of our strategy is presented in Algorithm 2. The procedure `FindBudget` is employed to find the value B. The **While** block constructs the sequence $\mathbf{t^A}$. The most technical part of the proof, Lemma 2, consists of proving that the separation cost of this sequence is at a constant factor of $sepcost^*(I)$. The **Repeat** block is responsible for constructing the sequence $\mathbf{t^B}$. The proof that the sequence obtained concatenating $\mathbf{t^A}$ and $\mathbf{t^B}$ covers a constant fraction of the total number of pairs is given by Lemma 3.

In order to complete this high level overview of our central result, let us now give a general idea of the proof that $sepcost(I, \mathbf{t_A})$ is $O(sepcost^*(I))$. As a warm up, let us consider an instance I' of the DFEP where every test has the same cost 1/2. Let B be a lower bound on the cost required to cover all pairs of I'. Let $\mathbf{t}'$ and $\mathbf{t}^*$ be, respectively, the sequence of tests obtained by running Wolsey's algorithm with function f_2 and budget B and a sequence of tests with minimum separation cost among the sequences whose total cost is at least B. Note that $sepcost^*(I') \geq sepcost(I', \mathbf{t}^*)$ because any sequence that covers all pairs has total cost at least B. Moreover, let ℓ be such that $2^{\ell-1} \leq B \leq 2^\ell - 1$. For $j = 0, \ldots, \ell$, let $i_j = 2^{j+2} - 2$ and $i_j^* = 2^{j+1}$. Finally, let $P^{[j]}$ and $P_*^{[j]}$ be, respectively, the sum of the probabilities of the objects covered by the first i_j tests of $\mathbf{t}'$ and the first i_j^* tests of $\mathbf{t}^*$.

By grouping the tests of sequences $\mathbf{t}'$ and $\mathbf{t}^*$ into powers of 2, it is not difficult to show that

$$sepcost(I', \mathbf{t}') \leq 1 + \sum_{j=0}^{\ell-2} 2^{j+1}(1 - P^{[j]}),$$

and

$$sepcost^*(I') \geq sepcost(I', \mathbf{t}^*) \geq \frac{3}{4} + \frac{1}{2}\sum_{j=1}^{\ell-1} 2^j(1 - P_*^{[j]})$$

In addition, it is possible to show that the above upper and lower bounds differ by at most a constant factor. For that we use the fact that sequence $\mathbf{t}'$ is constructed by Wolsey's procedure, with function f_2, which allows us to guarantee that $P^{[j]} - P^{[j-1]} \geq \hat{\alpha}(P_*^{[j]} - P^{[j-1]})$, where $\hat{\alpha}$ is a certain constant.

The previous discussion gives a high level idea of how to obtain a constant approximation for $sepcost^*(I')$ when the instance I' has uniform testing costs. Now, let us focus on an instance I where the testing costs are non-uniform.

We can transform I into a new instance I_U where all testing costs are equal to 1/2 and such that:

(a) $sepcost^*(I_U) \leq sepcost^*(I)$.

(b) the separation cost of the sequence $\mathbf{t^A}$, obtained by Wolsey's procedure for instance I, with function f_2 and budget B, is at most twice the separation cost of that obtained by the same procedure, with the same parameters f_2 and B, for instance I_U.

These properties together with the fact that the separation cost of the sequence constructed by Wolsey's procedure for I_U, with function f_2 and budget B, is within a constant factor of $sepcost^*(I_U)$ imply that the separation cost of $\mathbf{t^A}$ is at a constant factor of $sepcost^*(I)$.

The instance I_U is obtained from I as follows: for each test $t \in T$, we create $2c(t)$ tests in I_U, all of them with cost 1/2; In addition, each object $s \in I$ generates

$$N = 2^{|T|} \prod_{t \in T} c(t)$$

objects in I_U, each of them with probability $p(s)/N$ and with the same class of s. Moreover, the relation between tests and objects in I_U is designed to guarantee: (i) Let $t' \in I_U$ be a test generated by $t \in I$ and let $s' \in I_U$ be an object generated by $s \in I$. If t' covers s' then t covers s (the reciprocal is not necessarily true); (ii) if $t \in I$ covers $s \in I$, then each test generated by t covers exactly $N/(2c(t))$ objects generated by s and every object generated from s is covered by exactly one test generated from t.

The Property (a) holds because if we have a sequence that covers all pairs for instance I we can obtain a sequence that covers all pairs for instance I_U by replacing each test t of the sequence for I with the tests of I_U generated by t. It is easy to prove that the sequence obtained covers all pairs in I_U and its separation cost is smaller than that of the sequence for I.

Property (b) holds because we can guarantee that if Wolsey's procedure applied to instance I, with budget B and function f_2, produces a sequence $\mathbf{t^A} =< t_1^A, \ldots, t_r^A >$ of tests, then the same procedure applied to instance I_U, with budget B and function f_2, produces a sequence $\mathbf{t^U}$ that consists of the concatenation of the tests generated from t_1^A with the tests generated from t_2^A and so on. It is not difficult to see that if the contribution of test t_i^A for the separation cost of sequence $\mathbf{t^A}$ is C then the contribution of the tests generated by t_i^A for the separation cost of $\mathbf{t^U}$ is at least $C/2$. This line of reasoning establishes property (b).

In the previous discussion we hid many technicalities that appear in the proofs of our results. As an example, in Lemma 2, instead of prov-

ing that $sepcost(I, \mathbf{t^A})$ is $O(sepcost^*(I))$ we prove that $sepcost_B(I, \mathbf{t^A})$ is $O(sepcost^*(I))$, where $sepcost_B(I, \mathbf{t^A})$ is a modified separation cost in which all objects not covered by $\mathbf{t^A}$ are charged B rather than its original separation cost. Despite this difference and some others, the essence of our arguments is outlined in the above discussion.

The key points for achieving $O(\log n)$-approximation w.r.t. the worst testing cost. By construction the sequence $\mathbf{t_I}$ obtained through the concatenation of sequences $\mathbf{t^A}$ and $\mathbf{t^B}$ has total cost at most $3B$, where B is the value given by procedure `FindBudget`. In addition, we have that B (as proved in Lemma 1) is a lower bound on the minimum total cost required to cover all pairs. Thus, by recursing $O(\log n)$ times we obtain a logarithmic approximation on the worst testing cost.

Description. Now we detail the algorithm (presented in Algorithm 2) and prove that it attains a logarithmic approximation for DFEP. The algorithm consists of 4 blocks. The first block (lines 1-2) is the base 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 3-4) 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. Then, the tests with cost larger than B are removed from T.

The third (lines 5-11) and the fourth (lines 12-19) blocks are responsible for the construction of the backbone of the decision tree (see Fig. 3.1) 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 5-11) constructs the first part of the backbone (sequence $\mathbf{t}^A$ in Fig.3.1) by iteratively selecting the test that covers the maximum uncovered mass probability per unit of testing cost (line 6). 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. At line 9, 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 $\mathbf{t}^A$.

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: $B \leftarrow$ `FindBudget`$(S, T, C, \mathbf{c})$, $spent \leftarrow 0$, $spent_2 \leftarrow 0$, $U \leftarrow S$, $k \leftarrow 1$
4: Remove from T all tests with cost larger than B
5: **while** there is a test in T of cost $\leq B - spent$ **do**
6: 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 $c(t) \leq B - spent$
7: **If** $k = 1$ **then** make t_1 root of D **else** t_k child of t_{k-1}
8: **for** every $i \in \{1, \ldots, \ell\}$ such that $(S^i_{t_k} \cap U) \neq \emptyset$ **and** $S^i_{t_k} \neq S^*_{t_k}$ **do**
9: Make $D^i \leftarrow$ `DecTree`$(S^i_{t_k} \cap U, T, C, \mathbf{p}, \mathbf{c})$ child of t_k
10: $U \leftarrow U \cap S^*_{t_k}$, $spent \leftarrow spent + c(t_k)$, $T \leftarrow T \setminus \{t_k\}$, $k \leftarrow k + 1$
11: **end while**
12: **if** $T \neq \emptyset$ **then**
13: **repeat**
14: let t_k be a test which maximizes $\frac{P(U)-P(U\cap S^*_t)}{c(t)}$ among all tests $t \in T$
15: Set t_k as a child of t_{k-1}
16: **for** every $i \in \{1, \ldots, \ell\}$ such that $(S^i_{t_k} \cap U) \neq \emptyset$ **and** $S^i_{t_k} \neq S^*_{t_k}$ **do**
17: Make $D^i \leftarrow$`DecTree`$(U \cap S^i_{t_k}, T, C, \mathbf{p}, \mathbf{c})$ child of t_k
18: $U \leftarrow U \cap S^*_{t_k}$, $spent_2 \leftarrow spent_2 + c(t_k)$, $T \leftarrow T \setminus \{t_k\}$, $k \leftarrow k + 1$
19: **until** $B - spent_2 < 0$ **or** $T = \emptyset$
20: $D' \leftarrow$ `DecTree`$(U, T, C, \mathbf{p}, \mathbf{c})$; Make D' a child of t_{k-1}
21: 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

Similarly, the fourth block (the **repeat-until** loop) constructs the second part of the backbone (sequence $\mathbf{t}^B$ in Fig.3.1) by iteratively selecting the test that covers the maximum number of uncovered pairs per unit of testing cost (line 14). The line 20 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 6 (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 14 (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.

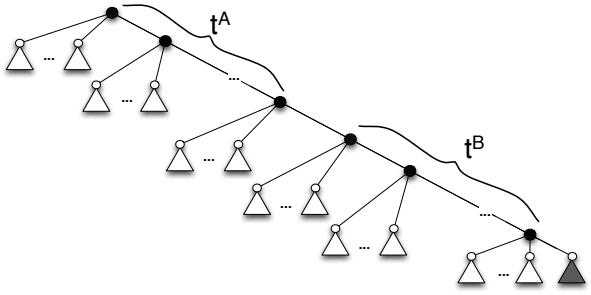

Figure 3.1: The structure of the decision tree built by `DecTree`: white nodes correspond to recursive calls. In each white subtree, the number of pairs is at most $P(S)/2$, while in the lowest-right gray subtree it is at most $8/9P(S)$ (see the proof of Theorem 4).

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 $totcost(I, \mathbf{t}_I) \leq 3totcost^*(I)$.*

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.

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 9,17 and 20. We have that

$$\frac{cost_E(D^{\mathbb{A}}(I))}{OPT_E(I)} = \frac{sepcost(I, \mathbf{t}_I) + \sum_{I' \in \mathcal{I}} cost_E(D^{\mathbb{A}}(I'))}{OPT_E(I)} \tag{3-10}$$

$$\leq \frac{sepcost(I, \mathbf{t}_I)}{OPT_E(I)} + \max_{I' \in \mathcal{I}} \frac{cost_E(D^{\mathbb{A}}(I'))}{OPT_E(I')} \tag{3-11}$$

$$\leq \delta + \max_{I' \in \mathcal{I}} \frac{cost_E(D^{\mathbb{A}}(I'))}{OPT_E(I')} \tag{3-12}$$

$$\leq \delta + \max_{I' \in \mathcal{I}} \{1 + \beta \log P(I')\} \tag{3-13}$$

$$\leq \delta + 1 + \beta \log 8P(S)/9 = 1 + \beta log(P(S)). \tag{3-14}$$

The first equality follows by the recursive way the algorithm `DecTree` builds the decision tree. Inequality (3-11) follows from (3-10) by the subadditivity property (Proposition 2) and simple algebraic manipulations. The inequality in (3-12) follows by Theorem 3 together with Theorem 1 yielding $sepcost(I, \mathbf{t}_I) \leq \delta\, OPT_E(I)$. The inequality (3-13) follows by induction (we are using $P(I')$ to denote the number of pairs of instance I').

To prove that the inequality in (3-14) holds we have to argue that every instance $I' \in \mathcal{I}$ has at most $\frac{8}{9}P(S)$ pairs. Let $U^i_{t_k} = S^i_{t_k} \cap U$ as in the lines 9 and 16. First we show that the number of pairs of $U^i_{t_k}$ is at most $P(S)/2$. We have $S^i_{t_k} \neq S^*_{t_k}$ and $S^*_{t_k}$ is the set with the maximum number of pairs in the partition $\{S^1_{t_k}, \ldots, S^\ell_{t_k}\}$, induced by t_k on the set S. It follows that $P(U^i_{t_k}) \leq P(S^i_{t_k}) \leq P(S)/2$. Now it remains to show that the instance I', recursively called, in line 20 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

$$\frac{cost_W(D^{\mathbb{A}}(I))}{OPT_W(I)} \leq \frac{totcost(I, \mathbf{t}_I) + \max_{I' \in \mathcal{I}}\{cost_W(D^{\mathbb{A}}(I'))\}}{OPT_W(I)} \tag{3-15}$$

$$\leq \frac{totcost(I, \mathbf{t}_I)}{OPT_W(I)} + \max_{I' \in \mathcal{I}} \frac{cost_W(D^{\mathbb{A}}(I'))}{OPT_W(I')} \tag{3-16}$$

$$\leq \frac{totcost(I, \mathbf{t}_I)}{totcost^*(I)} + \max_{I' \in \mathcal{I}} \frac{cost_W(D^{\mathbb{A}}(I'))}{OPT_W(I')} \tag{3-17}$$

$$\leq 3 + 1 + \rho \log(8P(S)/9) = 1 + \rho \log(P(S)) \tag{3-18}$$

Inequality (3-16) follows from the subadditivity property (Proposition 2) for the worst testing cost. The inequality (3-17) follows by Theorem 1. The inequality (3-18) 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 $8/9P(S)$ 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. ■

3.2.2 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 (3-9) 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 (3-9)—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. ■

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`.

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 $f' : R \subseteq T - \mathbf{t}^A \mapsto P(S) - P(U \cap \left(\bigcap_{t \in R} S_t^*\right))$ 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 $totcost(I, \mathbf{t}_I) \leq 3B$. ■

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,

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

where the last inequality follows from Lemma 2.

It remains to show that $totcost(I, \mathbf{t}_I) \leq 3totcost^*(I)$. 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.

3.3 $O(\log n)$ 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 sublogarithmic approximation is achievable for the minimum set cover problem under the standard assumption that $P \neq NP$. More precisely, by the result of Raz & Safra (1997) 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 $P = NP$ (Raz & Safra (1997); Feige (1998)).

We will show if an $o(\log n)$ approximation algorithm exists for minimization of the expected testing cost for the DFEP with exactly b classes ($b \geq 2$), then the same approximation can be achieved for the Minimum Set Cover problem. Due to the above inapproximability result for the Minimum Set Cover problem it follows 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. 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 the objects in U and value $i-1$ for the object o_i $(1 \leq i \leq b-1)$.

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 a 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 is 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

$$OPT_E(I_{DFEP}) = cost_E(D^*) \leq cost_E(D) \leq (h+1) = OPT(I_{SC}) + 1 \quad (3\text{-}19)$$

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

$$|\mathcal{T}| \leq cost(D, o_1) \leq \frac{cost_E(D)}{1-\eta(n+b-2)} \leq 2cost_E(D). \quad (3\text{-}20)$$

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

$$cost_E(D) \leq \alpha \log(n+b-1) OPT_E(I_{DFEP}) \quad (3\text{-}21)$$
$$\leq \alpha(OPT(I_{SC})+1)\log(n+b) \quad (3\text{-}22)$$
$$\leq 4\alpha \log n OPT(I_{SC}) \quad (3\text{-}23)$$

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

From D, as seen above we can construct a solution $\mathcal{T}$ for the set cover problem such that $|\mathcal{T}| \leq 2cost_E(D)$. 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 Raz & Safra (1997) 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 (3-19) becomes $OPT_W(I_{DFEP}) \leq OPT(I_{SC}) + 1$ and (3-20) becomes $|\mathcal{T}| \leq cost_W(D)$, leading to the inapproximability of the DFEP with respect to 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 (2003) based on the inapproximability result on Minimum Set Cover of Feige (1998).

We have proved the following theorem

Theorem 5 *If $P \neq NP$, the DFEP does not admit an $o(\log n)$ approximation neither for the minimization of the worst case testing cost nor for the minimization of the expected testing cost.*

3.4 Conclusions and Open Problems

In this Chapter, we presented an algorithm for the $DFEP$ that achieves an $O(log(n))$ approximation with respect to both the expected testing cost and the worst testing cost, simultaneously, for the classical version of the problem where each test has a fixed cost and no classification errors are allowed. Our result closes the gap left open by the previous $O(\log 1/p_{\min})$ approximation for the expected testing cost shown by Golovin et al. (2010) and Bellala et al. (2012). We recall that our result is the best possible approximation achievable with respect to either optimization measure, under the assumption that $\mathcal{P} \neq \mathcal{NP}$).

4
Trading-Off expected and worst cost in the DFEP

In this Chapter, we characterize the best possible trade-off achievable when optimizing the construction of a decision tree with respect to both the worst and the expected cost. We show that for every $\rho > 0$ there is a decision tree D with worst testing cost at most $(1 + \rho)OPT_W$ and expected testing cost at most $\frac{1}{1-e^{-\rho}}OPT_E$, where OPT_W and OPT_E denote the minimum worst testing cost and the minimum expected testing cost of a decision tree for the given instance. We also show that this is the best possible trade-off in the sense that there are infinitely many instances for which we cannot obtain a decision tree with both worst testing cost smaller than $(1 + \rho)OPT_W$ and expected testing cost smaller than $\frac{1}{1-e^{-\rho}}OPT_E$.

We first derive an upper bound by presenting a general procedure that merges decision trees built according to different optimization criteria: given a parameter $\rho > 0$, a decision tree D_W with worst testing cost W and a decision tree D_E with expected testing cost E, our merging procedure produces a decision tree D with worst testing cost at most $(1 + \rho)W$ and expected testing cost at most $\frac{1}{1-e^{-\rho}}E$. For the analysis of our procedure we employ techniques from non-linear programming (NLP) (Bazaraa et al. (1993)). Then we make use of the the probability distribution used in the analysis of the upper bound —obtained by the optimal solution of the NLP— as a starting point for constructing non-trivial instances that guarantee that the upper bound is tight.

4.1
Trade-off: Upper Bound

In this section, we show our upper bound on the achievable trade-off between worst and expected testing cost for the decision tree optimization problem. Our proof is constructive, that is, we show a procedure for constructing a decision tree guaranteeing the desired trade off.

Given a positive number j, and two decision trees D_E and D_W for instance I, the procedure `CombineTrees`(D_E,
D_W,j) (See Algorithm 3) constructs a new decision tree D^j for I whose worst testing cost is increased by at most j w.r.t the worst testing cost of D_W,

i.e., $cost_W(D^j) \leq j + cost_W(D_W)$. Our algorithm uses the definition of a *j-replaceable node*, by which we mean a node v in D such that the total cost of the tests on the path from the root of D to v (including v) is larger than j and the cost of the path from the root of D to the parent of v is smaller than or equal to j. The procedure `Trade-Off` repeatedly uses `CombineTrees` to create several decision trees (one of these trees being D_W) with increasingly worst testing cost and chooses the one with the best expected testing cost. We will show that this way it can guarantee the best possible trade off.

Algorithm 3 Computes trade off tree between D_W and D_E

Procedure `CombineTrees`(D_E, D_W, j)

1: $D^j \leftarrow D_E$
2: Traverse D^j and construct $R = \{v \mid v$ is a j-replaceable node of $D^j\}$
3: **for** each $v \in R$ **do**
4: Replace in D^j the subtree rooted at v with D_W
5: **return** D^j

Procedure `Trade-Off`(D_E, D_W, C)

1: **for** $j = 0, \ldots, C$ **do**
2: $D^j \leftarrow$ `CombineTrees`(D_E, D_W, j)
3: $j^* \leftarrow \arg\min_{0 \leq j \leq C} cost_E(D^j)$
4: **return** D^{j^*}

Proposition 3 *The decision tree D^j returned by* `CombineTrees` *has worst testing cost at most $j + cost_W(D_W)$.*

Proof: Let s be an object in S. In the following, we identify s with the leaf associated to it. To establish the proof we show that for all $s \in S$ it holds that $cost(D^j, s) \leq j + cost_W(D_W)$.

If s is not a descendant of a replaceable node in D_E then the cost of the path from the root of D_E to s is at most j. Since this path remains the same in D^j, we have that $cost_W(D^j, s) \leq j$. On the other hand, if s is a descendant of a replaceable node v in D_E, then $cost(D^j, s) \leq j + cost_W(D_W)$ because $cost(D^j, s)$ is the sum of (i) the cost of the path from the root of D^j to the parent of v, which is at most j, and (ii) the cost to reach s in the decision tree D_W, which is at most $cost_W(D_W)$ ■

Now we analyze the decision tree $D = D^{j^*}$ output by `Trade-Off`(D_E, D_W, C), where C is an integer parameter. Notice that D is the decision tree with minimum expected testing cost among the decision trees $D^0, D^1, D^2, \ldots, D^C$, where D^j is the decision tree returned by `CombineTrees`(D_E, D_W, j). It follows from the previous proposition that $cost_W(D) \leq C + cost_W(D_W)$.

The analysis of the expected testing cost of D is more involved. In order to simplify the notation we will let $W = cost_W(D_W)$. We also assume for simplicity in the following that test costs are integers. Given a decision tree D' and an object/leaf $s \in S$ with $cost(D', s) = \kappa$ we will say that s has cost κ in D'.

Let p_i, with $i = 1, \ldots, C$, be the sum of the probabilities of objects with cost i in D_E and p_{C+1} be the sum of the probabilities of the objects with cost larger than C in D_E. Clearly:

$$cost_E(D_E) \geq \sum_{i=1}^{C+1} p_i \cdot i$$

Furthermore, for $j = 0, \ldots, C$, we have that:

$$cost_E(D^j) \leq \sum_{i=1}^{j} p_i \cdot i + \left((j + W) \sum_{i=j+1}^{C+1} p_i \right)$$

because the objects whose cost in D_E is larger than j have cost at most $j + W$ in D^j.

Moreover, for a probability distribution $\mathbf{q} = (q_1, \ldots, q_{C+1})$, let

$$f(\mathbf{q}) = \min_{j=0,\ldots,C} \left\{ \frac{\sum_{i=1}^{j} q_i \cdot i + (j + W) \sum_{i=j+1}^{C+1} q_i}{\sum_{i=1}^{C+1} q_i \cdot i} \right\}.$$

and let $\mathbf{p} = (p_1, \ldots, p_{C+1})$. Thus, we have

$$\frac{cost_E(D)}{cost_E(D_E)} = \min_{j=0,\ldots,C} \frac{cost_E(D^j)}{cost_E(D_E)} \leq f(\mathbf{p}) \leq \max_{\mathbf{q} \in \mathcal{P}} f(\mathbf{q}), \tag{4-1}$$

where $\mathcal{P} = \{(q_1, q_2, \ldots, q_{C+1}) | \sum_{i=1}^{C+1} q_i = 1 \text{ and } q_1, q_2, \ldots, q_{C+1} \geq 0\}$. The next lemma gives the exact value of $\max_{\mathbf{q} \in \mathcal{P}} f(\mathbf{q})$.

Lemma 4 *Let* $\mathbf{p}^* = (p_i^*, \ldots, p_{C+1}^*)$*, with*

$$p_i^* = \begin{cases} \frac{(W-1)^{i-1}}{W^i}, & i = 1, \ldots, C \\ \\ \frac{(W-1)^C}{W^C}, & i = C + 1 \end{cases} \tag{4-2}$$

We have that $\mathbf{p}^* \in \mathcal{P}$ *and*

$$f(\mathbf{p}^*) = \frac{1}{\left(1 - \left(\frac{W-1}{W}\right)^{C+1}\right)} = \max_{\mathbf{q} \in \mathcal{P}} f(\mathbf{q}).$$

Proof: The statements $\mathbf{p}^* \in \mathcal{P}$ and

$$f(\mathbf{p}^*) = \frac{1}{\left(1 - \left(\frac{W-1}{W}\right)^{C+1}\right)}$$

can be verified through simple calculations (See Appendix B.1).

To prove that $f(\mathbf{p}^*) = \max_{\mathbf{q}\in\mathcal{P}} f(\mathbf{q})$, we first show that

$$\max_{\mathbf{q}\in\mathcal{P}} f(\mathbf{q}) = \max_{\mathbf{q}'} \min_{j=0,\ldots,C} \left\{ \sum_{i=1}^{j} q'_i \cdot i + (j+W) \sum_{i=j+1}^{C+1} q'_i \right\} \quad \text{s. t.} \tag{4-3}$$

$$\sum_{i=1}^{C+1} i \cdot q'_i = 1 \tag{4-4}$$

$$q'_i \geq 0, \;\; i = 1, \ldots, C+1 \tag{4-5}$$

In fact, let $\boldsymbol{r}$ and let $K = \sum_{i=1}^{C+1} i \cdot r_i$. Then, $\mathbf{r}/K$ is a solution for the problem defined by Equations (4-3)-(4-5) and its objective value is $f(\mathbf{r})$. Conversely, let $\mathbf{r}'$ be the optimal solution of the problem defined by Equation (4-3)-(4-5) and let z' be the corresponding objective function. Moreover, let

$$K' = \frac{1}{\sum_{i=1}^{C+1} r'_i}.$$

Then, $K'\mathbf{r}' \in \mathcal{P}$ and $f(K'\mathbf{r}') = z'$,

Therefore, we can analyze the optimum value of the optimization problem defined by Equations (4-3)-(4-5). This problem can be formulated as a linear program as follows:

$$\max z \quad \text{s. t.} \tag{4-6}$$

$$z \quad -\sum_{i=1}^{j} i \cdot q_i - (j+W)\left(\sum_{i=j+1}^{C+1} q_i\right) \leq 0, \;\; j = 0, \ldots, C \tag{4-7}$$

$$\sum_{i=1}^{C+1} i \cdot q_i = 1 \tag{4-8}$$

$$q_i \geq 0, \;\; i = 1, \ldots, C+1 \tag{4-9}$$

Thus, to show that

$$\max_{q\in\mathcal{P}} f(\boldsymbol{q}) = \frac{1}{\left(1 - \left(\frac{W-1}{W}\right)^{C+1}\right)}.$$

it suffices to exhibit a feasible solution for the dual of the above LP with objective value $\frac{1}{\left(1-\left(\frac{W-1}{W}\right)^{C+1}\right)}$.

The dual of the LP defined by Equations (4-6)-(4-8) is given by

$$\min \lambda_E \quad \text{s. t.} \tag{4-10}$$

$$k \cdot \lambda_E - \sum_{j=0}^{k-1}(j+W)\lambda_j - \sum_{j=k}^{C} k\lambda_j \leq 0, \;\; k = 1, \ldots, C+1 \tag{4-11}$$

$$\sum_{j=0}^{C} \lambda_j \geq 1 \tag{4-12}$$

$$\lambda_j \geq 0, \;\; j = 0, \ldots, C \tag{4-13}$$

It is possible to show (see Appendix B.2 for calculations) that $\lambda^* = (\lambda_E^*, \lambda_0^*, \ldots, \lambda_C^*)$, where

$$\lambda_j^* = \frac{W^j (W-1)^{C-j}}{W^{C+1} - (W-1)^{C+1}},$$

for $j = 0, ..., C$ and $\lambda_E^* = \frac{1}{\left(1-\left(\frac{W-1}{W}\right)^{C+1}\right)}$ is a feasible solution for the dual problem, which establishes the lemma. ■

Thus, by setting $C = \lfloor \rho W \rfloor$ we get the following theorem.

Theorem 6 *Fix an instance I of the decision tree optimization problem and let D_E be a decision tree such that $cost_E(D_E) = OPT_E(I)$. For every $\rho > 0$ there exists a decision tree D such that*

$$cost_W(D) \leq (1+\rho) OPT_W(I) \quad \textit{and} \quad cost_E(D) \leq \left(\frac{1}{1-e^{-\rho}}\right) OPT_E(I).$$

Proof: Let $W = OPT_W(I)$, and $C = \lfloor \rho W \rfloor$. Let D_W be a decision tree such that $cost_W(D_W) = W$. It follows from the analysis above that the decision tree D output by `Trade-Off`(D_E, D_W, C) has worst testing cost at most $C + W < (1+\rho)W$ and expected testing cost smaller than

$$\frac{1}{\left(1-\left(\frac{W-1}{W}\right)^{C+1}\right)} OPT_E(I) \leq \left(\frac{1}{1-\left(\frac{W-1}{W}\right)^{\rho W}}\right) OPT_E(I) \leq \left(\frac{1}{1-e^{-\rho}}\right) OPT_E(I)$$

■

4.2 Trade-off: Lower Bound

In this section we show that the result of Theorem 6 is tight in the sense that no better trade off is possible in general. This is proved in Theorem 8 below. The main ingredient of this result is the construction of a family of instances for which the following theorem holds.

Theorem 7 *Fix integers $W > 1$ and $C > 0$, and let $\eta = \eta(W, C)$ be a number such that $\eta < \frac{1}{W^{2(C+1)}}$. There exists an instance I such that the following hold:*

1. $OPT_W(I) \leq W$.

2. $OPT_E(I) \leq (1-\eta)\Big(W\left(1-\left(\frac{W-1}{W}\right)^C\right) + \lfloor \log W \rfloor \left(\frac{W-1}{W}\right)^C\Big) + (W + C + \lfloor \log W \rfloor)\eta$.

3. If a decision tree D for I is such that $cost_W(D) \leq W + C$ then it holds that $cost_E(D) \geq W(1-\eta) - \eta C$.

Theorem 8 *For any fixed $\rho > 0$ and $\epsilon > 0$, there are infinitely many instances I of the decision tree problem such that no decision tree can simultaneously guarantee worst testing cost smaller than $OPT_W(I)(1+\rho)$ and expected testing cost smaller than $OPT_E(I)\left(\frac{1}{1-e^{-\rho}}\right) - \epsilon$*

Proof: Fix integers $W > 1/\rho$ and $C = \lceil \rho W \rceil$. Then, let a value η and an instance I be defined as by the previous theorem. From this result, it follows that every decision tree D, with $cost_W(D) \leq (1+\rho)W \leq W + C$, satisfies $cost_E(D) \geq W(1-\eta) - \eta C \geq W(1 - \eta(1+\rho+1/W))$. Thus,

$$\frac{cost_E(D)}{OPT_E(I)} \geq \frac{1-\eta(1+\rho+1/W)}{\Big(1-\left(\frac{W-1}{W}\right)^C + \frac{\lfloor \log W \rfloor}{W}\left(\frac{W-1}{W}\right)^C\Big)(1-\eta) + \frac{(W+C+\lfloor \log W \rfloor)\eta}{W}} \tag{4-14}$$

$$\geq \frac{1-\eta(1+\rho+1/W)}{\Big(1-\left(\frac{W-1}{W}\right)^{(\rho W+1)}\left(1-\frac{\lfloor \log W \rfloor}{W}\right)\Big)(1-\eta) + \frac{(W+\rho W+1+\lfloor \log W \rfloor)\eta}{W}} \tag{4-15}$$

By definition $\eta \to 0$ for $W \to \infty$. Accordingly, it is not hard to see that the right hand side expression goes to $\frac{1}{1-e^{-\rho}}$ as $W \to \infty$. Therefore, for any $\epsilon > 0$ there exists W_ϵ such that for every $W \geq W_\epsilon$ the right hand side of (4-15) is larger than $\frac{1}{1-e^{-\rho}} - \epsilon$, hence the instance I has the desired property. ■

4.2.1
The structure of the instance I in Theorem 7

Given the integers $W > 1$ and $C > 0$ and the number $\eta < \frac{1}{W^{2(C+1)}}$, we define the following instance $I = (S, T, \mathcal{C}, \mathbf{p}, \mathbf{c})$.

The set of objects S. For the sake of simplifying notation, let $L_W = \lfloor \log W \rfloor$. The set of objects is divided into the objects of type i (for each $i = 1, \ldots, C + L_W$) and *light* objects. The latter will have total probability mass η which will be asymptotically 0, i.e., negligible with respect to the probability of the other (non-light) objects. For each $i = 1, \ldots, C + L_W$ there are 2^i objects of type i, which we denote by $S^{(i)} = \{o_1^{(i)}, \ldots, o_{2^i}^{(i)}\}$.

For each $i = 1, \ldots, C$ and $j = 1, \ldots, 2^i$, the probability of $o_j^{(i)}$ is

$$\frac{(W-1)^{i-1}}{2^i W^i}(1-\eta).$$

Hence, the total probability of objects of type i is

$$p(S^{(i)}) = \frac{(W-1)^{i-1}}{W^i}(1-\eta).$$

Note that this is exactly the probability distribution of the optimal solution of the NLP presented in the previous section adjusted by $(1-\eta)$.

For each $i = C+1, \ldots, C+L_W$ and $j = 1, \ldots, 2^i$, the probability of $o_j^{(i)}$ is $(1-\eta)\left(\frac{W-1}{W}\right)^C \frac{1}{2^C(2^{L_W+1}-2)}$. Hence, for the total cumulative probability of objects of type larger than C we have

$$p\left(S^{(C+1)} \cup \cdots \cup S^{(C+L_W)}\right) = \left(\frac{W-1}{W}\right)^C (1-\eta).$$

Finally, there exists one light object for each non-light object. Each light object has the same probability and we denote by S^L the set of the light objects, and set $p(S^L) = \eta$.

The partition into classes $\mathcal{C}$. Each object belongs to a different class.

A tree representation of the non-light objects. For later purposes it is convenient to visualize the set of non-light objects as a complete binary tree $\mathcal{T}$ of depth $C + L_W$ as shown in Fig. 4.1. By the ith level of $\mathcal{T}$ we understand the set of nodes at distance i from the root.

For $i = 1, \ldots, C + L_W$ the objects of type i are identified with the nodes at level i of $\mathcal{T}$. Therefore, for $i = 1, \ldots, C + L_W$ and $j = 1, \ldots, 2^i$, the jth node (counting from left to right) in level i is identified with object $o_j^{(i)}$ of $S^{(i)}$. We use $O_j^{(i)}$ to denote the set of objects of the subtree of $\mathcal{T}$ rooted at $o_j^{(i)}$.

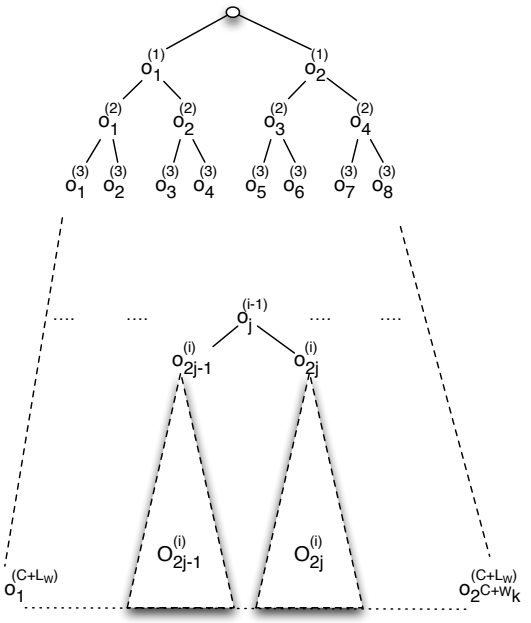

Figure 4.1: The tree representation of the non-light objects.

The set of tests T. The set T of available tests is easily explained with reference to the tree's representation of the objects presented above. The values taken by a test can be interpreted as a partition of the set of objects, each value corresponding to the subset of objects for which the test has that value. Therefore, we describe a test by the way it partitions or splits the set of objects.

There is one test of type 1, which we denote with $t_1^{(1)}$. It splits the objects as follows:

- Group 1. The single object $\{o_1^{(1)}\}$;
- Group 2. The single object $\{o_2^{(1)}\}$;
- Group 3. The set $O_1^{(1)} - \{o_1^{(1)}\}$ and its corresponding light objects;
- Group 4. The set $O_2^{(1)} - \{o_2^{(1)}\}$ and its corresponding light objects.
- Group 5. The two light objects associated with the objects $\{o_1^{(1)}, o_2^{(1)}\}$;

For each $i = 2, \ldots, C + L_W$ and $j = 1, \ldots, 2^{i-2}$ the set T includes a test $t_j^{(i)}$ which splits the set of objects into 5 groups as follows:

- Group 1. The single object $\{o_{2j-1}^{(i)}\}$;
- Group 2. The single object $\{o_{2j}^{(i)}\}$;
- Group 3. $O_1^{(1)} - \{o_1^{(1)}, o_{2j-1}^{(i)}\} - O_{2j}^{(i)}$ and its corresponding light objects;
- Group 4. $O_2^{(1)} \cup \{o_1^{(1)}\} \cup \left(O_{2j}^{(i)} \setminus \{o_{2j}^{(i)}\}\right)$ and its corresponding light objects;
- Group 5. The two light objects associated with the objects $\{o_{2j-1}^{(i)}, o_{2j}^{(i)}\}$;

For each $i = 2, \ldots, C + L_W$ and $j = 2^{i-2} + 1, \ldots, 2^{i-1}$ the set T includes a test $t_j^{(i)}$ which splits the set of objects into 5 groups as follows:

- Group 1. The single object $\{o_{2j-1}^{(i)}\}$;
- Group 2. The single object $\{o_{2j}^{(i)}\}$;
- Group 3. $O_1^{(1)} \cup \{o_2^{(1)}\} \cup \left(O_{2j-1}^{(i)} \setminus \{o_{2j-1}^{(i)}\}\right)$ and its corresponding light objects;
- Group 4. $O_2^{(1)} - \{o_2^{(1)}, o_{2j}^{(i)}\} - O_{2j-1}^{(i)}$ and its corresponding light objects;
- Group 5. The two light objects associated with the objects $\{o_{2j-1}^{(i)}, o_{2j}^{(i)}\}$;

For each $i = 1, \ldots, C + L_W$, we will refer to tests $\{t_j^{(i)} \mid j = 1, \ldots, 2^{i-1}\}$ as the tests of type i. Figures 4.2 and 4.3 illustrate the split corresponding to the test $t_j^{(i)}$ for some $i > 1$. By the test associated with a non-light object o we mean the non-costly test that separates the two children of o, that is, for $o = o_j^{(i)}$ the test associated to o is $t_j^{(i+1)}$. This terminology will be extensively used in our proofs.

Finally, T includes a test denoted by t^* which separates each single object.

The cost of the tests. The tests of type $i = 1, \ldots, C + L_W$ have cost 1 while the test t^* has cost W. We will refer to test t^* as *the costly test.*

Some simple properties of the tests that can be verified by inspection will be used in our analysis.

Fact 1 *The following properties hold for the instance above*

a Let $o_{2j-1}^{(i)}$ and $o_{2j}^{(i)}$ be two non-light objects that are siblings in $\mathcal{T}$. Then, the only tests that separate them are the costly test and the test $t_j^{(i)}$.

b If two light objects are associated with objects that are siblings in $\mathcal{T}$ then the only test that separates them is the costly test.

4.2.2 Proof of Theorem 7

Proof of 1. The first item of Theorem 7 follows because a tree with the costly test t^* at the root has worst testing cost W.

Proof of 2. To prove the second item we construct a decision tree $D^C(I)$ for instance I, which we call the Canonical Decision Tree, and we evaluate its expected testing cost.

If we ignore the leaves—which can be added in the *natural* way—the structure of the nodes associated with tests in the canonical decision tree $D^C(I)$ can be obtained as follows: start with the tree of objects $\mathcal{T}$ and remove

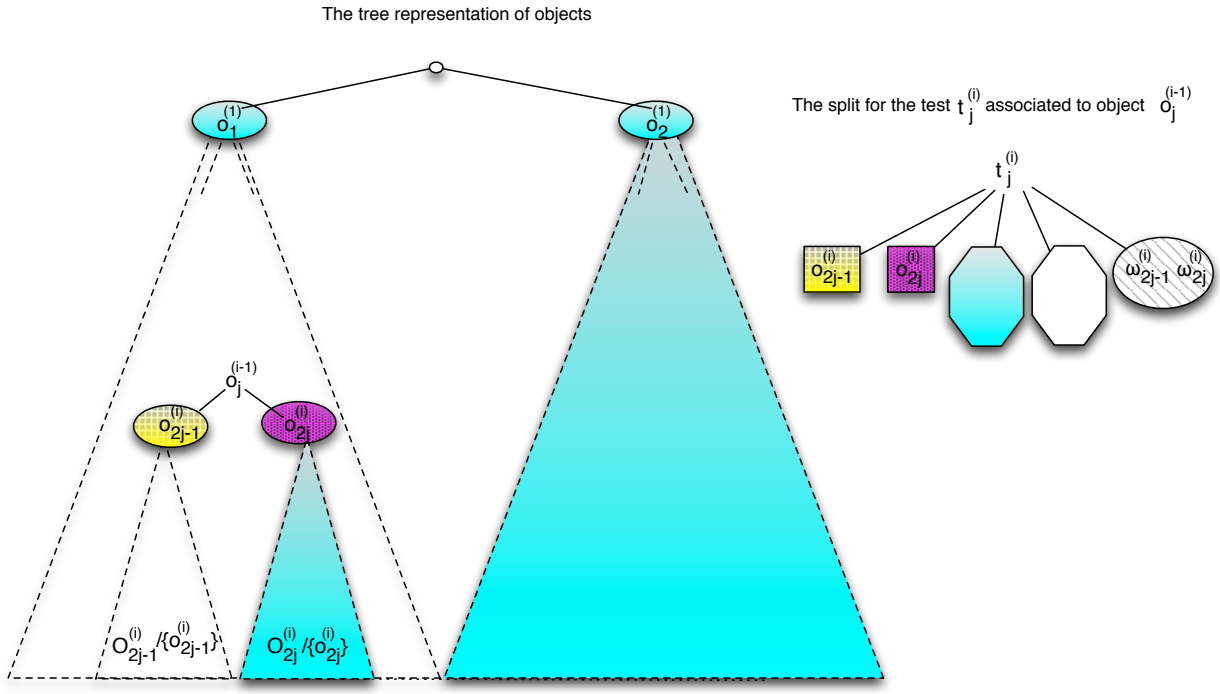

Figure 4.2: The split of a test $t_j^{(i)}$ corresponding to an object in the left subtree of the tree of objects. Groups are represented by different patterns. We denote by $\omega_k^{(i)}$ the light object associated with the object $o_k^{(i)}$.

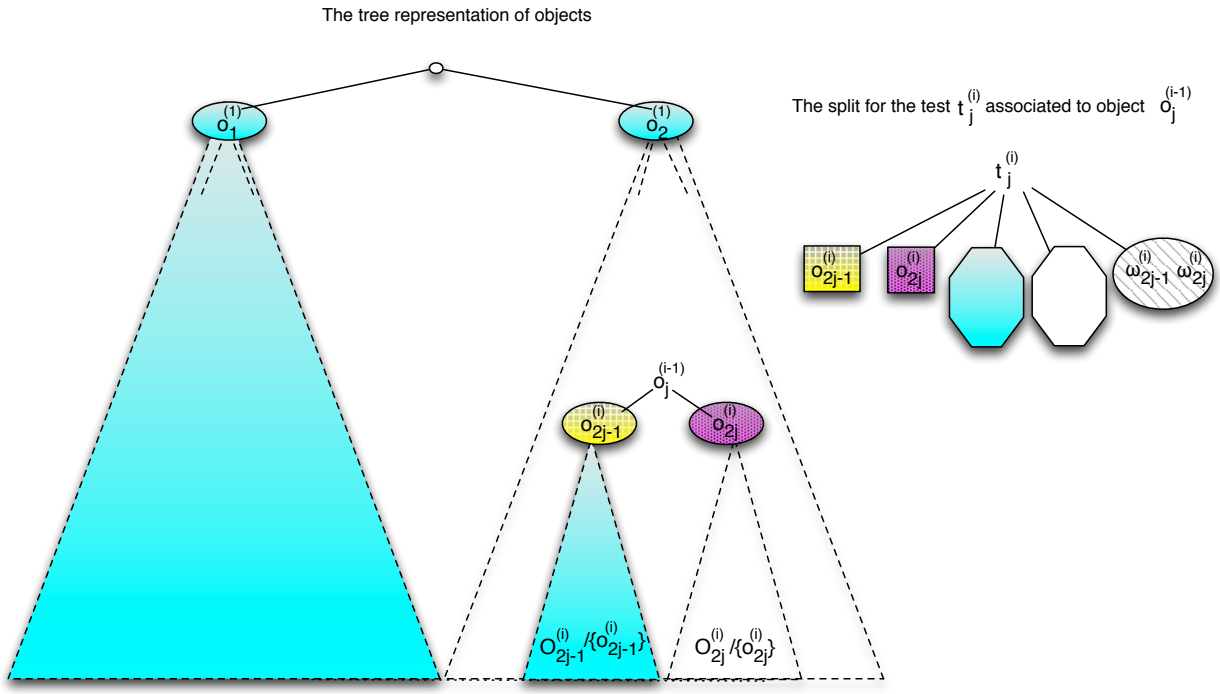

Figure 4.3: The split of a test $t_j^{(i)}$ corresponding to an object in the right subtree of the tree of objects. Groups are represented by different patterns. We denote by $\omega_k^{(i)}$ the light object associated with the object $o_k^{(i)}$.

every object at level $C + L_W$ (the last one).; replace the root of $\mathcal{T}$ with the test $t_1^{(1)}$ and each node $o_j^{(i)}$ with the test $t_j^{(i+1)}$, for $i = 1, \ldots C + L_W - 1$. Identify the edges going from node $o_j^{(i)}$ to its left and right children with the outcomes of $t_j^{(i+1)}$ represented by group 3 and 4 in its definition. Finally, for each $i = 1, \ldots C+L_W$ and $j = 1, \ldots 2^{i-1}$ add a new child to the node associated to test $t_j^{(i)}$ and associate it to the costly test t^*. The branch leading to this new child is associated to the outcome of the test $t_j^{(i)}$ represented by the light objects in group 5 (according to the definition of tests given above). Fig. 4.4 shows the resulting tree.

It is not too hard to verify that

$$cost_E(D^C(I)) \leq (1-\eta)\sum_{j=1}^{C} j\frac{(W-1)^{j-1}}{W^j} + (1-\eta)(C+L_W)\left(\frac{W-1}{W}\right)^C + (W+C+L_W)\eta \quad (4\text{-}16)$$

$$= (1-\eta)W\left(1-\left(\frac{W-1}{W}\right)^C\right) + (1-\eta)L_W\left(\frac{W-1}{W}\right)^C + (W+C+L_W)\eta. \quad (4\text{-}17)$$

Inequality (4-16) follows by observing that in the canonical decision tree every non-light object of type larger than C has cost at most $C+L_W$ and their total probability is $(1-\eta)\left(\frac{W-1}{W}\right)^C$. Accordingly, every light object has cost at most $W+C+L_W$ where the W accounts for the cost of the costly test needed to separate it from the other objects, and η is the total probability mass of the light objects. In order to obtain (4-17) we use

$$\sum_{j=1}^{C} j\frac{(W-1)^{j-1}}{W^j} = W - (C+W)\left(\frac{W-1}{W}\right)^C.$$

Thus, we have proved point 2. of Theorem 7

Proof of 3. In order to establish the last statement of the theorem we will need some additional notation and some intermediate results. For a decision tree D, we use $Obj(\nu)$ to denote the set of non-light objects associated with the leaves in the subtree of D rooted at ν.

We will use the following propositions:

Proposition 4 *Let D be a decision tree for instance I. Let ν be an internal node of D such that $Obj(\nu)$ is non empty. Then there are two sibling nodes/objects of the tree of objects $\mathcal{T}$, name them x_1 and x_2, such that $x_1, x_2 \in Obj(\nu)$ and each object in $Obj(\nu)$ is a descendant of either x_1 or x_2 in $\mathcal{T}$.*

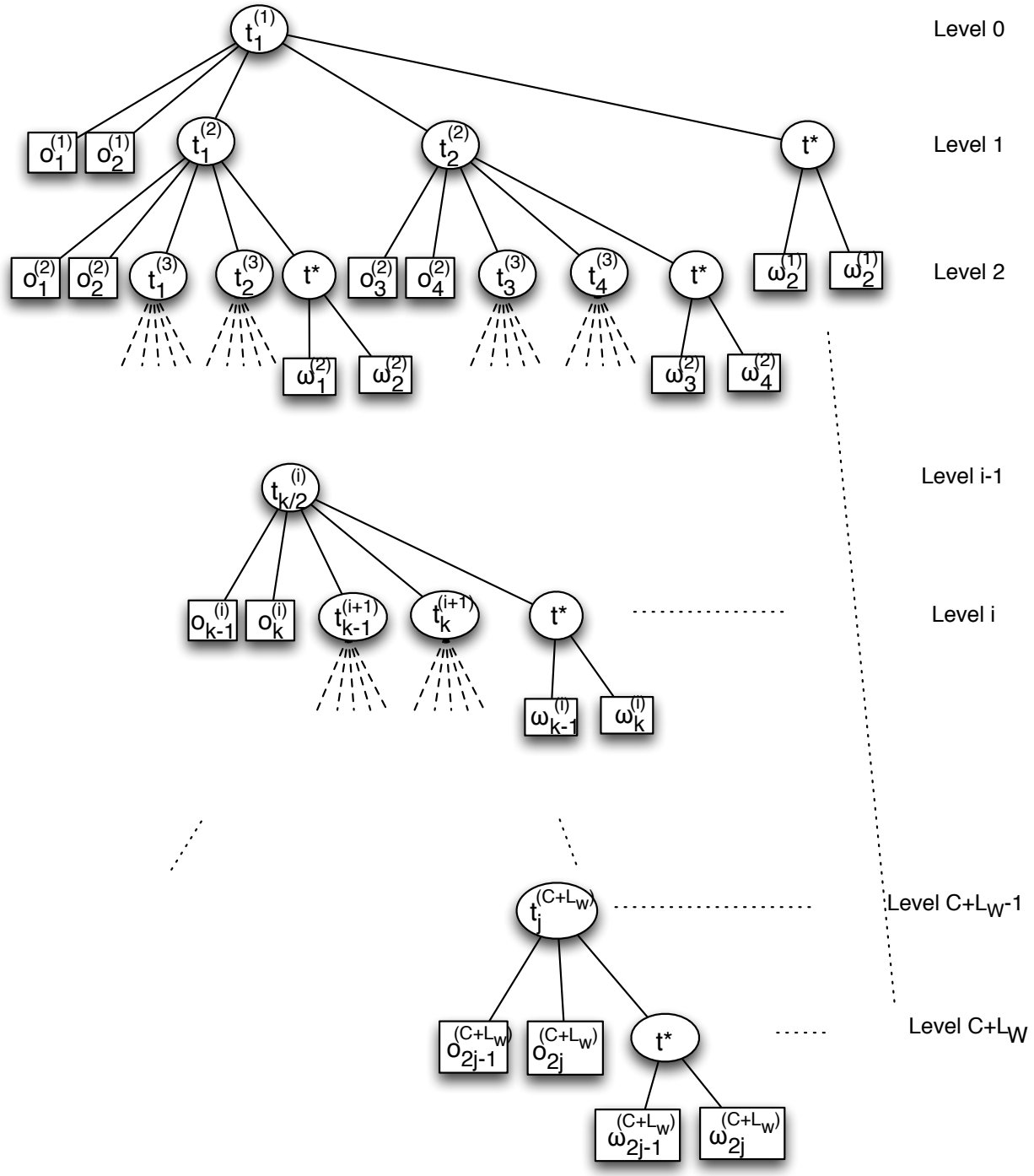

Figure 4.4: The structure of the canonical decision tree $D^C(I)$. Here $\omega_k^{(i)}$ denotes the light-object associated to the (non-light) object $o_k^{(i)}$.

Proof: We say that an object o in $Obj(\nu)$ is maximal if no object in $Obj(\nu)$ is a proper ancestor of o in $\mathcal{T}$. In order to establish the proposition it is enough to show that the set of maximal objects in $Obj(\nu)$ has exactly two objects and those are siblings in $\mathcal{T}$.

Let $\mathcal{M}(v)$ be the set of maximal objects of $Obj(\nu)$. By Fact 1 (a), clearly, if an object belongs to $Obj(\nu)$ then so does its sibling in $\mathcal{T}$. Therefore, if an object belongs to $\mathcal{M}(\nu)$ then so does its sibling in $\mathcal{T}$. For the sake of contradiction, let us assume that we $|\mathcal{M}(\nu)| \geq 3$. In this case $o_1^{(1)}$ and $o_2^{(1)}$ do not belong to $\mathcal{M}(v)$, for otherwise we would have $\mathcal{M}(\nu) = \{o_1^{(1)}, o_2^{(1)}\}$. Let $x_1, x_2 \in \mathcal{M}(\succeq)$ be two siblings in $\mathcal{T}$ and let y be another object in $\mathcal{M}(v)$. We assume that $x_1, x_2 \in O_1^{(1)}$ (the argument for the other case is analogous so we can omit it). We have two cases:

(i) $y \in O_2^{(1)}$. Since $o_1^{(1)}$ does not belong to $Obj(\nu)$ then there is a test in D, which is a proper ancestor of ν and separates $o_1^{(1)}$ from x_1. This test has to satisfy at least one of the following three conditions: (a) it is the test $t_1^{(1)}$; (b) it is associated with the parent of x_1; (c) it is associated with an object o in $O_1^{(1)}$ such that x_1 is not in the right subtree below o in $\mathcal{T}$. However, in all these cases, such a test would also separate x_1 from y which is a contradiction because they are both assumed to be in $Obj(\nu)$.

(ii) $y \in O_1^{(1)}$. Now, let z the least common ancestor of x_1 and y in $\mathcal{T}$. Let z' be the child of z that it is an ancestor of x_1. Note that $z' \neq x_1$, for otherwise y would be a descendant of either x_1 or its sibling x_2 and, as a consequence, it would not be maximal.

Because z' is not in $Obj(\nu)$ there is a test, say ν', that is a proper ancestor of ν in D and it separates z' from x_1. Therefore, one of the following possibilities holds:

- ν' is associated with z;
- ν' is associated with the the parent of x_1;
- ν' associated with an object o that simultaneously satisfy: (a) o is a proper ancestor of x_1 in $\mathcal{T}$; (b) o is a descendant of z' in $\mathcal{T}$ and (c) x_1 lies in the right subtree of o in $\mathcal{T}$.

If ν' is either associated with z or the parent of x_1 then it separates x_1 from y. If ν' is associated with an object that simultaneously satisfies (a), (b) and (c) then it also separates x_1 from y. In all cases, we have a contradiction because x_1 and y are together in $Obj(\nu)$.

■

Proposition 5 *The following inequality holds*[1]*:* $Pr[O_k^{(i)} - o_k^{(i)}] \leq (W - 1)Pr[o_k^{(i)}]$*, for any* $1 \leq i \leq C + L_W$ *and* $1 \leq k \leq 2^i$*.*

Proof: We split the proof into two cases:

Case 1. $i \leq C$. In this case, we have that

$$Pr[O_k^{(i)} - o_k^{(i)}] = \left(\frac{1 - \sum_{s=1}^{i-1} \sum_{j=1}^{2^s} Pr[o_j^s]}{2^i} - \frac{\frac{(W-1)^{i-1}}{W^i}}{2^i} \right)(1 - \eta) =$$

$$\left(\frac{\frac{(W-1)^{i-1}}{W^{i-1}} - \frac{(W-1)^{i-1}}{W^i}}{2^i} \right)(1 - \eta) = (W - 1)Pr[o_k^{(i)}]$$

Case 2. $C < i \leq C + L_W$. In this case, all the objects in $O_k^{(i)}$ have the same probability $(\frac{W-1}{W})^C \frac{1}{2^C(2^{L_W+1}-2)}(1 - \eta)$. Moreover we have $|O_k^{(i)}| \leq 2^{L_W} - 1 \leq W - 1$. Thus, it follows that

$$Pr[O_k^{(i)} - o_k^{(i)}] < (W - 1) \left(\frac{W - 1}{W} \right)^C \frac{1}{2^C(2^{L_W+1} - 2)}(1 - \eta) \qquad (4\text{-}18)$$

$$= (W - 1)Pr[o_k^{(i)}]. \qquad (4\text{-}19)$$

■

We say that a test t (an object o) occurs at cost level κ in a decision tree D if the total cost of tests on the path from the root of D to the parent of t (o) is κ.

Proposition 6 *Let* D *be a decision tree for instance* I *and let* D' *be the tree obtained from* D *by removing all subtrees rooted at costly tests. Then,* D' *has at most* 2^ℓ *objects occurring at cost level* ℓ *for every* ℓ*.*

Proof: First, note that all leaves in D' are associated with non-light objects. Indeed, as a consequence of Fact 1 (b), the deletion of subtrees rooted at costly nodes also removes all leaves associated with light objects.

Let ν be an arbitrarily chosen internal node in D'. Note that it is enough to prove that ν has at most two children that are internal nodes and at most two children that are leaves because, in this case, a simple inductive argument can be used to establish that D' has at most 2^ℓ objects occurring at cost level ℓ for every ℓ.

First, we prove that ν has at most two children that are leaves. Let ν be a node in D' and let $t_j^{(i)}$ be the test corresponding to ν. If ν has more than

[1]for the sake of readability here we use the notation $Pr[\,]$ for the probability of objects

two leaves as children, then there is an object, say o, with $o \notin \{o_{2j-1}^{(i)}, o_{2j}^{(i)}\}$, corresponding to one of these leaves. Let t be the test corresponding to the parent of o in $\mathcal{T}$ ($t_1^{(1)}$ if o has type 1). Since, by Fact 1 (a) t is the only non-costly test separating o from its sibling and t is not the test corresponding to ν, we have that t must be an ancestor of ν. But then o cannot be in $Obj(\nu)$ since it must be a leaf child of the node corresponding to t. Hence we have a contradiction.

Finally, we prove that ν has at most two children that are internal nodes. Note that the node ν in D has at most three children that are internal nodes, corresponding to the groups 3-5 in the definition of the test's splits. However, the internal node associated with group 5 must be a costly test, for otherwise there would be a node in D that does not provide information which is not possible according to our definition of the decision tree problem. Since all costly tests are removed it follows that at most two children are internal nodes.

■

Lemma 5 *Let D be a decision tree for the instance I such that $cost_W(D) \leq W + C$. Then $cost_E(D) \geq W(1-\eta) - \eta C$.*

Proof: Let D be a decision tree with minimum expected testing cost among all decision trees for I with worst testing cost not larger than $W + C$.

First, we argue that every non-costly test in D, that has at least one non-light object as a descendant, occurs at cost level at most $C - 1$. For the sake of contradiction, let us assume that some non-costly test that has at least one non-light object as a descendant occurs at cost level larger than or equal to C. Let ν be the node of D corresponding to such a test and let o be an object in $Obj(\nu)$. Assume that $o \in O_1^{(1)}$ (the proof for the other case is analogous so that we omit it). Both the light object associated with o and the light object associated with o's sibling are also in subtree of D rooted at ν because the only tests that separate them from o are the costly test and the test corresponding to the parent of o ($t_1^{(1)}$ if $o = o_1^{(1)}$). However, none of these tests can be a proper ancestor of ν in D, for otherwise we would have $o \notin Obj(\nu)$. Thus, since ν does not correspond to a costly test, there must be a costly test in the subtree of D rooted at ν to separate these light objects. This implies that $cost_W(D) > C + W$, which is a contradiction.

Now, we argue that there exists a tree $\tilde{D}$ with worst testing cost at most $C + W$ and expected testing cost not much larger than that of D such that all costly tests in $\tilde{D}$, which are ancestors of at least one non-light object, occur at cost level C. For that, let ν be an internal node of D associated with a costly test that occurs at cost level smaller than C

and such that $Obj(\nu)$ is non-empty. By Proposition 4 the set of non-light objects in $Obj(\nu)$ can be partitioned into three groups $\{o_{2j-1}^{(s)}\}$, $\{o_{2j}^{(s)}\}$ and $Obj(\nu)\setminus\{o_{2j-1}^{(s)}, o_{2j}^{(s)}\} \subseteq \left(O_{2j-1}^{(s)} \cup O_{2j}^{(s)}\right) - \{o_{2j-1}^{(s)}, o_{2j}^{(s)}\}$, for some $1 \leq s \leq C+L_W$ and some $1 \leq j \leq 2^s$. From Proposition 5, we have

$$\begin{aligned} Pr[Obj(\nu) \setminus \{o_{2j-1}^{(s)}, o_{2j}^{(s)}\}] &\leq Pr[\{O_{2j-1}^{(s)}\} \setminus \{o_{2j-1}^{(s)}\}] \\ &+ Pr[\{O_{2j}^{(s)}\} \setminus \{o_{2j}^{(s)}\}] \qquad (4\text{-}20) \\ &\leq Pr[\{o_{2j-1}^{(s)}, o_{2j}^{(s)}\}](W-1). \qquad (4\text{-}21) \end{aligned}$$

Let $l(\nu)$ be the set of light objects that are separated by the costly test associated with ν; and let D' be the decision tree obtained by replacing this test with the test $t_j^{(s)}$ and then using costly tests as children of $t_j^{(s)}$ to separate objects of $Obj(v)$ that are grouped together by $t_j^{(s)}$. This modification reduces the cost of the leaves associated with $o_{2j-1}^{(s)}$ and $o_{2j}^{(s)}$ by $W-1$ and increases by 1 the cost of the leaves associated to the objects in $Obj(\nu) \setminus \{o_{2j-1}^{(s)}, o_{2j}^{(s)}\}$ and the cost of the objects in $l(\nu)$. In formulas, using (4-20)-(4-21), we have

$$\begin{aligned} cost_E(D') &= cost_E(D) - (W-1)Pr[\{o_{2j-1}^{(s)}, o_{2j}^{(s)}\}] \\ &\quad + Pr[Obj(\nu) \setminus \{o_{2j-1}^{(s)}, o_{2j}^{(s)}\}] + Pr[l(\nu)] \qquad (4\text{-}22) \\ &\leq cost_E(D) + Pr[l(\nu)]. \qquad (4\text{-}23) \end{aligned}$$

By repeated application of the above transformation we can obtain a decision tree $\tilde{D}$ such that: $cost_W(\tilde{D}) \leq C + W$; for each node ν of $\tilde{D}$ associated with a costly test, either $Obj(\nu)$ is empty or ν occurs at cost level C; $cost_E(\tilde{D}) \leq cost_E(D) + \eta C$. The term ηC in the last inequality comes from (4-22)-(4-23) telling that each repetition of the transformation might increases the cost of some light object by 1. Each light object can be involved in at most C such transformations, since after such a number of transformation the light object would be a leaf child of a costly test at level $\geq C$. Since in total light objects have probability mass η the cumulative increase given by the transformations is at most ηC.

Now, we lower bound the expected testing cost of $\tilde{D}$ by considering only the contribution provided by the non-light objects. Our first observation is that there are at most 2^ℓ non-light objects occurring at level ℓ for $\ell = 1, \ldots, C$. To see that, let $\overline{D}$ be tree obtained from $\tilde{D}$ by removing all subtrees rooted at costly tests. Because all costly tests that have at least one non-light object as a descendant occur at cost level C, it follows that all non-light objects that occur at cost level smaller than or equal to C in $\tilde{D}$ are not affected by the deletion.

Moreover, it follows from Proposition 6 that there are at most 2^ℓ leaves at level ℓ associated with a non-light object in $\overline{D}$, and as a consequence, also in $\tilde{D}$.

For each $\ell = 1, 2, \ldots, C$ let $\tilde{p}_\ell$ be the sum of the probabilities of the *non-light* objects that occur at cost level ℓ in $\tilde{D}$. For each $k = 1, \ldots, C$ we have that

$$\sum_{\ell=1}^{k} \tilde{p}_\ell \leq \left(\sum_{\ell=1}^{k} \frac{(W-1)^{\ell-1}}{W^\ell} \right)(1-\eta). \tag{4-24}$$

In fact, for each k there are at most $2^{k+1} - 2$ leaves associated with non-light objects in the first k levels. In addition the set of $2^{k+1} - 2$ objects of largest probability in I is given by the set of objects of type $1, \ldots, k$, whose cumulative probability coincides with the right-hand-side expression.

Then, ignoring the contribution of the light objects, we can write

$$\begin{aligned}
cost_E(\tilde{D}) &\geq \sum_{\ell=1}^{C} \ell \cdot \tilde{p}_\ell + (C+W)\left((1-\eta) - \sum_{\ell=1}^{C} \tilde{p}_\ell\right) && (4\text{-}25)\\
&= \sum_{j=1}^{C} \left((1-\eta) - \sum_{\ell=1}^{j-1} \tilde{p}_\ell\right) + W\left((1-\eta) - \sum_{\ell=1}^{C} \tilde{p}_\ell\right) && (4\text{-}26)\\
&\geq \left(\sum_{j=1}^{C} \left(1 - \sum_{\ell=1}^{j-1} \frac{(W-1)^{\ell-1}}{W^\ell}\right) \right.\\
&\quad \left. + \; W\left(1 - \sum_{\ell=1}^{C} \frac{(W-1)^{\ell-1}}{W^\ell}\right)\right)(1-\eta) && (4\text{-}27)\\
&= \left(\sum_{j=1}^{C} \frac{(W-1)^{j-1}}{W^{j-1}} + W\left(\frac{(W-1)^C}{W^C}\right)\right)(1-\eta) && (4\text{-}28)\\
&= W(1-\eta) && (4\text{-}29)
\end{aligned}$$

where (4-26) is a rewriting of $cost_E(\tilde{D})$ in terms of the contribution of the internal nodes/tests by cost level; and (4-27) follows from (4-26) because of (4-24). By the construction of $\tilde{D}$ we finally have the desired result $cost_E(D) \geq cost_E(\tilde{D}) - \eta C \geq W(1-\eta) - \eta C$ ■

4.3 Uniform Probabilities

In the construction of the lower bound we used both non-uniform test costs and non-uniform probabilities. In this Section, we extend the lower bound to the case of uniform probabilities by the following transformation: Take an instance $I = (S, \mathcal{C}, T, \mathbf{p}, \mathbf{c})$ as described in the proof of Theorem 7 and produce the instance $\tilde{I} = (\tilde{S}, \tilde{\mathcal{C}}, \tilde{T}, \tilde{\mathbf{p}}, \tilde{\mathbf{c}})$ as follows: for each object $o \in S$ create a class $\tilde{C}_o$ in $\tilde{\mathcal{C}}$ and create $p(o)/\prod_{o \in S} p(o)$ new objects in $\tilde{S}$ setting them as all

belonging to the class $\tilde{C}_o$. Set the probability of each created objects $\tilde{o} \in \tilde{S}$ to $\tilde{p}(\tilde{o}) = \prod_{o \in S} p(o)$. Hence, for each $o \in S$, the total probability of the objects in class $\tilde{C}_o$ is equal to $p(o)$.

Finally for each test $t \in T$ create a corresponding test $\tilde{t}$ in $\tilde{T}$. If object $o \in S$ is in group g of test $t \in T$ then each object $o' \in \tilde{C}_o$ is in group g of test $\tilde{t}$. It is not hard to realise that for the instance $\tilde{I}$ the class $\tilde{C}_o$ behaves exactly as the single objects o in the instance I. This construction shows that the limit on the best trade-off achievable also holds for the case of uniform probabilities.

4.4 Conclusions and Open Problems

In this Chapter, we provided a complete characterization of the best possible trade-off achievable when optimizing the construction of a decision tree with respect to both the worst and the expected cost, for the version of the $DFEP$ where the testing costs are fixed and no classification errors are allowed. We showed that for every $\rho > 0$ there is a decision tree D with worst testing cost at most $(1 + \rho)OPT_W$ and expected testing cost at most $\frac{1}{1-e^{-\rho}}OPT_E$, where OPT_W and OPT_E denote the minimum worst testing cost and the minimum expected testing cost of a decision tree for the given instance. We prove that these bounds are sharp by showing that there are infinitely many instances for which we cannot obtain a decision tree with both worst cost smaller than $(1+\rho)OPT_W(I)$ and expected cost smaller than $(1/(1-e^{-\rho}))OPT_E(I)$. Moreover, we extended the construction of the lower bound considering only uniform probabilities, showing that the limit on the best trade-off achievable also holds for this case.

An interesting question which remains open regards the case of uniform testing costs. We ask whether for every $\epsilon > 0$, there is some integer n_0 such that every instance I with uniform testing costs and with more than n_0 objects, admits a decision tree D such that $cost_E(D) \leq (1 + \epsilon)OPT_E(I)$ and $cost_W(D) \leq (1 + \epsilon)OPT_W(I)$. We notice that this result holds for length restricted prefix codes Milidiú & Laber (2001), the special case of decision tree construction mentioned in Chapter 1.

5
A logarithmic approximation for value dependent testing costs

In this Chapter, we present a greedy algorithm for the minimization of the worst testing cost for the value dependent variation of the $DFEP$ and prove that our algorithm is an $O(log(n))$ approximation for the case where all tests have two outcomes of binary tests. We also show that when a test can have more than two different answers, however, our greedy strategy is not sufficient to provide an $O(log(n))$-approximation. We present an instance of the $DFEP$ for which the algorithm produces a tree with worst cost $\Omega(n)$ times worse than the optimal worst cost. Finally, we present a second greedy algorithm that attains an $O(n)$-approximation for multiway tests.

5.1
The DividePairs Algorithm

We recall that, in this variant of the $DFEP$, if we apply a test t on an object $s \in S$, we get an answer $t(s)$ and pay a cost $c^{t(s)}(t)$. Thus, each test can be associated with ℓ different costs since $t(s) \in \{1, \ldots, \ell\}$. We will also make use of a proposition used in Chapter 3, but for purpose of organization, we restate it here.

Proposition 7 *Let $(S, C, T, \mathbf{p}, \mathbf{c})$ be an instance of the DFEP and let S' be a subset of S. Then, $OPT_W(S') \leq OPT_W(S)$.*

Finally, our algorithm, called DividePairs, chooses for the root of the tree the test t that minimizes:

$$\max_{1 \leq i \leq \ell} \left\{ \frac{c^i(t)}{P(S) - P(S_t^i)} \right\} \tag{5-1}$$

over all available tests that separate at least one pair of objects. Then the objects in S are splitted according to the values of t for each object, and DividePairs is recursively called for each (non empty) new group of objects. When all objects in a group are from the same class, a leaf is created. We analyze the approximation of the algorithm when $\ell = 2$. In this case, each test $t \in T$ splits S into two subsets: S_t^1 and S_t^2.

In order to analyze the algorithm, we use $Cost(S)$ to denote the cost of the decision tree that DividePairs constructs for a set of objects S. Let τ

be the first test selected by DividePairs. We can write the ratio between the worst testing cost of the decision tree generated by DividePairs and the cost of the decision tree with minimum worst testing cost as

$$\frac{Cost(S)}{OPT_W(S)} = \frac{\max\{c^1(\tau) + Cost(S^1_\tau), c^2(\tau) + Cost(S^2_\tau)\}}{OPT_W(S)} \tag{5-2}$$

Let q be such that $c^q(\tau) + Cost(S^q_\tau) = \max\{c^1(\tau) + Cost(S^1_\tau), c^2(\tau) + Cost(S^2_\tau)\}$ in equation (5-2). We have that:

$$\frac{Cost(S)}{OPT_W(S)} = \frac{c^q(\tau) + Cost(S^q_\tau)}{OPT_W(S)} \leq \frac{c^q(\tau)}{OPT_W(S)} + \frac{Cost(S^q_\tau)}{OPT_W(S^q_\tau)} \tag{5-3}$$

where the inequality follows from Proposition 7. If $P(S^q_\tau) = 0$ we have that $Cost(S^q_\tau)/OPT_W(S) = 0$ so that it is not necessary to use Proposition 7. In fact, in this case the fraction $Cost(S^q_\tau)/OPT_W(S^q_\tau)$ is eliminated and (5-3) becomes an equality.

The following lemma shows that $OPT_W(S)$ is at least $c^q(\tau)P(S)/(P(S) - P(S^q_\tau))$.

Lemma 6 *$c^q(\tau)P(S)/(P(S) - P(S^q_\tau))$ is a lower bound on the worst testing cost of the optimal tree.*

Proof: First, we note that in the set of decision trees with minimum worst testing cost, there is a tree D^* in which every internal node has two children. This is true because we could remove nodes with just one child without increasing the worst testing cost. Let v be an arbitrarily chosen internal node in D^*, let γ be the test associated with v and let $R \subseteq S$ be the set of objects associated with the leaves of the subtree rooted at v. Let i be such that $c^i(\tau)/(P(S) - P(S^i_\tau))$ is maximized and j be such that $c^j(\gamma)/(P(S) - P(S^j_\gamma))$ is maximized. We have that:

$$\frac{c^q(\tau)}{P(S) - P(S^q_\tau)} \leq \frac{c^i(\tau)}{P(S) - P(S^i_\tau)} \leq \frac{c^j(\gamma)}{P(S) - P(S^j_\gamma)} \tag{5-4}$$

$$\leq \frac{c^j(\gamma)}{P(R) - P(R^j_\gamma)} \tag{5-5}$$

The last inequality in (5-4) holds due to the greedy choice. To prove inequality (5-5), we only have to show that $P(S) - P(S^j_\gamma) \geq P(R) - P(R^j_\gamma)$. Let r^R_γ (resp. r^S_γ) be the number of pairs in R (resp. S) separated by test γ. Since $R \subseteq S$ we have that $r^R_\gamma \leq r^S_\gamma$ and $P(R^i_\gamma) \leq P(S^i_\gamma)$ for $i = 1, 2$. Also, note that:

$$P(S) = r^S_\gamma + P(S^1_\gamma) + P(S^2_\gamma) \tag{5-6}$$

$$P(R) = r_\gamma^R + P(R_\gamma^1) + P(R_\gamma^2) \tag{5-7}$$

Hence, we have that $P(S) - P(S_\gamma^j) \geq P(R) - P(R_\gamma^j)$. Thus, we have concluded that inequality (5-5) holds.

For a node v, let $S(v)$ be the set of objects associated with the leaves of the subtree rooted at v. Let $v_1, v_2, \ldots, v_p$ be a root-to-leaf path on D^* as follows: v_1 is the root of the tree, and for each $i = 1, \ldots, p-1$ the node v_{i+1} is a child of v_i associated with the branch j that maximizes $c^j(t_i)/(P(S) - P(S_{t_i}^j))$, where t_i is the test associated with v_i. We denote by $c_{t_i}^*$ the cost that we have to pay going from v_i to v_{i+1}. It follows from inequaltity (5-5) that

$$\frac{[P(S(v_i)) - P(S(v_{i+1}))]\, c^q(\tau)}{P(S) - P(S_\tau^q)} \leq c_{t_i}^* \tag{5-8}$$

for $i = 1, \ldots, p-1$. Since the cost of the path from v_1 to v_p is not larger than the worst testing cost of the optimal decision tree, we have that

$$OPT_W(S) \geq \sum_{i=1}^{p-1} c_{t_i}^* \geq \frac{c^q(\tau)}{P(S) - P(S_\tau^q)} \sum_{i=1}^{p-1} (P(S(v_i)) - P(S(v_{i+1}))) = \frac{c^q(\tau) P(S)}{P(S) - P(S_\tau^q)},$$

where the second inequality follows from (5-8) and the last identity holds because $S(v_1) = S$ and $P(S(v_p)) = 0$.

■

Replacing the bound on $OPT_W(S)$ given by the previous lemma in equation (5-3) we get that

$$\frac{Cost(S)}{OPT_W(S)} \leq \frac{P(S) - P(S_\tau^q)}{P(S)} + \frac{Cost(S_\tau^q)}{OPT_W(S_\tau^q)} \tag{5-9}$$

Note that:

$$\frac{P(S) - P(S_\tau^q)}{P(S)} = \sum_{i=1}^{P(S)-P(S_\tau^q)} \left(\frac{1}{P(S)}\right) \leq \sum_{i=1}^{P(S)-P(S_\tau^q)} \left(\frac{1}{P(S_\tau^q) + i}\right) \tag{5-10}$$

We shall prove by induction on the number of pairs that for each $G \subset S$, $Cost(G)/OPT_W(G) \leq H(P(G))$, where $H(n) = \sum_{i=1}^{n} 1/i$. If $P(G) = 1$ (base case), we must have $n = 2$ so that the first test selected by DividePairs, say t, separates G in two leaf nodes. On the other hand, the test in the root of the optimal tree, say t^*, also has to separate the two objects, otherwise it could be discarded. Therefore, for the base case we have that $\max\{c^1(t), c^2(t)\} \leq$

$\max\{c^1(t^*), c^2(t^*)\} = OPT_W(G)$, which implies that $Cost(G)/OPT_W(G) \leq 1$. For the inductive step, from (5-9) and (5-10) we have that

$$\frac{Cost(S)}{OPT_W(S)} \leq \sum_{i=1}^{P(S)-P(S_\tau^q)} \left(\frac{1}{P(S_\tau^q)+i} \right) + H(P(S_\tau^q)) = H(P(S)).$$

Since $P(S) \leq 2\ln(n)$, we have the following theorem

Theorem 9 *There is an $O(\log n)$ approximation for the version of the DFEP with binary tests.*

5.2 Multiway tests

In this section, we show that when $\ell > 2$, the greedy strategy used by DividePairs can build a decision tree with a worst testing cost $\Omega(n)$ times worse than the optimal one.

Let M, C and ϵ be constant values, with $\epsilon \approx 0$, $M > \epsilon$ and $C >> M$. Consider an instance of the DFEP with a set S of n objects (where n is divisible by 3), $s_1, \ldots, s_n$, where each object constitutes a class of its own (this particular case is also referred to as an *identification problem*), and 3 tests. Test t_1 splits S into 3 equal subsets: $S_{t_1}^1 = \{s_1, \ldots, s_{n/3}\}$, $S_{t_1}^2 = \{s_{n/3+1}, \ldots, s_{2n/3}\}$ and $S_{t_1}^3 = \{s_{2n/3+1}, \ldots, s_n\}$. The costs associated with t_1 are $c^1(t_1) = c^2(t_1) = c^3(t_1) = (M-\epsilon)(\binom{n}{2} - \binom{n/3}{2})$. Test t_2 also splits S into 3 subsets: $S_{t_2}^1 = \{s_1\}$, $S_{t_2}^2 = \{s_2\}$ and $S_{t_2}^3 = \{s_3, \ldots, s_n\}$, and has costs $c^1(t_2) = c^2(t_2) = \epsilon$ and $c^3(t_2) = M(\binom{n}{2} - \binom{n-2}{2})$. Finally, test t_3 splits S into $n-1$ subsets: $S_{t_3}^1 = \{s_1, s_2\}$ and, for $i = 2, \ldots, n-1$, $S_{t_3}^i = \{s_{i+1}\}$, and has associated costs $c^1(t_3) = C(\binom{n}{2} - 1)$ and $c^2(t_3) = \ldots = c^{n-1}(t_3) = \epsilon$. Figure 5.1 shows how each test splits S.

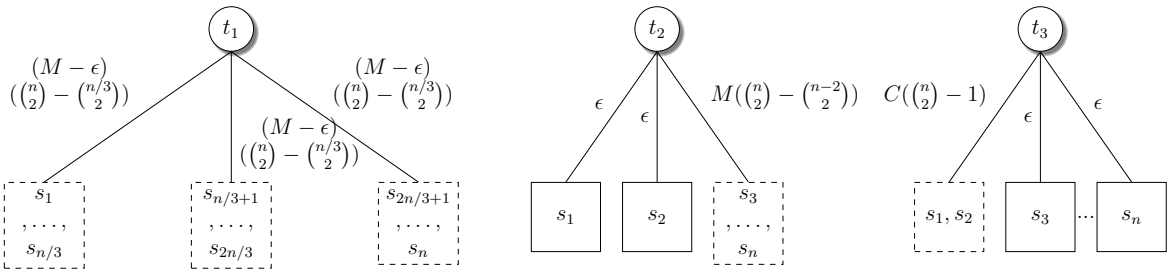

Figure 5.1: Subsets created by tests t_1, t_2 and t_3. Squares indicate leaf nodes and dashed squares indicate subsets with objects from different classes. Costs are indicated in the edges.

Consider the first iteration of DividePairs. The ratio obtained by (5-1) for t_1 is given by:

$$\max\left\{\frac{c^1(t_1)}{P(S)-P(S^1_{t_1})}, \frac{c^2(t_1)}{P(S)-P(S^2_{t_1})}, \frac{c^3(t_1)}{P(S)-P(S^3_{t_1})}\right\}$$

$$= \max\left\{\frac{(M-\epsilon)(\binom{n}{2}-\binom{n/3}{2})}{(\binom{n}{2}-\binom{n/3}{2})}, \frac{(M-\epsilon)(\binom{n}{2}-\binom{n/3}{2})}{(\binom{n}{2}-\binom{n/3}{2})}, \frac{(M-\epsilon)(\binom{n}{2}-\binom{n/3}{2})}{(\binom{n}{2}-\binom{n/3}{2})}\right\} = M-\epsilon$$

For test t_2, we have:

$$\max\left\{\frac{c^1(t_2)}{P(S)-P(S^1_{t_2})}, \frac{c^2(t_2)}{P(S)-P(S^2_{t_2})}, \frac{c^3(t_2)}{P(S)-P(S^3_{t_2})}\right\}$$

$$= \max\left\{\frac{\epsilon}{\binom{n}{2}}, \frac{\epsilon}{\binom{n}{2}}, \frac{M(\binom{n}{2}-\binom{n-2}{2})}{\binom{n}{2}-\binom{n-2}{2}}\right\} = M$$

Finally, for test t_3, we have:

$$\max\left\{\frac{c^1(t_3)}{P(S)-P(S^1_{t_3})}, \frac{c^2(t_3)}{P(S)-P(S^2_{t_3})}, \ldots, \frac{c^{n-1}(t_3)}{P(S)-P(S^{n-1}_{t_3})}\right\}$$

$$= \max\left\{\frac{C(\binom{n}{2}-1)}{\binom{n}{2}-1}, \frac{\epsilon}{\binom{n}{2}}, \ldots, \frac{\epsilon}{\binom{n}{2}}\right\} = C$$

Hence, DividePairs chooses test t_1 for the root of the tree, paying a cost of $(M-\epsilon)(\binom{n}{2}-\binom{n/3}{2}) = \Omega(n^2)$. But it is possible to build a decision tree which separates all pairs using only tests t_2 and t_3 paying a cost equal to $O(n)$. This tree is shown in Figure 5.2. We have to pay a cost of ϵ to reach s_1 and s_2 and a cost of $M(\binom{n}{2}-\binom{n-2}{2})+\epsilon = M(2n-3)+\epsilon$ to reach any other object in the tree. Thus, we reduced the cost by a factor of $\Omega(n)$.

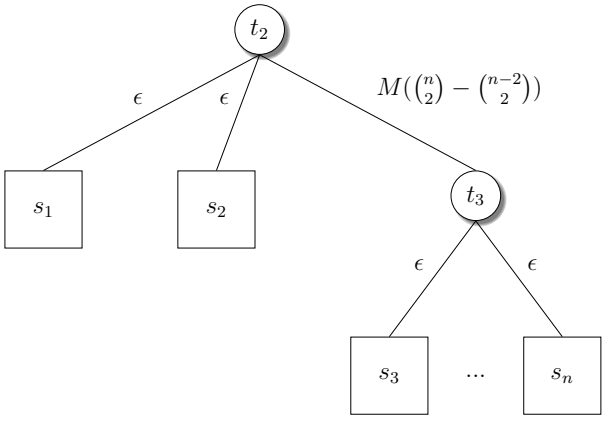

Figure 5.2: Decision tree with maximum cost equal to $O(n)$.

5.3 An n-approximation for Multiway Tests

Let $c^*(t) = \max\{c^1(t), c^2(t), \ldots, c^\ell(t)\}$. We can modify the greedy approach to simply select the test that minimizes $c^*(t)$ (and separates at least one pair of objects) at each step. Let τ be the first test selected by the new greedy algorithm and q be such that $c^q(\tau) + Cost(S^q_\tau) = \max\{c^1(\tau) + Cost(S^1_\tau), \ldots, c^\ell(\tau) + Cost(S^\ell_\tau)\}$. We have that:

$$\frac{Cost(S)}{OPT_W(S)} = \frac{c^q(\tau) + Cost(S^q_\tau)}{OPT_W(S)} \leq \frac{c^q(\tau)}{OPT_W(S)} + \frac{Cost(S^q_\tau)}{OPT_W(S^q_\tau)} \tag{5-11}$$

The second fraction can be eliminated using the same argument in (5-3), if $Cost(S^q_\tau) = 0$. Since the algorithm minimizes $c^*(t)$, we have that $OPT_W(S) \geq c^*(\tau) \geq c^q(\tau)$. From this and (5-11) we have:

$$\frac{Cost(S)}{OPT_W(S)} \leq 1 + \frac{Cost(S^q_\tau)}{OPT_W(S^q_\tau)} \tag{5-12}$$

By induction in the number of objects, we assume (for a set G) that $Cost(G)/OPT_W(G) \leq |G|$. For $n = 2$ (base case), the algorithm uses only one test (say, t). Let t^* be the test in the root of the optimal tree. We have that $Cost(G) = \max_i\{c^i(t)\} \leq \max_i\{c^i(t^*)\} = OPT_W(G)$. Therefore, $Cost(G)/OPT_W(G) \leq 1 < 2$. For the inductive step, we have:

$$\frac{Cost(S)}{OPT_W(S)} \leq 1 + |S^q_\tau| \leq |S| = n \tag{5-13}$$

Thus, the algorithm achieves an n-approximation.

5.4 Conclusions and Open Problems

In this Chapter, we presented a greedy algorithm for the minimization of the worst testing cost for the value dependent variation of the $DFEP$ (where we drop the assumption that the cost of each test is fixed). We prove that our algorithm is an $O(log(n))$ approximation for the case where all tests have two outcomes of binary tests. When a test can have more than two different answers we showed that a greedy strategy is not sufficient to provide an $O(log(n))$-approximation, presented an instance of the $DFEP$ for which the algorithm produces a tree with worst cost $\Omega(n)$ times worse than the optimal worst cost. Finally, we presented a second greedy algorithm that attains an $O(n)$-approximation for multiway tests.

A question which is left open is whether a logarithmic approximation can be guaranteed also for instances where tests are not restricted to be binary.

6
A randomized rounding algorithm for the $DFEP$ with bounded number of errors

In this Chapter, we study a modified version of the the $DFEP$ where the goal is to construct an oblivious decision tree that incurs in at most k classification errors, where k is a given integer. We present a randomized rounding approximation algorithm that, given a parameter $0 < \epsilon < 1/2$, builds an oblivious decision tree with cost at most $(3/(1-2\epsilon))log(n)OPT(I)$ and produces at most (k/ϵ) errors, where $OPT(I)$ is the optimal cost for an instance I. We recall that The logarithmic factor in the cost of the tree is the best possible attainable, even for $k = 0$, since the Set Cover problem reduces to our problem (see Section 3.3). For clarity purposes, we redefine an instance I of the $DFEP$, with some minor modifications.

An instance is defined as a $5-$tuple $I = (S, C, T, c, k)$ 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, $\mathbf{c}$ is a cost function and k is an integer. A test $t_i \in T$, when applied to an object $s \in S$, outputs a number $t_i(s)$ in the set $\{1, \ldots, o\}$. The cost function $\mathbf{c}$ assigns to each test t_i a cost $c(t_i) \in \mathbb{R}^+$.

We note that here each test has, again, a fixed cost, and thus we use the same notation of the previous Chapters where the cost is independent of the output of the test.

6.1
A Randomized Rounding Approximation Algorithm

In this Section, we present an integer program formulation for the $DFEP$ and a randomized rounding approach to get a logarithmic approximation. For convenience, we say that for any pair (s_i, s_j) of objects we define T^{ij} as the set of tests that separate s_i and s_j).

Our integer programming model has the following variables: for each test $t_i \in T$ there is a corresponding variable x_i equal to 1 if t_i is in the solution (decision tree) and 0 otherwise. We also use a variable y_i for each object $s_i \in S$, and we say that $y_i = 1$ if object s_i **can** receive the wrong classification. Thus, if $y_i = 1$, s_i can be put in **any** class, including the correct one. We will show that the ODTBE is equivalent to the ILP presented below.

$$\begin{aligned}
&\text{minimize} \quad \sum_{i=1}^{|T|} c(t_i)x_i \\
&\text{s. t.} \\
&\qquad \sum_{\ell:t_\ell \in T^{ij}} x_\ell \quad \geq 1 - y_i - y_j, \quad \text{for each pair } (s_i, s_j) \text{ of objects} \\
&\qquad \sum_{j=1}^{n} y_j = k \\
&\qquad x_i \in \{0,1\}, i = 1, ..., |T| \\
&\qquad y_i \in \{0,1\}, i = 1, ..., n
\end{aligned} \tag{6-1}$$

We have to prove that any feasible solution (oblivious decision tree) for the ODTBE problem leads to a feasible solution of the ILP with the same cost, and any feasible solution for the ILP leads to a feasible decision tree with the same cost. Note that a tree is feasible if: (i) it incurs in at most k classification errors and (ii) it separates all pairs of objects that receive different classifications.

Lemma 7 *If D is a feasible decision tree with cost C, then it corresponds to at least one feasible solution of the ILP with cost C.*

Proof: Let D be a feasible decision tree that incurs in at most k classification errors and let $k' \leq k$ be the number of objects that end in a leaf node with the wrong classification. By the definition of x_i, we set $x_i = 1$ if test t_i is in the tree and $x_i = 0$ otherwise. We first set $y_i = 1$ for all k' objects incorrectly classified by the tree, and then choose any subset[1] of size $k - k'$ from the remaining objects and set their correspondent y_i values to 1. Finally, we set all the remaining y_i values to 0. Clearly, $\sum_{i=1}^{n} y_i = k$. Consider now the following constraint:

$$\sum_{\ell:t_\ell \in T^{ij}} x_\ell \geq 1 - y_i - y_j \tag{6-2}$$

We have to show that this constraint is not violated for any pair of objects. Let (s_a, s_b) be a pair of objects. (note that by the definition of a pair, s_a and s_b belong to different classes). We have the following cases:

Case 1: $y_a = y_b = 1$. In this case, $\sum_{\ell:t_\ell \in T^{ab}} x_\ell \geq -1$, which is true since $x_i \in \{0,1\}$.

Case 2: $y_a = 1$ and $y_b = 0$ (or $y_a = 0$ and $y_b = 1$). We have that $\sum_{\ell:t_\ell \in T^{ab}} x_\ell \geq 0$, which also trivially holds.

[1]Note that any possible choice will lead to a solution with cost C.

Case 3: $y_a = y_b = 0$. By our choice of the y variables, $y_i = 0$ implies that s_i was correctly classified. Since s_a and s_b come from different classes, there is at least one test t_i that separates s_a and s_b in the tree, or otherwise they would end up in the same leaf node. Thus, $\sum_{\ell:t_\ell \in T^{ab}} x_\ell \geq 1 - y_a - y_b$.

■

Lemma 8 *If (x, y) is a feasible solution for the ILP with cost C', then we can construct a feasible decision tree D with the same cost corresponding to this solution.*

Proof: First, we construct a decision tree by choosing the tests t_i for which $x_i = 1$, which will split the objects in several subsets (which will be the leaf nodes). Clearly, the cost of this tree is equal to C'. In order to complete the proof we have to show that there is a choice of classes for these subsets that is consistent with the definition of the y values (i. e., if $y_i = 0$ then object s_i have to receive its original class). First, it is easy to see that any pair (s_a, s_b) of objects from different classes with $y_a = y_b = 0$ will end up in different leaf nodes. This is ensured by the constraint:

$$\sum_{\ell:t_\ell \in T^{ij}} x_\ell \geq 1 - y_i - y_j \tag{6-3}$$

Thus, if two objects from different classes **have to** receive their correct classes, then they will end up in different leaf nodes.

For each leaf node having an object s_i such that $y_i = 0$, set the class of this leaf as the class of s_i. Since the other objects in this leaf have their y values equal to 1 or belong to the same class of s_i, this step will not violate the condition of the y values.

Finally, for the remaining leaf nodes, we can choose any classification (since all elements in these leaf nodes have their y values equal to one). By construction, all objects s_i with $y_i = 0$ will receive a correct classification, and therefore the number of misclassified objects in the tree will be bounded above by k, because there are at most k objects in these leaves.

■

We can obtain an approximation algorithm to solve the ODTBE, using the ILP formulation (6-1). Consider the procedure presented in Algorithm 4.

Algorithm 4 Approximation algorithm for the ODTBE

Input: A 5-tuple $I = (S, T, C, \mathbf{c}, k)$ and a parameter $0 < \epsilon < 1/2$

Output: An oblivious decision tree for I

1. Solve the linear relaxation of the integer program. Let (x^*, y^*) be its optimal solution.

2. Set $y_i = 0$ if $y_i^* < \epsilon$ and $y_i = 1$ if $y_i^* \geq \epsilon$.

3. For every constraint of the type $\sum_{\ell:t_\ell \in T^{ij}} x_l \geq 1 - y_i - y_j$ with $y_i = 0$ and $y_j = 0$, pick each $t_l \in T^{ij}$ with probability x_ℓ^*. Repeat this step $\left(\frac{3}{1-2\epsilon}\right) log(n)$ times and take the union of the selected tests.

4. Build an oblivious decision tree using the tests selected in Step 3 in the following manner: for each subset S' of objects that ended up in the same leaf node:

   4.1 If there are two objects s_i and s_j in S', from different classes, with $y_i = y_j = 0$ then **Return** FAIL;

   4.2 If every object $s_i \in S'$, with $y_i = 0$, belong to the same class, define the class of the leaf node to be the class of s_i.

   4.3 If $y_i = 1$ for all $s_i \in S'$, define the leaf node class to be the one which incurs in the fewest number of classification errors.

As we can see in Step 4.1, the algorithm may fail. We will show that this event occurs with low probability.

Lemma 9 *The probability of Algorithm 4 failing is at most* $(n-1)/2n^2$.

Proof: Let (s_i, s_j) be a pair of objects such that $y_i = y_j = 0$. We have that $y_i^* < \epsilon$ and $y_j^* < \epsilon$. Thus, $\sum_{\ell:t_\ell \in T^{ij}} x_\ell^* \geq 1 - y_i^* - y_j^* \geq 1 - 2\epsilon$. In a single draw of step *3* of Algorithm 4, the pair (s_i, s_j) is **not** separated with probability:

$$Pr[(s_i, s_j) \text{ is not separated}] = \prod_{\ell:t_\ell \in T^{ij}} \left(1 - x_\ell^*\right) \leq \prod_{\ell:t_\ell \in T^{ij}} \frac{1}{e^{x_\ell^*}} \leq \frac{1}{e^{1-2\epsilon}}$$

Thus, if we pick the union of $\big(3/(1-2\epsilon)\big)\log(n)$ such sets, the probability that the pair (s_i, s_j) remain together is bounded above by $\frac{1}{e^{3\log(n)}} = 1/n^3$. Since we have at most $n(n-1)/2$ pairs of objects from different classes, with

probability $1-(n-1)/2n^2$ we produce a tree that separates all pairs of objects from different classes that have y values equal to 0.

■

The next lemma bounds the number of misclassifications.

Lemma 10 *Algorithm 4 produces a tree with at most* $\frac{k}{\epsilon}$ *classification errors.*

Proof: Consider the ratio y_i/y_i^* for each object s_i. If $y_i^* < \epsilon$, y_i is set to 0 and the error does not increase compared with the linear relaxation. If $y_i^* \geq \epsilon$, we have that $y_i \leq \left(\frac{1}{\epsilon}\right) y_i^*$, which leads to at most $\frac{k}{\epsilon}$ errors. ■

Finally, we calculate the expected cost of our solution.

Lemma 11 *Algorithm 4 produces a tree with an* $O(log(n))$ *approximation.*

Proof: Let C be the expected cost of our solution. We have that:

$$E[C] = \sum_{i=1}^{|T|} c(t) Pr[x_t = 1] = \left(\frac{3}{1-2\epsilon}\log(n)\right) \sum_{i=1}^{|T|} c(t_i) x_i^* = O(\log(n)) OPT(I)$$

■

Putting together the previous lemmas we get our main result.

Theorem 10 *Let* $I = (S, C, T, \mathbf{c}, k)$ *be an instance of ODTBE, with* $|S| = n$*, and let* $0 < \epsilon < 1/2$*. Then, there exists a polynomial time algorithm for ODTBE that, with probability at least* $1 - (n-1)/2n^2$*, builds an oblivious decision tree with cost* $O(\log(n) OPT(I))$ *that misclassifies at most* k/ϵ *samples, where* $OPT(I)$ *is the cost of the optimal solution for* I*.*

6.2 Conclusions and Open Problems

In this Chapter, we presented a randomized rounding algorithm that, given a modified instance of the problem with an additional integer $k \geq 0$, and given a parameter $0 < \epsilon < 1/2$, builds an oblivious decision tree with cost at most $(3/(1-2\epsilon))ln(n)OPT(I)$ and produces at most (k/ϵ) errors, where $OPT(I)$ denotes the cost of the oblivious decision tree with minimum cost among all oblivious decision trees for instance I that make at most k classification errors.

Finally, we can ask two open questions. The first one is if it is still possible to achieve a logarithmic factor without violating the error constraint. The second one is if we can extend the integer linear programming model to solve the general problem, where we drop the constraint that requires an oblivious decision tree.

7
Conclusions

In this Thesis, we presented several algorithms so solve different variants of the $DFEP$. We now summarize our main results and open problems.

Our first contribution is an algorithm for the $DFEP$ that achieves an $O(log(n))$ approximation with respect to both the expected testing cost and the worst testing cost, simultaneously, for the classical version of the problem where each test has a fixed cost and no classification errors are allowed. Our result closes the gap left open by the previous $O(\log 1/p_{\min})$ approximation for the expected testing cost shown by Golovin et al. (2010) and Bellala et al. (2012). We recall that our result is the best possible approximation achievable with respect to either optimization measure, under the assumption that $\mathcal{P} \neq \mathcal{NP}$).

Our second contribution is a complete characterization of the best possible trade-off achievable when optimizing the construction of a decision tree with respect to both the worst and the expected cost. We again considered fixed testing costs and no classification errors. We showed that for every $\rho > 0$ there is a decision tree D with worst testing cost at most $(1+\rho)OPT_W$ and expected testing cost at most $\frac{1}{1-e^{-\rho}}OPT_E$, where OPT_W and OPT_E denote the minimum worst testing cost and the minimum expected testing cost of a decision tree for the given instance. We prove that these bounds are sharp by showing that there are infinitely many instances for which we cannot obtain a decision tree with both worst cost smaller than $(1+\rho)OPT_W(I)$ and expected cost smaller than $(1/(1-e^{-\rho}))OPT_E(I)$. Moreover, we extended the construction of the lower bound considering only uniform probabilities, showing that the limit on the best trade-off achievable also holds for this case. As an open problem, we can ask whether for every $\epsilon > 0$, there is some integer n_0 such that every instance I with uniform testing costs and with more than n_0 objects, admits a decision tree D such that $cost_E(D) \leq (1+\epsilon)OPT_E(I)$ and $cost_W(D) \leq (1+\epsilon)OPT_W(I)$.

Our third contribution was a greedy algorithm for the minimization of the worst testing cost for the value dependent variation of the $DFEP$ (where we drop the assumption that the cost of each test is fixed). We prove that our algorithm is an $O(log(n))$ approximation for the case where all tests have two outcomes of binary tests. When a test can have more than two different answers we showed that a greedy strategy is not sufficient to provide

an $O(log(n))$-approximation, presented an instance of the $DFEP$ for which the algorithm produces a tree with worst cost $\Omega(n)$ times worse than the optimal worst cost. Finally, we presented a second greedy algorithm that attains an $O(n)$-approximation for multiway tests. An open problem is whether a logarithmic approximation can be guaranteed also for instances where tests are not restricted to be binary.

Our fourth and last contribution was a randomized rounding algorithm that, given a modified instance of the problem with an additional integer $k \geq 0$, and given a parameter $0 < \epsilon < 1/2$, builds an oblivious decision tree with cost at most $(3/(1-2\epsilon))ln(n)OPT(I)$ and produces at most (k/ϵ) errors, where $OPT(I)$ denotes the cost of the oblivious decision tree with minimum cost among all oblivious decision trees for instance I that make at most k classification errors. The open questions related to this problem are if it is still possible to achieve a logarithmic factor without violating the error constraint and if we can extend our model to non-oblivious decision trees.

Bibliography

Adler, M. & Heeringa, B. (2008), Approximating optimal binary decision trees, APPROX '08 / RANDOM '08, pp. 1–9.

Arkin, E. M., Meijer, H., Mitchell, J. S. B., Rappaport, D. & Skiena, S. S. (1993), Decision trees for geometric models, *in* 'Proceedings of the ninth annual symposium on Computational geometry', SCG '93, pp. 369–378.

Aslam, J., Rasala, A., Stein, C. & Young, N. (1999), Improved bicriteria existence theorems for scheduling, *in* 'Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, pp. 846–847.

Bazaraa, M. S., Sherali, H. D. & Shetty, C. M. (1993), *Nonlinear programming: theory and algorithms*, John Wiley & Sons.

Bellala, G., Bhavnani, S. K. & Scott, C. (2012), 'Group-based active query selection for rapid diagnosis in time-critical situations', *IEEE Trans. Inf. Theor.* **58**(1), 459–478.

Buro, M. (1993), 'On the maximum length of huffman codes', *Information processing letters* **45**(5), 219–223.

Chakaravarthy, V., Pandit, V., Roy, S., Awasthi, P. & Mohania, M. (2007), Decision trees for entity identification: Approximation algorithms and hardness results, *in* 'PODS', pp. 53–62. DOI `http://doi.acm.org/10.1145/1265530.1265538`.

Chakaravarthy, V. T., Pandit, V., Roy, S. & Sabharwal, Y. (2009), Approximating decision trees with multiway branches, *in* 'Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I', ICALP '09, pp. 210–221.

Cicalese, F., Jacobs, T., Laber, E. & Molinaro, M. (2010), 'On greedy algorithms for decision trees', *Algorithms and Computation* pp. 206–217.

Cormen, T. H. (2009), *Introduction to algorithms*, MIT press.

Feige, U. (1998), 'A threshold of ln n for approximating set cover', *Journal of the ACM (JACM)* **45**(4), 634–652.

Gandhi, R., Khuller, S. & Srinivasan, A. (2001), Approximation algorithms for partial covering problems, *in* 'International Colloquium on Automata, Languages, and Programming', Springer, pp. 225–236.

Garey, M. R. (1974), 'Optimal binary search trees with restricted maximal depth', *SIAM Journal on Computing* **3**(2), 101–110.

Golovin, D., Krause, A. & Ray, D. (2010), Near-optimal bayesian active learning with noisy observations, *in* J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel & A. Culotta, eds, 'NIPS', Curran Associates, Inc, pp. 766–774. URL `http://books.nips.cc/papers/files/nips23/NIPS2010_1100.pdf`.

Guillory, A. & Bilmes, J. (2009), Average-case active learning with costs, *in* 'Proceedings of the 20th international conference on Algorithmic learning theory', ALT'09, pp. 141–155.

Gupta, A., Nagarajan, V. & Ravi, R. (2010), Approximation algorithms for optimal decision trees and adaptive tsp problems, *in* 'Proceedings of the 37th international colloquium conference on Automata, languages and programming', ICALP'10, pp. 690–701.

Hanneke, S. (2006), 'The cost complexity of interactive learning', *unpublished*.

Hyafil, L. & Rivest, R. L. (1976), 'Constructing optimal binary decision trees is np-complete', *Inf. Process. Lett.* **5**(1), 15–17.

Kohavi, R. & Li, C.-H. (1995), Oblivious decision trees, graphs, and top-down pruning, *in* 'IJCAI', Citeseer, pp. 1071–1079.

Kosaraju, Przytycka & Borgstrom (1999), On an optimal split tree problem, *in* 'WADS: 6th Workshop on Algorithms and Data Structures'.

Laber, E. S. & Nogueira, L. T. (2004), 'On the hardness of the minimum height decision tree problem', *Discrete Applied Mathematics* **144**(1), 209–212.

Langley, P. & Sage, S. (1994), Oblivious decision trees and abstract cases, *in* 'Working notes of the AAAI-94 workshop on case-based reasoning', Seattle, WA, pp. 113–117.

Larmore, L. L. (1987), 'Height restricted optimal binary trees', *SIAM Journal on Computing* **16**(6), 1115–1123.

Larmore, L. L. & Hirschberg, D. S. (1990), 'A fast algorithm for optimal length-limited huffman codes', *Journal of the ACM (JACM)* **37**(3), 464–473.

Milidiú, R. L. & Laber, E. S. (2001), 'Bounding the inefficiency of length-restricted prefix codes', *Algorithmica* **31**(4), 513–529.

Moshkov, M. (2003), 'Approximate algorithm for minimization of decision tree depth', *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing* pp. 579–579.

Rasala, A., Stein, C., Torng, E. & Uthaisombut, P. (2002), Existence theorems, lower bounds and algorithms for scheduling to meet two objectives, *in* 'Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms', Society for Industrial and Applied Mathematics, pp. 723–731.

Raz, R. & Safra, S. (1997), A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np, *in* 'Proceedings of the twenty-ninth annual ACM symposium on Theory of computing', ACM, pp. 475–484.

Saettler, A. M. (2013), 'On the simultaneous minimization of worst testing cost and expected testing cost with decision trees', *Master Thesis*.

Sattler, K.-U. & Dunemann, O. (2001), Sql database primitives for decision tree classifiers, *in* 'Proceedings of the tenth international conference on Information and knowledge management', ACM, pp. 379–386.

Schlimmer, J. C. et al. (1993), Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning., *in* 'ICML', Citeseer, pp. 284–290.

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

Wolsey, L. A. (1982*a*), 'An analysis of the greedy algorithm for the submodular set covering problem', *Combinatorica* **2**(4), 385–393.

Wolsey, L. A. (1982*b*), 'Maximising real-valued submodular functions: Primal and dual heuristics for location problems', *Mathematics of Operations Research* **7**(3), 410–425.

A
Proofs for Chapter 3

A.1
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-tuple $(S, C, \mathcal{X}, \mathbf{p}, \mathbf{c})$, 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:

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

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` presented in (and henceforth referred to as) Algorithm 5 when it is executed on the instance (I_P, B). This algorithm corresponds to lines 5,6,10 and 11 of the `While` loop of Algorithm 2. In Algorithm 5, the greedy criterion 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.

Algorithm 5

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

1: $seq \leftarrow \emptyset$, $U \leftarrow S$, $k \leftarrow 1$
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*, $U \leftarrow U \backslash X_k$, $B \leftarrow B - c(X_k)$, $\mathcal{X} \leftarrow \mathcal{X} \backslash \{X_k\}$, $k \leftarrow k+1$
5: **end while**

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

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 within 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 $sepcost_B(I_P, \mathbf{X}^A) \leq 2 sepcost(I', \mathbf{Z})$.

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. The set of classes remains the same, i.e., $C' = C$. 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 include θ. Note that

$$|F(\theta)| = \left(\prod_{Y \in \mathcal{X}_s} n(Y) \right) / n(X). \tag{A-2}$$

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 5 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

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

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

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 $2B \geq 2^\ell = i^*_{\ell-1}$, 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:

$$\begin{aligned} sepcost(I', \mathbf{X}^*) &= \sum_{i=1}^{q^*} c(X^*_i)(Q - p^*_{i-1}) && \text{(A-3)} \\ &\geq \sum_{i=1}^{i^*_0}(1/2)\cdot(Q - p^*_{i-1}) \\ &\quad + \sum_{j=1}^{\ell-1}\sum_{k=i^*_{j-1}+1}^{i^*_j}(1/2)\cdot(Q - p^*_{k-1}) && \text{(A-4)} \\ &\geq \frac{2Q - p^*_1}{2} + \sum_{j=1}^{\ell-1}(2^j - 2^{j-1})(Q - P^{[j]}_*) && \text{(A-5)} \\ &\geq \frac{3Q}{4} + \frac{1}{2}\sum_{j=1}^{\ell-1}2^j(Q - P^{[j]}_*) && \text{(A-6)} \end{aligned}$$

The inequality in (A-4) follows from (A-3) by considering in the summation on the right hand side of (A-4) only the first $i^*_{\ell-1} = 2^\ell \leq B \leq q^*$ tests.

The term $(2Q - p^*_1)/2$ in the inequality (A-5) is the contribution of the first two tests of the sequence $\mathbf{X}^*$ to the separation cost. To prove that $\frac{2Q-p^*_1}{2} \geq \frac{3Q}{4}$, yielding (A-6), we note that that $p^*_1 \leq Q/2$ because the probability covered by the first test X^*_1 of sequence $\mathbf{X}^*$ is $p(X)/n(X) \leq Q/2c(X) \leq Q/2$, where

X is the test that generates X_1^*. In the last inequality we used the fact that $c(X) \geq 1$ 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 $l \geq k+1$, 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

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

where the last inequality follows from equation A-6. 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 5) when it is executed on instance (I_P, B).

Claim 3. There exists an execution of procedure `GreedyPSP` (Algorithm 5) 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 $\mathbf{X}^\mathbf{A}$ that is not the test which maximizes $p(U \cap X)/c(X)$ among all the tests in $\mathcal{X}$ in line (*) of Algorithm 5. Note that X_{r+1}^A is chosen by Algorithm 5 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

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

Thus, to establish the claim it suffices to show that

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

where $\mathbf{Z}$ is a possible output of `GreedyPSP` (Algorithm 5) 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 A.2.

Proposition 8 *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* H *in* $\mathcal{X}'$, *with* X *being the test in* $\mathcal{X}$ *from which* H *is generated, it holds that*

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

(iii) $\mathbf{Z}$ *is a feasible output for* `GreedyPSP` *(Algorithm 5) 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 $\mathbf{X}^A$, we can write $sepcost_B(I_P, X_1^A, \ldots, X_r^A)$ as follows

$$sepcost_B(I_P, \langle X_1^A, \ldots, X_r^A \rangle) = \sum_{j=1}^{r} C_j \cdot p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right) + B \cdot p\left(S - \bigcup_{j=1}^{r} X_j^A\right) \quad \text{(A-7)}$$

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

$$\begin{aligned} sepcost(I', \mathbf{Z}) &= \sum_{j=1}^{r} \sum_{\kappa=1}^{n(X_j^A)} p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)\left(C_{j-1} + \kappa \cdot \frac{1}{2}\right) \\ &\quad + \sum_{\kappa=1}^{2z} p\left(Z_\kappa^{[r+1]} - \left(\bigcup_{i=1}^{r} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[r+1]}\right)\right)\left(C_r + \kappa \cdot \frac{1}{2}\right) \\ &\quad + B \cdot p\left(S' - \left(\bigcup_{i=1}^{r} \bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa=1}^{2z} Z_\kappa^{[r+1]}\right)\right), \end{aligned} \tag{A-8}$$

where we have split $\mathbf{Z}$ 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 $j = 1, \ldots, r$ and $1 \leq \kappa \leq n(X_j^A)$, and also for $j = r+1$ and $\kappa \leq 2z$, 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 8 (i) we have that $Z_\kappa^{[j]} \cap \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right) = \emptyset$.

Moreover, by Proposition 8 (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 (A-8) can be spilt into the objects covered by 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 (A-8) as follows

$$\begin{aligned} sepcost(I', \mathbf{Z}) \;=\; & \sum_{j=1}^{r} \sum_{\kappa=1}^{n(X_j^A)} \frac{p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right)}{n(X_j^A)} \left(C_{j-1} + \kappa \cdot \frac{1}{2}\right) \\ & + \sum_{\kappa=1}^{2z} \frac{p\left(Y - \bigcup_{i=1}^{r} X_i^A\right)}{n(Y)} \left(C_r + \kappa \cdot \frac{1}{2}\right) \\ & + \frac{n(Y) - 2z}{n(Y)} p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \cdot B \\ & + p\left(S - Y - \bigcup_{j=1}^{r} X_j^A\right) B \end{aligned} \tag{A-9}$$

Via simple algebraic manipulation on the first term in the right hand side of (A-9) 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 (A-9) 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

$$\begin{aligned} sepcost(I', \mathbf{Z}) \;\geq\; & \sum_{j=1}^{r} p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right) \left(C_{j-1} + \frac{c(X_j^A)}{2}\right) + p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \frac{2z}{n(Y)} \frac{B + C_r}{2} \\ & + \frac{n(Y) - 2z}{n(Y)} p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \cdot B + p\left(S - Y - \bigcup_{j=1}^{r} X_j^A\right) B \end{aligned} \tag{A-10}$$

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 (A-10) can be lower bounded with $p(Y - \bigcup_{j=1}^{r} X_j^A) \cdot B/2$ and we get

$$sepcost(I', \mathbf{Z}) \geq \sum_{j=1}^{r} p\left(X_j^A - \bigcup_{i=1}^{j-1} X_i^A\right) \frac{C_j}{2} + p\left(Y - \bigcup_{i=1}^{r} X_i^A\right) \frac{B}{2} + p(S - Y - \bigcup_{j=1}^{r} X_j^A)B \tag{A-11}$$

Putting together (A-11) and (A-7) we have the desired result

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

It remains to argue about the case where X_{r+1}^A does not exist, which means that all tests that maximize the greedy criterion in Algorithm 5 have cost smaller than the current budget B. In this case, the analysis becomes simpler and can be easily handled in the same way as above. In fact, the only difference is that the last term in (A-7) disappears, as do all the terms referring to Y and $\mathbf{Z}^{[r+1]}$.

The lemma follows from the correctness of the three claims. ■

A.2
The Proof of Proposition 8

Proposition 8. *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* H *in* $\mathcal{X}'$, *with* X *being the test in* $\mathcal{X}$ *from which* H *is generated, it holds that*

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

(iii) $\mathbf{Z}$ *is a feasible output for* `GreedyPSP` *(Algorithm 5) 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 $H - W$ are precisely the elements of H 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 H that are generated from s, and each one of them has probability $p(s)/N$. Hence we have

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

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

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

Claim. for each $j = 1, \ldots, r$ and $1 \leq \kappa \leq n(X_j^A)$, and also for $j = r+1$ and $\kappa \leq 2z$, we have that

$$\frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)}{c(Z_\kappa^{[j]})} \geq \frac{p\left(H - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)}{c(H)} \tag{A-12}$$

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

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 criterion 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 (A-12), and X be the test in $\mathcal{X}$ from which H is generated. Then we have

$$\begin{aligned}
R &\leq \frac{p\left(H - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)}{c(H)} && \text{(A-13)}\\
&= \frac{p(X - \bigcup_{i=1}^{j-1} X_i^A)}{c(X)} && \text{(A-14)}\\
&\leq \frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)}{c(Z_\kappa^{[j]})} && \text{(A-15)}\\
&= \frac{p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right)}{c(Z_\kappa^{[j]})} && \text{(A-16)}
\end{aligned}$$

Inequality (A-13) holds since the set whose probability is considered at the numerator of the right hand side of (A-13) is a superset of the set whose probability is considered at the numerator of the right hand side of (A-12).

Inequality (A-14) follows from (A-13) by property (ii) above.

In order to prove (A-15) 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[1] 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, (A-16) follows from (A-15) 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,
$p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right) - \left(\bigcup_{\kappa'=1}^{\kappa-1} Z_{\kappa'}^{[j]}\right)\right) = p\left(Z_\kappa^{[j]} - \left(\bigcup_{i=1}^{j-1}\bigcup_{\kappa'=1}^{n(X_i^A)} Z_{\kappa'}^{[i]}\right)\right)$. ■

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

B
Proofs for Chapter 4

B.1
Calculations for $\mathbf{p}^*$ in Lemma 4

In order to verify that $\mathbf{p}^* \in \mathcal{P}$ it suffices to use the fact that $p_0^*, \ldots, p_C^*$ form a geometric progression with ratio $(W-1)/W$.

To show that

$$f(\mathbf{p}^*) = \frac{1}{1 - \frac{(W-1)^{C+1}}{W^{C+1}}}$$

it is enough to use the following identities

$$\sum_{i=1}^{C+1} i \cdot p_i^* = W - (C+W)\frac{(W-1)^C}{W^C} + (C+1)\frac{(W-1)^C}{W^C}$$

$$\sum_{i=1}^{j} i \cdot p_i^* = W - (j+W)\frac{(W-1)^j}{W^j}$$

$$\sum_{i=j+1}^{C+1} p_i^* = \frac{(W-1)^j}{W^j}$$

B.2
Calculations with the dual solution in Lemma 4

In this section we prove that $\lambda^* = (\lambda_E^*, \lambda_0^*, \lambda_1^*, \ldots, \lambda_C^*)$ is a feasible solution for the dual problem given by Equations (4-10)- (4-13).

To verify that constraint (4-12) is satisfied we use the fact that $\lambda_0^*, \lambda_1^*, \ldots, \lambda_C^*$ form a geometric sequence. In addition, to verify that constraint (4-11) holds we use the following identities:

$$W\sum_{j=0}^{k-1} \lambda_j^* = \frac{(W-1)^{C+1-k} \cdot W^{k+1} - W(W-1)^{C+1}}{W^{C+1} - (W-1)^{C+1}}$$

$$k\sum_{j=k}^{C} \lambda_j^* = \frac{k \cdot W^{C+1} - kW^k(W-1)^{C-k+1}}{W^{C+1} - (W-1)^{C+1}},$$

$$\sum_{j=0}^{k-1} j\lambda_j^* = \frac{(k-1)W^k \cdot (W-1)^{C-k+1} - W^k(W-1)^{C-k+2} + W(W-1)^{C+1}}{W^{C+1} - (W-1)^{C+1}}$$

and

$$k\lambda_E^* = \frac{kW^{C+1}}{W^{C+1} - (W-1)^{C+1}}$$