A LINEAR PROGRAMMING MODEL FOR SEQUENTIAL TESTING

BY LILIYA FEDZHORA

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Endre Boros

and approved by

New Brunswick, New Jersey

October, 2008

ABSTRACT OF THE DISSERTATION

A LINEAR PROGRAMMING MODEL FOR SEQUENTIAL TESTING

by LILIYA FEDZHORA

Dissertation Director: Endre Boros

In this study, a linear programming model is formulated that finds an optimal strategy for many decision-making problems that typically arise in homeland security, banking, medicine, and engineering. We consider the problem of deploying a set of tests most effectively when the goal is to detect as many as possible "bad" objects among the vast majority of "good" ones, for example, in searching for contraband. The study assumes that functional dependency between test results and object type is unknown. The model finds an optimal testing strategy in the form of a decision tree that minimizes the expected cost given a detection rate or maximizes the detection rate given the budget. The mathematical basis for the model is a polyhedral description of all decision trees in higher dimensional space.

Decision trees are widely used in data mining and machine learning. A notion of VC-dimension is used to evaluate the bounds of the sample size required for the learning model. For some classes of Boolean functions VC-dimension is already known, for example, for monomials and threshold functions. In Chapter 10, the VC-dimension of Horn functions is derived, and also the VC-dimension of a more general class of k-quasi-Horn functions. In Chapter 11 we state and prove a criterion for k-quasi-Horn functions that generalizes McKinsey's theorem. Also, necessary and sufficient conditions for function to be bidual k-quasi-Horn are stated and proved.

Acknowledgements

I thank Endre Boros, my dissertation adviser, who guided my progress and taught me so much about research. He was always there for help and advice, and I am grateful to him for his invaluable support and encouragement.

I shall always be grateful to the late Peter Hammer[1] for his trust and his support. He helped me on many different fronts, both within and outside the realm of my studies. We shared many interesting and entertaining discussions on a wide range of topics.

I thank Bela Vizvari, Andras Prekopa, Paul Kantor, and Vladimir Gurvich for serving on my dissertation committee and for providing many valuable comments and suggestions on this dissertation.

I thank Terry Hart, Clare Smietana, Lynn Agree, and Jacquelyn Thompson, who contributed so much help and advice.

I thank my parents, my brother, and my extended family in Philadelphia for their love and support throughout my entire education.

Finally, I thank my husband Oleg, who has shared the ups and downs with me throughout this journey, and I am eternally grateful to him.

[1]Peter Hammer (1936-2006) passed away on December 27, 2006, in a tragic car accident.

Dedication

To my husband

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Should a bank grant a credit card to a particular applicant? Does a cargo ship that arrives at a U.S. port from a foreign country carry containers with illegal contents? Has an emergency room patient suffered a heart attack? Decision-making or diagnostic problems arise in many areas, including astronomy ([70]), engineering ([21]), image and language processing ([18, 52, 67]), homeland security ([71, 69, 2, 56]), manufacturing and production ([28, 33, 34, 36]), and medicine ([35, 45, 48, 53, 57]). Some real-life examples are provided in Chapter 8. Diagnoses are based on test results, such as blood pressure readings or radiation levels. For some problems, functional dependency between tests results and diagnosis (decision function) is known, but for others it is not.

We consider the problem of deploying a set of tests most effectively when the goal is to detect as many as possible "bad" objects among the vast majority of "good" ones, for example, in detecting contraband or diagnosing patients with rare diseases. (For the purpose of focusing on diagnostic problems, we will refer to the subjects being analyzed as "objects", whether they are individuals or inanimate entities.) Tests have costs in terms of such factors as time, money, and pain. Making decisions about the object type also incurs costs. When we decide that an object is good based on the test results, the costs of that decision are usually negligible; for example, if a diagnosis is favorable, or good, a doctor releases a healthy patient. In most cases, the decision that an object is bad costs considerably more than the tests; for example, a diagnosis is unfavorable, or bad, so a sick patient checks into the hospital and undergoes a costly treatment.

Thus, an object is routed through various tests according to the results it produces. This process can be conceptualized by saying that a particular design of the testing schedule produces a tree. The first test to be applied is at the root node. According to the initial root-node test results, the object being tested may be either released (i.e., declared good), or forwarded to another test, or declared bad. When test results are expensive to learn we can use given labeled data to derive an empirical distribution for each test and type.

One of the most important applications of the diagnosis problem is in homeland security. Because the vast majority of imported goods come to the United States by sea, inspecting containers at U.S. ports to prevent illegal items from entering the country is critical. Obviously, inspecting every container is economically infeasible, and only about 2% to 3% of the containers are inspected each year (see, e.g., [54] for more details).

In [71], the container inspection problem is approached by completely enumerating all possible binary decision trees. One of the two assumptions that is made about a decision function is completeness; that is, in some cases, to decide whether a container is dangerous, the readings from all sensors (tests) are required. The second assumption is monotonicity; that is, if the readings of any two containers at all the sensors are the same except one, then the container with the more distinctively suspicious reading is likely to be more dangerous. Even under these assumptions, the number of binary decision trees grows doubly exponentially in the number of sensors, and the problem becomes computationally challenging for more than four sensors. (More definitions are provided in Chapter 3, and a more detailed description of this model can be found in Chapter 4.)

Sensitivity analysis of the results of the above model is described in [2]. To make the model more tractable, at least two (opposite by nature) approaches have been considered in the literature. In [82] more restrictive assumptions are made about the binary decision trees. In [56] a broader class of binary decision trees is discussed. The notion of a class of complete, monotonic Boolean functions is generalized and a notion of a class of complete and monotonic binary decision trees is defined. A search algorithm based on [22, 23, 59] is introduced that allows us to find an optimum decision tree for at least five sensors.

In Chapter 5 we introduce some additional definitions, and in Chapters 6 and 7 we formulate a linear programming model that finds an optimal strategy for problems with unknown diagnosis function. The model finds an optimal strategy in the form of a decision tree that minimizes the expected cost given the detection rate or maximizes the detection rate given the budget.

While most of the literature considers the simplest type of decision trees, binary, our model suggest mapping the problem from the space of all decision trees to the space of all ordered test sequences. This approach proves to be very efficient, because the dimension of the space of all decision trees is $O(n!M(n))$, where $M(n) \approx 2^{\frac{2^n}{\sqrt{n}}}$, comparing to the dimension of the space of all ordered test sequences of $O(n!)$, where n is the number of tests. Also, our model is not limited to binary trees only; we may choose different numbers of branches (decisions) for different tests. The literature on multi-fold decision trees is very scarce, which makes our model even more valuable, because sometimes it is more natural to divide the results of the tests into more than two categories. However, the main reason to use manifold decision trees is directly related to a difficult problem of choosing "good" thresholds (cut-off points).

The average cost of a strategy depends on the choice of thresholds. For different threshold levels we have different average costs. One method for choosing thresholds can be based on the observation made in [51] that a decision as to whether a container is dangerous does not depend on the sensor reading but on the odds ratio of dangerous to safe containers corresponding to the reading. In general, the larger the number of thresholds (not necessarily optimal ones) the lower the average costs.

We consider two types of objects and two types of decisions, but our model can be generalized for any number of different types of objects and decisions. Decision tree regression is a generalization of a decision tree when a decision is not discrete but continuous (see, e.g., [81]).

The fundamentals of decision analysis were developed in [77, 78, 55, 8]. For the sequential testing a large body of literature exists on the type of problems when the decision function is known (see, e.g., [15, 16, 14, 73, 72, 19, 7, 27, 20, 30, 39, 44, 46, 49, 50, 65, 68]).

An example of a decision tree representation of Boolean function $f(x_1, x_2, x_3) = x_1x_2 \vee x_1x_3$ is given in Figure 1.1. Given the costs of learning the values of the variables x_1, x_2, and x_3, and different criteria for decision trees (e.g., to choose the cheapest one) we may prefer to choose one decision tree over another to learn the value of the function. In general, even if the cost of tests are uniform it is very hard to find the shallowest decision tree representation for a given Boolean function, which is a major problem in circuit complexity.

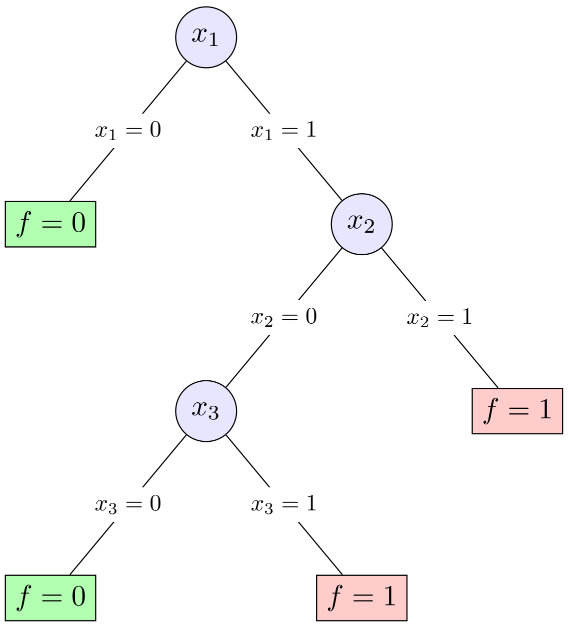

Figure 1.1: A decision tree representation for a Boolean function $f(x_1, x_2, x_3) = x_1x_2 \vee x_1x_3$.

For a simple case of a series Boolean function (i.e., the decision is positive if and only if all the test results are positive), the optimal strategy is to conduct tests in the order of the ratio of cost to probability that the reading will be positive (see, e.g., [61, 19, 64, 1]).

Another simple Boolean function (also called system) when the decision is positive if and only if any of the test results are positive is called parallel. Series-parallel systems are the systems in which components can be grouped hierarchically into series or parallel subsystems. Sequential testing of series-parallel systems when subsystems are tested one by one in the order of their cost/probability ratios was proposed in [44] and was claimed to be optimal for any series-parallel system. It was shown in [7, 64, 16, 73] that it is true only for some special structures of series-parallel systems. In [82] a system of equations for general total cost of inspection is developed for general case of n sensors in series-parallel system. The optimal testing strategy for even more complex Boolean functions was considered in [21].

Decision trees are widely used in data mining and machine learning ([17, 60, 79]). To evaluate the bounds of the sample size required for learning model, a notion of VC-dimension is used (see, e.g., [5, 9, 74]).

Because finding an exact function c from a known class C is usually difficult, we find a function (hypothesis) h from a hypotheses space H that is very close to c. In other words, we minimize the error of h, and if $err(h)$ is small we consider h to be "approximately correct." Because we learn the values of a function from a random sample, it is possible that a bad sample will prevent our ability to find h that is approximately correct. Thus, we allow the learning algorithm to fail with some small probability. In this case we call h "probably approximately correct" or PAC. In other words, h is PAC compliant if

$$Pr[err(h) \leq \epsilon] \geq 1 - \delta.$$

When the set of all hypotheses H is finite, the goal is to find $h \in H$ that is PAC compliant for a sample size of m. The VC-dimension of H, d, provides us with a bound on m to achieve ϵ and δ. Formally, $err(h) \leq \epsilon$ for

$$m = O\left(\frac{d\ ln(1/\epsilon) + ln(1/\delta)}{\epsilon}\right).$$

A point $x = (x_1, ..., x_n) \in \mathbb{B}^n$ is a *true point* of the Boolean function f if $f(x) = 1$, and if $f(x) = 0$ we call x a *false* point, where $\mathbb{B} = \{0, 1\}$. We denote by $T(f)$ the set of true points of Boolean function f, and by $F(f)$ the set of false points of f. Let G be a class of Boolean functions $g : \mathbb{B}^n \to \mathbb{B}$. A set $S \subseteq \mathbb{B}^n$ is said to be *shattered* by G if for any partition of S into two subsets T and F there exists function $g \in G$ for which $T \subseteq T(g)$ and $F \subseteq F(g)$. The *Vapnik-Chervonenkis dimension* of G, denoted by $VCdim(G)$ is defined as the maximum cardinality of a subset $S \subseteq \mathbb{B}^n$ that is shattered by G.

For some classes of Boolean functions VC-dimension is already known, for example, for monomials and threshold functions [4]. In Chapter 10 we derive the VC-dimension of Horn functions, and also the VC-dimension of a more general class of k-quasi-Horn functions. A Boolean function f: $\mathbb{B}^n \to \mathbb{B}$ is *Horn* if it has a DNF representation with at most one negative literal in each term. For any nonnegative integer constant k a Boolean function g: $\mathbb{B}^n \to \mathbb{B}$ is *k-quasi-Horn* if it has a DNF representation with at most k negative literals in each term.

In Chapter 11 we state and prove a criterion for k-quasi-Horn functions that generalizes McKinsey's theorem. Also, necessary and sufficient conditions for function to be bidual k-quasi-Horn are stated and proved. For any Boolean function f its dual f^d is defined as $f^d(x) = \overline{f(\overline{x})}$ for all $x \in \mathbb{B}^n$, where $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$. A function f is *bidual k-quasi-Horn*, if both f and f^d are k-quasi-Horn.

Chapter 2

Main Results

First we formulate a large-scale linear programming model that finds an optimal strategy for a type of problem with unknown diagnosis function. The model finds an optimal strategy in the form of a decision tree that minimizes the expected cost given a detection rate or maximizes the detection rate given the budget. The model is based on a polyhedral description of all decision trees in the space of possible inspection histories (see Chapter 7 for more details). Polyhedral characterization of decision trees applies even if we consider shape restrictions, such as depth of the tree or precedence constraints between tests.

The polyhedral approach is based on the notions of history and subhistory (or path and subpath). For each object and its test results we associate a history, which is the sequence of pairs of test and the range in which the test result belongs, and a final decision about an object. For instance, for three tests with four possible ranges for test results $\pi = ((1,3),(2,4),(3,2),0)$ is an example of a history from a set of all possible histories P, where π describes that the result of the first test falls into the third range; the results on the second test are in the fourth range; at the third test the result is in the second range, and the decision about an object is negative. For every history $\pi \in P$ we say that ν is a subhistory of π and denote $\nu \prec \pi$, if there exists a history $\pi_0 \in P$ such that $(\nu, \pi_0) = \pi$. For the history mentioned above we have four subhistories (including two trivial ones - empty subhistory and history itself):

$$\begin{aligned}
\nu_1 &= (), \\
\nu_2 &= ((1,3)), \\
\nu_3 &= ((1,3),(2,4)), \\
\nu_4 &= ((1,3),(2,4),(3,2)),
\end{aligned}$$

and the following relation holds:

$$() \prec ((1,3)) \prec ((1,3),(2,4)) \prec ((1,3),(2,4),(3,2)) \prec ((1,3),(2,4),(3,2),0).$$

We use a set of equalities to describe a polytope, the vertices of which correspond to decision trees (there is a one-to-one correspondence between vertices and decision trees). In fact, optimal strategy is usually not a single decision tree (pure strategy), but a combination of several decision trees (mixed strategy). In pure strategy, the value of the function is the same for exactly the same values of variables. In mixed strategies we can have different values of the function for the same values of the variables. (See Figures 5.1 and 5.2 in Chapter 5 for examples of mixed strategies.)

Decision trees are widely used in data mining and machine learning ([17, 60, 79]). To evaluate the bounds of the sample size required for learning model, a notion of VC-dimension is used (see, e.g., [5, 9, 74]).

Let us denote by H_k the class of k-quasi-Horn functions $h : \mathbb{B}^n \to \mathbb{B}$, and by P the class of positive Boolean functions. In Chapter 10 we state and prove the VC-dimension of k-quasi-Horn functions.

Theorem 1 *$VCdim(H_k) = \Theta(VCdim(P))$, when $k \ll n$ and $VCdim(P) = \binom{n}{\lfloor \frac{n+1}{2} \rfloor}$.*

In the first part of Chapter 11 we use the notion of k-zeros-conjunction (for more details see Chapter 11) to generalize and prove the McKinsey theorem, which states that a Boolean function is Horn if and only if the set of its false points is closed under conjunction.

Theorem 2 *A Boolean function is k-quasi-Horn if and only if the set of its false points is closed under k-zeros-conjunctions.*

The criteria for bidual Horn functions and DNFs can be found in [31]. We generalize these results and formulate necessary and sufficient conditions for bidual k-quasi-Horn functions and DNFs in the second part of Chapter 11.

We denote by $P(t)$ and $N(t)$ respectively the set of indices of positive and negative literals in the term t, and by $V(t) = P(t) \cup N(t)$ the set of all variable indices in the term t. For any $s \geq 2$ terms $t_{i_1}, \ldots, t_{i_s}$ let us denote by $t^+_{i_1,i_2,\ldots,i_s}$ the positive term such that

$$P(t^+_{i_1,i_2,\ldots,i_s}) = P(t_{i_1}) \cup P(t_{i_2}) \cup \ldots \cup P(t_{i_s}),$$

and by $t^{\pm}_{i_1,i_2,\ldots,i_s}$ the positive term such that

$$P(t^{\pm}_{i_1,i_2,\ldots,i_s}) = V(t_{i_1}) \cup V(t_{i_2}) \cup \ldots \cup V(t_{i_s}).$$

We denote by

$$\tilde{N}_{i_2,i_3,\ldots,i_s}(t_{i_1}) = N(t_{i_1}) \setminus (N(t_{i_2}) \cup N(t_{i_3}) \cup \ldots \cup N(t_{i_s}))$$

the set of indices of the unique (with respect to the terms t_{i_2}, t_{i_3}, $\ldots$, t_{i_s}) negated literals in the term t_{i_1}. To simplify the notations we will use $\tilde{N}(t_{i_1})$ instead of $\tilde{N}_{i_2,i_3,\ldots,i_s}(t_{i_1})$ when all the terms are defined without any ambiguity. We define a positive DNF

$$\hat{t}^{\pm}_{i_1,i_2,\ldots,i_s} = \bigwedge_{q\in\{1,2,\ldots,s\}} \Big(\Big(\bigwedge_{l\in P(t_{i_q})} x_l\Big) \wedge \Big(\bigvee_{l\in \tilde{N}(t_{i_q})} x_l\Big)\Big).$$

In Section 11.4 we formulate and prove necessary and sufficient conditions for bidual k-quasi-Horn functions and DNFs.

Chapter 3

Definitions

Let us assume that we have a set of n Boolean variables $x_i \in \mathbb{B}$, $i \in \mathbb{N}$ and a Boolean function $f(x) = f(x_1, \dots, x_n)$, where $\mathbb{B} = \{0, 1\}$. A Boolean function can be described in different ways. We can represent Boolean function by a formula; for example, $f(x_1, x_2) = x_1 \vee x_2$. Alternatively, we can describe this Boolean function by its truth table (see Table 3.1).

Also, it can be represented by a decision tree (see Figure 3.1). The nodes denoted by circles in Figure 3.1 represent variables. The edges correspond to the values of the variables. For each node and each set of edges that follow that node, the values of variables are assigned to the edges in monotone increasing order; for example, the leftmost edge corresponds to the smallest possible value of the variable, and the rightmost edge corresponds to the largest possible value of the variable. In the case of Boolean variables, the left edge corresponds to the value of zero of a variable, and the right edge corresponds to the value of one. A node that is not followed by edges is called a *leaf*. Leaves store the value of the function and are denoted by squares. Every edge is followed by a leaf or a node that corresponds to a variable. In our example if the value of x_1 is 1 then $f(x_1, x_2) = 1$. Otherwise, if $x_1 = 0$, we must know the value of x_2 to calculate the value of the function. The diagram in Figure 3.1 is an example of a binary decision tree. A *binary decision tree* (BDT) represents a process of the sequential learning of variable values to make a decision about the value of the function, where each node except leaves has exactly two edges. A node that corresponds to the variable that we learn first is called a *root*.

We assume that each variable appears at most once in any path from the root to any leaf of the BDT. It is a reasonable assumption, because we do not need to learn a value of a variable if we have already learned it.

Let us consider the number of BDTs for different number of variables. When we do not

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Table 3.1: Function $f(x_1, x_2) = x_1 \vee x_2$.

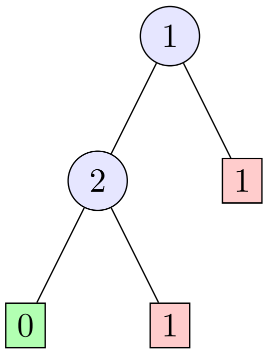

Figure 3.1: Function $f(x_1, x_2) = x_1 \vee x_2$.

have any variables there are two possible BDTs, assigning a value of 0 or 1 to the function. Let us denote by $\mathcal{D}_n$ the set of distinct BDTs that can be formed for a Boolean function of n Boolean variables, and by $N_n = |\mathcal{D}_n|$ the number of such distinct BDTs. We have that $N_0 = 2$ and $N_1 = 6$ (see Figure 3.2). For n Boolean variables the number of BDTs is described by a recursive formula $N_n = 2 + n(N_{n-1})^2$. For two Boolean variables there are 74 possible BDTs; for three Boolean variables we have 16,430 possible BDTs; for four Boolean variables there are more than 1 billion possible BDTs, and for five Boolean variables there are 5x10^{18} distinct BDTs (see [71]).

Let C_i denote the cost to learn a value of the variable x_i, and $p_i = Prob(x_i = 0)$ denote the probability that $x_i = 0$.

Example 1 *The cost of the BDT in Figure 3.1 to learn the value of the function $f(x_1, x_2) = x_1 \vee x_2$ equals $C_1 + p_1 C_2$. The cost of the BDT in Figure 3.3 to learn the value of the same function equals $C_2 + p_2 C_1$. Depending on the values of C_i and p_i we compare trees and choose the least expensive one. For example, given the choice between BDTs in Figure 3.1 and Figure 3.3 for $C_1 = 1$, $C_2 = 2$, and $p_1 = p_2 = .5$, we choose the BDT in Figure 3.1, because it costs $1 + .5(2) = 2$, whereas the cost of BDT in Figure 3.3 is $2 + .5(1) = 2.5$.*

Binary decision trees are also used for classification when the system function is unknown, or may not even exist. In binary classification we call an outcome of 0 *negative*, and outcome of 1 we call *positive*. If based on the values of the variables we predict a positive (negative) outcome, but observe a negative (positive) outcome, we call it *false positive (negative)*. Consider the following example.

Example 2 *Assume that the values of variable x_1 that correspond to negative observations follow standard normal distribution, and the values of the same variable that correspond to positive observations follow normal distribution with mean and variance equal to one. We have many different BDTs that classify our observations; for example, if $x_1 < .5$ declare an observation negative, otherwise positive. In this case 0.5 is a threshold level that is used for classification. Area A+B in Figure 3.4 represents a fraction of correctly classified negative observations, area B - a fraction of false negative observation, C - a fraction of false positive,*

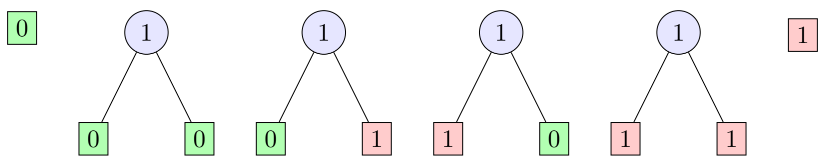

Figure 3.2: All possible distinct BDTs for a Boolean function with one variable.

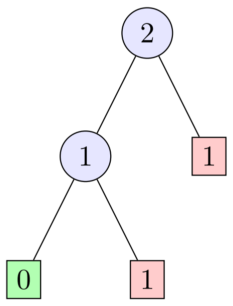

Figure 3.3: Another BDT to learn the value of the function $f(x_1, x_2) = x_1 \vee x_2$.

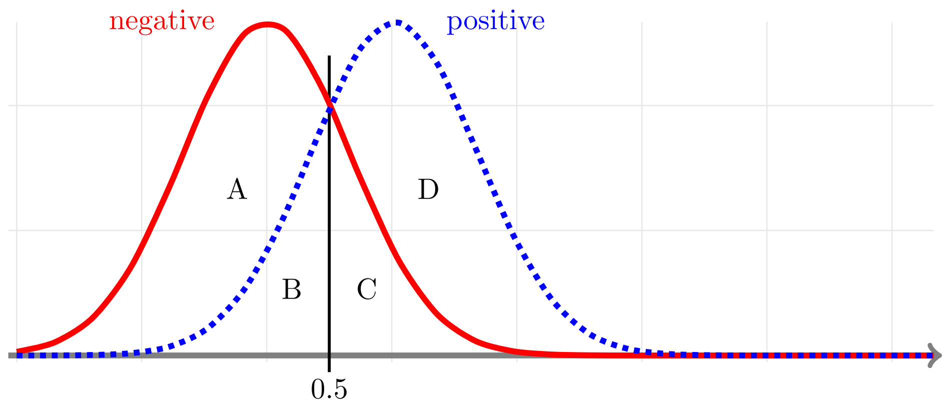

Figure 3.4: Distributions of the values of the variable x_1 for negative and positive observations, indicated respectively by solid and dotted curves.

and C+D - a fraction of correctly classified positive observations. If costs of misclassification are equal; that is, the cost of false positive and false negative are equal, we minimize the fraction of misclassified outcomes by minimizing the sum of the areas that represent the fraction of misclassified observations (B+C). In case of different misclassification cost we impose weights on the areas B and C.

In general, not only learning the variables values has costs, but also the function value itself, or the fact that observation is misclassified. A patient with positive test results undergoes an expensive procedure, and for a patient with false negative results the cost of a procedure is even more expensive because of lost time. Let us denote by C_P and C_N the cost of positive and negative classification, and by C_{FP} and C_{FN} the cost of false positive/negative classification respectively. We denote by $U(D,i)$ the *utilization* of the set of the nodes that correspond to the variable x_i and decision tree D; that is, if we repeatedly learn the value of the function for different values of the variables using decision tree D, $U(D,i)$ corresponds to the frequency of learning the value of variable x_i. For example, for binary decision tree D in Figure 3.1 we have $U(D,1) = 1$ and $U(D,2) = p_1$, because we must always learn the value of variable x_1, and the value of the variable x_2 we learn only when the value of the variable x_1 equals zero. The cost $C(D)$ of a binary decision tree D is

$$C(D) = \sum_{i=1}^{n} U(D,i)C_i + U(D,P)(C_P + Q_{FP}C_{FP}) + U(D,N)(C_N + Q_{FN}C_{FN}),$$

where $U(D,P)$ and $U(D,N)$ denote the frequency of positive and negative classification respectively. Q_{FP} and Q_{FN} denote the fraction of false positive and false negative classification respectively. We denote by $\Delta(D)$ the *detection rate*; that is, the fraction of correctly classified positive observations. Obviously,

$$\Delta(D) = 1 - Q_{FP}.$$

Chapter 4

A Model from the Literature

In this section we describe an application of BDT to approach a classification problem considered in [71]. Containers that arrive at U.S. ports may carry illegal nuclear materials, and must be classified as either dangerous (positive) or not dangerous (negative). A decision as to whether a container is dangerous is based on four different tests with binary outcomes. The testing procedure follows BDT structure; that is, tests are performed sequentially, and after a test a choice must be made: to perform another test or to decide whether a container is dangerous. Given the costs of tests, classifications, and misclassifications, the problem is to find the least expensive testing strategy for a given detection rate.

The tests with binary outcomes correspond to Boolean variables x_i, $i = 1, \ldots, n$. To find the least expensive BDT only BDTs that correspond to complete and increasing (monotone) Boolean functions are considered (called feasible BDTs), which significantly decreases the number of BDTs under consideration (see Table 4.1). Completeness means that we consider only testing strategies that utilize all four tests. Monotonicity supports the intuitive fact that switching the results of one of the tests from negative to positive increases the likelihood of positive outcome.

Depending on whether the observations are positive or negative, the test results y_i considered in [71] follow a Normal distribution $N(\mu(P,i), \sigma^2(P,i))$ or $N(\mu(N,i), \sigma^2(N,i))$ with the corresponding probability density functions $\phi(P,i,u)$ or $\phi(N,i,u)$, respectively. To transform y_i into a Boolean variable x_i we choose a threshold t_i and say that $x_i = 0$ if $y_i < t_i$, and $x_i = 1$ otherwise. For each feasible BDT a two level suboptimization is applied in the evaluation of each BDT to obtain the set of threshold values that minimize the cost function given a detection rate:

Number of variables	Number of all BDTs	Number of feasible BDTs
2	74	4
3	$16,430$	60
4	$1,079,779,602$	$11,808$
5	5x10^{18}	$263,515,920$

Table 4.1: Number of all and feasible BDTs depending on the number of variables.

$$\min_{t\in\mathcal{R}^n,\ D\in\mathcal{D}_n} C(D,t) = \sum_{i=1}^{n} U(D,i,t)C_i + U(D,P,t)(C_P + Q_{FP}C_{FP}) + U(D,N,t)(C_N + Q_{FN}C_{FN})$$

subject to

$$\Delta(D) \geq \Delta_0,\ D \in \mathcal{D}_n.$$

Then, given all feasible BDTs and corresponding cost, a BDT with the smallest cost is chosen

$$\min_{D\in\mathcal{D}_n} C(D).$$

The model was tested for four tests and the data in Table 4.2. We also have that $C_P = 600$. For $\Delta_0 = .815$ the optimal BDT with $t = (2.6, 1.07, 1.16, 2.2)$ is shown on Figure 4.1. We use "CHK" for positive observations, because after the tests we check containers for dangerous content, and "OK" is used for negative observations, because we declare containers safe and let them go.

test i	1	2	3	4
C_i	.32	.92	57	176
$\mu(N, i)$	0	0	0	0
$\mu(P, i)$	4.37	1.53	2.9	4.6
$\sigma^2(N, i)$	1	1	1	1
$\sigma^2(P, i)$	1	1	1	1

Table 4.2: Data for four tests.

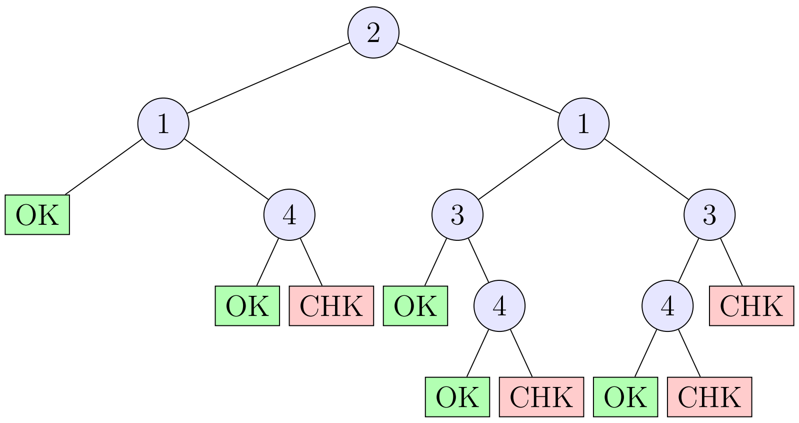

Figure 4.1: Optimal testing strategy.

Chapter 5
More Definitions

We will use the terms BDT and decision strategy, or strategy, interchangeably hereafter because BDT is indeed a strategic plan to learn the values of the variables in a particular order to guess about the value of the hidden function.

5.1 Pure and Mixed Decision Trees

Assume that $\mathcal{T}$ is a set of object types and $\mathcal{T} = \{G, B\}$; in other words, each object is either of "good" (type G) or "bad" (type B) quality. We assume that we rarely observe the objects of bad quality, where "rarely" implies a subjective judgment and depends on the problem.

Strategies that we considered above are deterministic, in the sense that for exactly the same values of variables the value of the function is the same. We call such strategy *pure*. Let us consider another type of strategy, *mixed*, when for the same values of the variables we can have different values of the function. Figure 5.1 represents a general mixed strategy. Root node denoted by M means that the strategy is mixed. Nodes D_1 and D_2 represent pure strategies. The probability that we use the strategy D_1 to make a decision about the value of the function is denoted by α.

Consider a mixed decision tree on Figure 5.2. The value of the function is 40% of the time the same as the value of the variable x_1, and other 60% of the time the value of the function equals one independent of the value of the variable.

Pure inspection strategies do not necessarily provide the best possible solutions. We illustrate this statement by Figure 5.3, assuming that the decision function depends on a single variable.

We choose density functions $\phi(N, x) = 10e^{-10x}$ and $\phi(P, x) = 3e^{-3x}$. We assume that

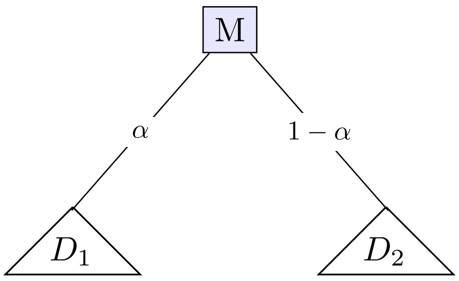

Figure 5.1: Mixed decision tree.

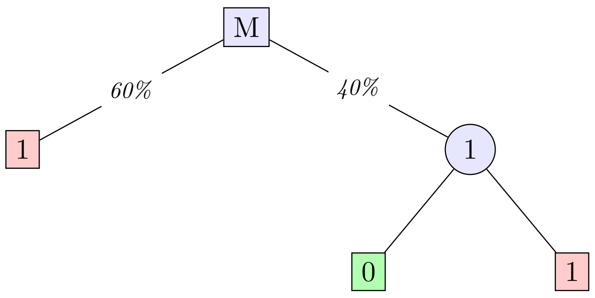

Figure 5.2: An example of a mixed decision tree.

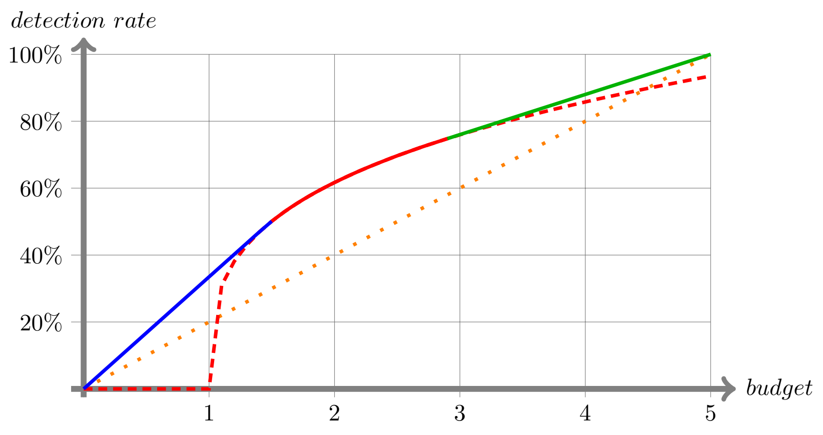

Figure 5.3: Pure strategies (dotted and dashed lines) and mixed strategies (solid line).

$C_1 = 1$ and $C_P = 5$.

The three pure strategies are:

(1) declare every observation positive until the budget is exhausted (denoted by dotted line on Figure 5.3);

(2) learn the value of the variable for all observations, and use the remaining budget to declare the observations positive in decreasing order of the odds that they are positive; that is, in decreasing order of the value of the variable x_1 (dashed line);

(3) declare all the observations negative.

The performance of the first two pure strategies is shown in Figure 5.3, whereas the third strategy clearly has a detection rate equal to zero. The best mixed strategy is also shown in the figure (solid line).

At very low budgets, the best strategy is a mixed use of the second and the third strategies (blue solid line). This use can be represented by a diagram as the first decision tree in Figure 5.4, where the small square represents the (random) mixing. As the budget increases we encounter a range of budgets where the optimal solution is the compound strategy (red solid line) represented by the second decision tree in Figure 5.4. Finally, the mixed strategy becomes a mixture of the first and the second strategies (green solid line), represented by the third decision tree in Figure 5.4.

Figure 5.5 illustrates another example showing when mixed testing strategies allow using a budget the most efficiently. Assume we have four pure decision trees with different costs and detection rates. Given the budget of \$10, the best pure strategy is strategy D_2, providing 40% detection rate. At the same time, a 50-50 mix of strategies, D_2 and D_3, provides a detection rate of 60%.

A notion of mixed strategy is extensively used in game theory (see, e.g., [62, 63, 76]). Usually, a player would use a mixed strategy when the opponent could benefit from knowing the next move. This property has at least two benefits in cases of container inspections. First, the strategy allows inspectors to achieve higher detection rates, and second, unlike pure strategy, advisories cannot adopted it.

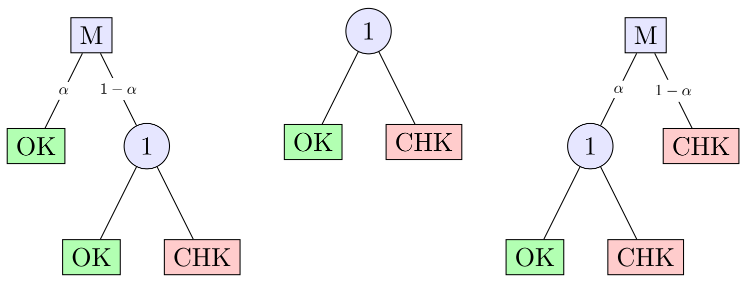

Figure 5.4: Decision trees that correspond to mixed strategies are denoted on the Figure 5.3 by blue, red, and green solid lines respectively.

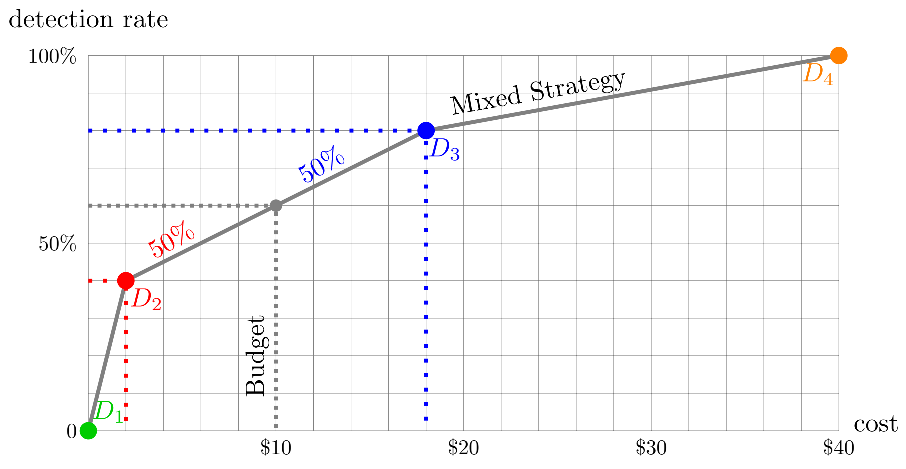

Figure 5.5: Best (mixed) strategy for a given budget.

5.2 Manifold Decision Trees and Choice of Thresholds

In a BDT every node, except a leaf, is followed by two branches. A more general case is a *manifold decision tree*, in which every non-leaf node is followed by several branches. An example of a manifold decision tree is shown in Figure 5.6, where node 2 has 3 branches.

For every node $i \in \mathcal{N}$, where $\mathcal{N} = \{1, \dots, n\}$ we fix k_i thresholds $t^i = (t^i_1, t^i_2, \dots, t^i_{k_i})$, that is, we have $k_i + 1$ different ranges for the values of the variable x_i. We denote by $\mathcal{R}_i = \{1, \dots, k_i + 1\}$ the set of the ranges for the node $i \in \mathcal{N}$.

Different methods are available to associate ranges with branches. One way is to do it monotonically; that is, for each variable range j corresponds to the j-th branch from the left. Another way is to associate a group of ranges with a branch based on some criteria. One criteria is odds ratio. For each range we calculate the ratio of the probability of a negative observation to the probability of a positive observation. Then we assign the ranges to the branches in increasing order of the odds ratios, where more than one range can be assigned to the same branch.

5.3 Decision Trees of Limited Depth

The *depth* of the decision tree is the length of the longest path in a tree, or, in other words, the maximum number of tests that we run for any object. For example, the decision tree on Figure 5.6 is of depth 2, because although for some objects it is enough to make a positive decision after the results of test 1, some of the objects require the results of both tests, 1 and 2, to make a decision. The restriction on the depth of the tree becomes useful when we have additional constraints; for example, constraints on time we can spend testing the objects. If any test requires an hour to obtain the results, and we cannot spend more than 3 hours testing an object, then we should restrict our analysis to the trees of depth at most 3, even if there are more than 3 available tests. We derive theoretical results without restricting the depth of the trees, but analogous analysis can be applied to limited depth trees. We provide solution of our model for limited depth trees in Chapter 8.

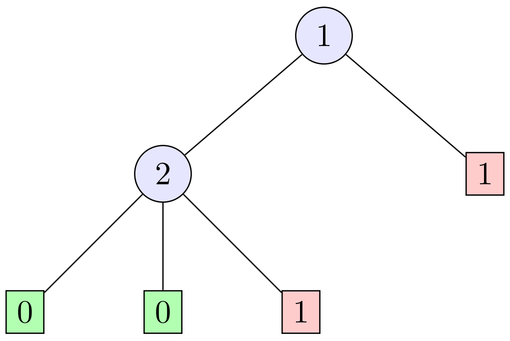

Figure 5.6: An example of a manifold decision tree.

Chapter 6
Macro Approach

6.1 Macro Model

Using macro approach we define a mixed decision tree with a variable

$$x = (x(D_1), \dots, x(D_{|\mathcal{D}|})),$$

where $x(D_i)$ is a proportion of objects that are tested using a decision tree D_i, for $i = 1, \dots, |\mathcal{D}|$, where $\mathcal{D}$ is a family of all feasible decision trees. Clearly we have that $\sum\limits_{D \in \mathcal{D}} x(D) = 1$ and $x(D) \geq 0$. Because all strategy costs, detection rate, and utilization of tests are linear with respect to mixing, we can define them as linear functions; that is, cost of a testing strategy x is defined as

$$C(x) = \sum_{D \in \mathcal{D}} C(D)x(D),$$

the detection rate of a testing strategy x as

$$\Delta(x) = \sum_{D \in \mathcal{D}} \Delta(D)x(D),$$

and the utilization of test $i \in \mathcal{N}$ in a testing strategy x as

$$U(x, i) = \sum_{D \in \mathcal{D}} U(D, i)x(D).$$

As an example, let us consider decision tree D_0 given in Figure 4.1. If D_0 is optimal, then we have $x(D_0) = 1$ and $x(D) = 0$ for all $D \in \mathcal{D}$ such that $D \neq D_0$. Also,

$$\begin{aligned} C(x) &= C(D_0), \\ \Delta(x) &= \Delta(D_0), \\ U(x,i) &= U(D_0,i) \text{ for } i \in \mathcal{N}. \end{aligned}$$

As another example, consider a decision tree in Figure 5.1. If the tree is optimal and $\alpha = 0.1$, then $x(D_1) = .1$, $x(D_2) = .9$, and $x(D) = 0$ for all $D \in \mathcal{D}$ such that $D \neq D_1$ and $D \neq D_2$. For this decision strategy

$$\begin{aligned} C(x) &= .1C(D_1) + .9C(D_2), \\ \Delta(x) &= .1\Delta(D_1) + .9\Delta(D_2), \\ U(x,i) &= .1U(D_1,i) + .9U(D_2,i) \text{ for } i \in \mathcal{N}. \end{aligned}$$

In general, we consider the problem of finding an optimal testing strategy that minimizes average costs of testing the objects, detects a certain prescribed fraction Δ_0 of bad objects, and limits the utilization of tests by given capacities $K(i)$. We can formulate that problem as the following linear programming problem:

$$\begin{array}{lll} \min\limits_{x} \sum\limits_{D \in \mathcal{D}} C(D)x(D) & & \\ \textit{subject to} & & \\ \qquad \sum\limits_{D \in \mathcal{D}} \Delta(D)x(D) & \geq & \Delta_0 \\ \qquad \sum\limits_{D \in \mathcal{D}} U(D,i)x(D) & \leq & K(i) \quad \text{for all } i \in \mathcal{N} \\ \qquad \sum\limits_{D \in \mathcal{D}} x(D) & = & 1 \\ \qquad x(D) & \geq & 0 \qquad \text{for all } D \in \mathcal{D}. \end{array} \tag{6.1}$$

Similarly, the problem of finding the testing strategy that maximizes the detection rate for a given budget C_0 and limited capacities can also be written as a linear programming problem:

$$\begin{aligned}
&\max_x \sum_{D\in\mathcal{D}} \Delta(D)x(D) \\
&\textit{subject to} \\
&\qquad \sum_{D\in\mathcal{D}} C(D)x(D) \le C_0 \\
&\qquad \sum_{D\in\mathcal{D}} U(D,i)x(D) \le K(i) \quad \text{for all } i\in\mathcal{N} \\
&\qquad \sum_{D\in\mathcal{D}} x(D) = 1 \\
&\qquad x(D) \ge 0 \quad \text{for all } D\in\mathcal{D}.
\end{aligned} \tag{6.2}$$

Many other variants of these problems, involving limits on rates of false positives, inspection times, etc., can be formulated analogously as linear programming problems.

The main practical difficulty in using the above models is their size. The number of variables in these formulations is double exponential in the number of tests. For example, in [71] more than 11000 binary decision trees are enumerated for 4 tests. With binary partitions of the sensor readings, however, we cannot achieve the best performance. To utilize the power of the above models, we want a much finer partition of tests results, so that the optimization will be able to select the best decision sequence for the decision trees appearing in the optimal mixed strategy. If we allow, for example, 10 thresholds for all tests, the number of variables in (6.1) (or (6.2)) will jump to well above 100 million, making it almost impossible to generate them all and to write in full detail the linear programming problems (6.1) and (6.2).

On the positive side, there are only $2 + |\mathcal{N}|$ constraints in the above formulations, and thus the optimal basic solution x^* involves at most $2 + |\mathcal{N}|$ decision trees with nonzero x^*_D values. This suggests that such a mixed strategy is far from being impractical, because it is not a complicated matter to switch between a small number of decision trees.

The small size of a basic solution also implies that *column generation* might be a good way of solving these very large linear programming problems. This technique was introduced first in [37, 38] for the solution of cutting stock problems, but since then column generation has become a standard approach, used widely and efficiently in numerous applications. For the sake of completeness we describe briefly the main ideas.

Let us consider, for example, problem (6.2) and its linear programming dual, which has only $2 + |\mathcal{N}|$ variables, α corresponding to the budget constraint, β_i, $i \in \mathcal{N}$ corresponding to the capacity constraints, and γ corresponding to the last balance equality. Using these we can write the dual of (6.2) as follows:

$$
\begin{aligned}
&\min\ C_0\,\alpha \ +\ \sum_{i\in\mathcal{N}} K(i)\,\beta_i \ +\ \gamma \\
&\textit{subject to} \\
&C(D)\,\alpha \ +\ \sum_{i\in\mathcal{N}} U(D,i)\,\beta_i \ +\ \gamma \ \geq\ \Delta(D) \qquad \text{for all } D \in \mathcal{D} \\
&\qquad\qquad\qquad \alpha,\gamma,\beta_i \ \geq\ 0 \qquad \text{for all } i \in \mathcal{N}.
\end{aligned}
\tag{6.3}
$$

Let us now consider a small subset $\widehat{\mathcal{D}} \subseteq \mathcal{D}$ of all decision trees, and suppose that we have solved (6.2) restricted to this set (i.e., we replace $\mathcal{D}$ everywhere in (6.2) by $\widehat{\mathcal{D}}$). Let us denote by x^* the optimal solution to this restricted variant of (6.2), and let α^*, β_i^*, $i \in \mathcal{N}$, and γ^* denote the corresponding optimal solution of the dual of the restricted problem (i.e., where we keep constraints in (6.3) only for $D \in \widehat{\mathcal{D}}$).

The theory of linear programming then implies (see e.g., [24]) that x^* is an optimal solution for the unrestricted problem (6.2) if and only if

$$
\min_{D\in\mathcal{D}} \left[C(D)\,\alpha^* \ +\ \sum_{i\in\mathcal{N}} U(D,i)\,\beta_i^* \ +\ \gamma^* \ -\ \Delta(D) \right] \ \geq\ 0. \tag{6.4}
$$

If this is the case, we can stop, and x^* is our optimal solution. Otherwise, we choose a decision tree D^* for which the minimum in (6.4) is attained, increment $\widehat{\mathcal{D}} = \widehat{\mathcal{D}} \cup \{D^*\}$, and return to the solution of the restricted (6.1) problem. Experience shows that, on average, this approach generates only a small number of the variables in (6.2). In fact, as a consequence of the results in [47], the number of times we need to solve (6.4) is limited by a polynomial of the rank of (6.2) and the largest of the determinants of (6.2).

However, for this to work, one has to be able to solve subproblem (6.4) efficiently. In its current form, problem (6.4) still seems to require the complete enumeration of all decision trees, which is an approach that does not scale well, as we noted before. In the following sections we show that problem (6.4) can in fact be reformulated such that it can be solved efficiently.

Chapter 7

Micro Approach

7.1 Histories and Decision Trees

We denote by $\mathcal{P}_d(k_1+1,\dots,k_n+1)$ the set of all possible histories of depth at most d; that is, paths from root to leaves of depth at most d for n tests, where k_i denotes the number of the thresholds applied at test i for all $i=1,\dots,n$. We say that $\pi \in \mathcal{P}_d(k_1+1,\dots,k_n+1)$ if either $\pi=(OK)$, or $\pi=(CHK)$, or $\pi=((i_1,r_{i_1}),\dots,(i_s,r_{i_s}),b)$, where $i_j \in \mathcal{N}$, $s \le d$, $r_{i_j} \in \mathcal{R}_{i_j}$ for all $j=1,\dots,s$, and $b \in \{OK, CHK\}$. For example, $\pi=(CHK)$ denotes a path when we check the objects without prior testing, and $\pi=((2,1),(1,3),OK)$ is a path when we send an object to test 2; if the results are in the first range, we send an object to test 1, and if the results are in the third range we do not check an object. We consider paths and decision trees without restriction on depth, but similar analysis can be provided for limited depth trees. To simplify notations we will use $\mathcal{P}$ instead of $\mathcal{P}_n(k_1+1,\dots,k_n+1)$ when it creates no ambiguity.

We denote by $\mathcal{P}(D)$ the set of histories that correspond to the testing strategy D. Consider a testing strategy D' represented by Figure 7.1. It consists of four histories:

$$\begin{aligned}
\pi_1 &= ((1,1),(2,1),OK)\\
\pi_2 &= ((1,1),(2,2),OK)\\
\pi_3 &= ((1,1),(2,3),CHK)\\
\pi_4 &= ((1,2),CHK).
\end{aligned}$$

Hence $\mathcal{P}(D')=\{\pi_1,\pi_2,\pi_3,\pi_4\}$, while $\mathcal{P}(2,3) \supseteq \mathcal{P}(D')$ is larger, including, for example, $\pi_5=((1,1),OK)$.

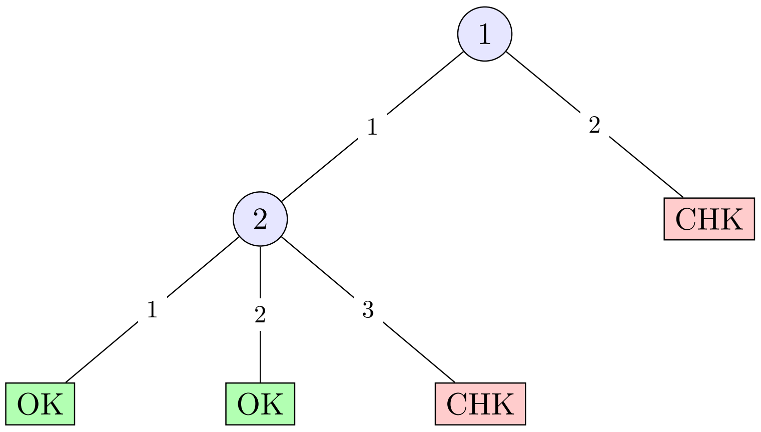

Figure 7.1: Testing strategy D'.

Let us recall from Chapter 2 that for every path $\pi \in \mathcal{P}$ we say that ν is a subpath of π and denote $\nu \prec \pi$, if there exists a path $\pi_0 \in \mathcal{P}$ such that $(\nu, \pi_0) = \pi$. We denote by $\widetilde{\mathcal{P}}$ the set of all subpaths that correspond to $\mathcal{P}$. We denote by $f(\tau, i, j)$ the fraction of the objects of type $\tau \in \mathcal{T} = \{good,\ bad\}$ to have test results on test i in the range j for all $i \in \mathcal{N}$ and $j \in \mathcal{R}_i$, or in other words

$$f(\tau, i, j) = \int\limits_{t_{j-1}}^{t_j} \phi(\tau, i, x) dx.$$

For all $\pi \in \mathcal{P}$ we denote by $g(\pi)$ and $b(\pi)$ respectively the proportion of good and bad objects that follow path π (we assume that the test results are stochastically independent):

$$\begin{aligned} g(\pi) &= \prod_{(i,j)\in\pi} f(G, i, j) \text{ for all } \pi \in \mathcal{P}, \\ b(\pi) &= \prod_{(i,j)\in\pi} f(B, i, j) \text{ for all } \pi \in \mathcal{P}. \end{aligned} \tag{7.1}$$

Because every container of either type must have some specific path, these parameters satisfy the equalities

$$\sum_{\pi\in\mathcal{P}} g(\pi) = \sum_{\pi\in\mathcal{P}} b(\pi) = 1. \tag{7.2}$$

7.2 The Polytope of Decision Trees

Let us consider a given set $\mathcal{N}$ of tests, a fixed set of thresholds for test results, and recall that we denote by $\mathcal{P}$ the set of all possible object histories, while $\mathcal{P}(D)$ denotes the set of histories of the objects inspected by strategy D. In other words, we have

$$\mathcal{P} \;=\; \bigcup_{D\in\mathcal{D}} \mathcal{P}(D).$$

Furthermore, any decision tree D can, equivalently, be represented by the vectors $(g(\pi) \mid \pi \in \mathcal{P}(D))$ and $(b(\pi) \mid \pi \in \mathcal{P}(D))$. Let us extend these vectors with zeros, and define in this way $g(D)$, $b(D) \in \mathbb{R}^{\mathcal{P}}$, where

$$g_\pi(D) \;=\; \begin{cases} g(\pi) & \text{if } \pi \in \mathcal{P}(D) \\ 0 & \text{if } \pi \in \mathcal{P} \setminus \mathcal{P}(D) \end{cases} \qquad \text{and} \quad b_\pi(D) \;=\; \begin{cases} b(\pi) & \text{if } \pi \in \mathcal{P}(D) \\ 0 & \text{if } \pi \in \mathcal{P} \setminus \mathcal{P}(D) \end{cases}$$

In fact a mixed strategy $x = (x(D) \mid D \in \mathcal{D})$ can also be represented naturally by the fractions of good and bad objects having a particular history $\pi \in \mathcal{P}$. The corresponding vectors $g(x)$, $b(x) \in \mathbb{R}^{\mathcal{P}}$ are convex combinations:

$$g(x) \;=\; \sum_{D \in \mathcal{D}} x(D) g(D) \quad \text{and} \quad b(x) \;=\; \sum_{D \in \mathcal{D}} x(D) b(D).$$

Thus, we can associate naturally two polytopes, P_g and P_b, to the set of tests and the fixed thresholds of their test results, corresponding to the two object types we considered. These polytopes are defined as

$$P_g = ConvHull\,\langle g(D) \mid D \in \mathcal{D} \rangle \;=\; \left\{ g(x) \;\middle|\; \begin{array}{c} \sum_{D \in \mathcal{D}} x(D) \;=\; 1 \\ x(D) \geq 0 \text{ for all } D \in \mathcal{D} \end{array} \right\}$$

and

$$P_b = ConvHull\,\langle b(D) \mid D \in \mathcal{D} \rangle \;=\; \left\{ b(x) \;\middle|\; \begin{array}{c} \sum_{D \in \mathcal{D}} x(D) \;=\; 1 \\ x(D) \geq 0 \text{ for all } D \in \mathcal{D} \end{array} \right\}.$$

The following statement provides a linear characterization of the polytopes P_g and P_b. This result turns out to be very useful for solving problem (6.2) (and other variants). We state the characterization only for polytope P_g, because the perfectly analogous claim can easily be shown in the same way for P_b, as well. To simplify notation, we assume that the tests results are stochastically independent, and parameters $g(\pi)$ and $b(\pi)$ are computed by (7.1) for all $\pi \in \mathcal{P}$. Let us add, however, that this is just a technical detail, and analogous linear description for the polytopes P_g and P_b can be derived for stochastically dependent tests, as well.

Theorem 3 *Given $y \in \mathbb{R}^{\mathcal{P}}$, we have $y \in P_g$ if and only if*

$$\sum_{\pi \in \mathcal{P}} y(\pi) \;=\; 1 \tag{7.3}$$

and

$$\sum_{\substack{\pi \in \mathcal{P} \\ (\nu,(i,r)) \prec \pi}} y(\pi) = f(G,i,r) \sum_{j=1}^{k_i+1} \sum_{\substack{\pi \in \mathcal{P} \\ (\nu,(i,j)) \prec \pi}} y(\pi) \tag{7.4}$$

for all $i \in \mathcal{N}$, $r \in \mathcal{R}_i$, $\nu \in \widetilde{\mathcal{P}}$.

Proof. We prove the statement by "peeling off" vectors $g(D)$ from y, until we arrive to $y = 0$, each time decreasing the number of histories $\pi \in \mathcal{P}$ for which $y(\pi) > 0$.

To be able to do this "peeling off" process, we set a real parameter $\delta = 1$, and claim that if y satisfies equalities (7.3) and (7.4), then there exists a decision tree $D \in \mathcal{D}$ such that $y(\pi) > 0$ for all $\pi \in \mathcal{P}(D)$. For such a decision tree D, we set

$$\lambda = \min_{\pi \in \mathcal{P}(D)} \frac{y(\pi)}{g(\pi)}.$$

Because of (7.2) and (7.3), we must have $0 \leq \lambda \leq 1$. Let us set $x(D) = \delta\lambda$. If $\lambda = 1$, then we must have $y = g(D)$, and our proof is complete. Since both y and $g(D)$ satisfies equalities (7.3) and (7.4), by setting

$$y' = \frac{y - \lambda g(D)}{1 - \lambda} \qquad \text{and} \qquad \delta' = \delta(1 - \lambda).$$

We obtain a vector y' that also satisfies (7.3) and (7.4), and which has at least one less history with a positive $y'(\pi)$ value than vector y had in the previous step. Thus, replacing y by y', δ by δ', and repeating the above steps, after a finite number of iterations, we can arrive at a convex combination of the vectors $g(D)$, $D \in \mathcal{D}$ which is equal to y, completing the proof of our statement.

What is left is to show our claim, namely that if y satisfies equalities (7.3) and (7.4), then there exists a decision tree $D \in \mathcal{D}$ such that $y(\pi) > 0$ whenever $g(\pi) > 0$ (recall that $g(D) = (g(\pi) \mid \pi \in \mathcal{P}(D))$.

We prove this claim by induction on the number of tests. Clearly, if we have only one test, then the claim is almost trivial, because we only have to try all possible final decisions at the leafs of the only possible trivial decision tree (consisting of only one node).

Let us now assume that we proved this claim for up to m tests, and consider the case when $|\mathcal{N}| = m + 1$. Consider an arbitrary history $\pi \in \mathcal{P}$ for which $y(\pi) > 0$, and let $i \in \mathcal{N}$ be the first test appearing in this history (i.e., for which $(i, j) \prec \pi$ for some $j \in \mathcal{R}_i$). Since $f(G, i, j) > 0$ for all $j \in \mathcal{R}_i$, we must also have histories $\pi' \in \mathcal{P}$ with $y(\pi') > 0$ and with $(i, j) \prec \pi'$ for all $j \in \mathcal{R}_i$. Let us denote by $\widehat{\pi}$ the history obtained from π by deleting the first sensor from π, and consider the sets

$$\mathcal{P}_j \;=\; \{\widehat{\pi} \mid \pi \in \mathcal{P},\ (i, j) \prec \pi\}$$

for $j \in \mathcal{R}_i$. Each set $\mathcal{P}_j$ is the set of all possible histories for the smaller set $\mathcal{N} \setminus \{i\}$ of tests. Furthermore, for an index $j \in \mathcal{R}_i$ let us define

$$\rho_j = \sum_{\pi \in \mathcal{P}, (i,j) \prec \pi} y(\pi)$$

and a new vector $y^j \in \mathbb{R}^{\mathcal{P}_i}$ defined by

$$y^j_{\widehat{\pi}} = \frac{1}{\rho_j} y(\pi) \quad \text{for all} \quad \widehat{\pi} \in \mathcal{P}_j.$$

Then, y^i is a vector satisfying equations (7.3) and (7.4), with respect to tests $\mathcal{N} \setminus \{i\}$. Thus, we must have a decision tree D_j such that $y^j_{\widehat{\pi}} > 0$ for all $\widehat{\pi} \in \mathcal{P}(D_j)$ by our inductive hypothesis. Because this holds for all $j \in \mathcal{R}_i$, by gluing these decision trees D_j, $j \in \mathcal{R}_i$ to test i as a root note, we obtain a decision tree D rooted at test i, satisfying our claim. $\square$

7.3 Micro Model

To provide an efficient linear programming based scheme, we note that by Theorem 3 every decision tree can be viewed as a particular (vertex) $y \in P_g$, and conversely, every vector $y \in P_g$ can be viewed as a mixed strategy. Thus, to solve (6.4) all we need is to represent $D \in \mathcal{D}$ by the corresponding y vector, and perform the minimization over $y \in P_g$ instead of $D \in \mathcal{D}$. If we can do this, such that we obtain an expression which is linear in y, then we can solve (6.4) as a linear programming problem.

To this end, let us associate cost, detection rate, load parameters, and so forth, to every history $\pi \in \mathcal{P}$. Let us denote by $\mathcal{N}(\pi)$ the set of tests appearing in π, and define

$$C(\pi) = \begin{cases} \displaystyle\sum_{i \in \mathcal{N}(\pi)} C_i & \text{if } OK \in \pi, \\ C_{CHK} + \displaystyle\sum_{i \in \mathcal{N}(\pi)} C_s & \text{if } CHK \in \pi, \end{cases}$$

$$\Delta(\pi) = \begin{cases} 0 & \text{if } OK \in \pi, \\ b(\pi) & \text{if } CHK \in \pi, \end{cases}$$

where $b(\pi)$ is defined in (7.1), and

$$U_i(\pi) = \begin{cases} 1 & \text{if } i \in \mathcal{N}(\pi), \\ 0 & \text{if } i \notin \mathcal{N}(\pi), \end{cases}$$

for $i \in \mathcal{N}$. Furthermore, to a vector $y \in P_g$ we associate

$$\begin{aligned} C(y) &= \sum_{\pi \in \mathcal{P}} C(\pi) y_\pi, \\ \Delta(y) &= \sum_{\pi \in \mathcal{P}} \Delta(\pi) \frac{y_\pi}{g_\pi}, \text{ and} \\ U_i(y) &= \sum_{\pi \in \mathcal{P}} U_i(\pi) y_\pi \quad \text{for } i \in \mathcal{N}. \end{aligned}$$

In particular, if $y = g(D)$ for a decision tree $D \in \mathcal{D}$, then we have $C(y) = C(D)$, $\Delta(y) = \Delta(D)$ and $U_i(y) = U_i(D)$ for $i \in \mathcal{N}$.

Thus, we can rewrite the optimization problem in (6.4) as the following linear programming problem:

$$\min_{\mathbf{y} \in P_g} \left[C(y)\, \alpha^* \;+\; \sum_{i \in \mathcal{N}} U_i(y)\, \beta_i^* \;+\; \gamma^* \;-\; \Delta(y) \right], \tag{7.5}$$

where we use formulae (7.3) and (7.4) from Theorem 3 to minimize over $\mathbf{y} \in P_g$.

Note that in the subsequent iterations only the objective function in (7.5) changes, and hence we can keep the previously optimal basis, and just re-optimize it with the new objective, which increases efficiency and, perhaps, will save time.

Chapter 8

Applications

8.1 Inspection of Containers at U.S. Ports

We applied our model to the container inspection problem considered in Chapter 4 using the same data, described in Table 4.2 with $C_{CHK} = 600$ and $\Delta_0 = .815$. For one threshold, for each of the four tests we obtained a decision tree depicted in Figure 8.1. It is a mixed strategy, when 7.9% of the containers are inspected differently than the other 92.1%. Restricting the model to the binary decision trees of depth 2 provides a solution shown in Figure 8.2. Figure 8.3 shows the optimal cost for unlimited depth and different number of thresholds (blue line), and we list the sizes of the problems and corresponding computing times in Table 8.1. While the average costs of decision trees in Figure 8.1 and 8.2 are higher if compared with pure strategy from the previous study (red dot), for four thresholds and more, we see a cost reduction, with 10% cost reduction for seven thresholds. In general, with multiple thresholds and depth 4 we obtain better results and improve computational time if compared with the classical approach.

This problem can be considered to be a learning problem, where we can learn from past test results from past occurrences. To compare our model with a J48 classifier in Weka ([80]), we generated 10,000 observations (9,980 negative and 20 positive) according to given normal distributions in Table 4.2. Using 10-folding cross-validation for J48, the average detection rate is $\Delta_0 = .85$, where for all the containers are first sent to sensor 4. For the same detection rate we solved our model for decision trees of depth two and five ranges for every test. The solution is a mixed decision tree, where 24% of the time we use the decision tree described in Figure 8.4, and the other 76% of the time the one in Figure 8.5. Despite carefully chosen thresholds in Weka and almost randomly chosen thresholds in our model, our solution provides a less costly decision tree: because all the containers in Weka are sent

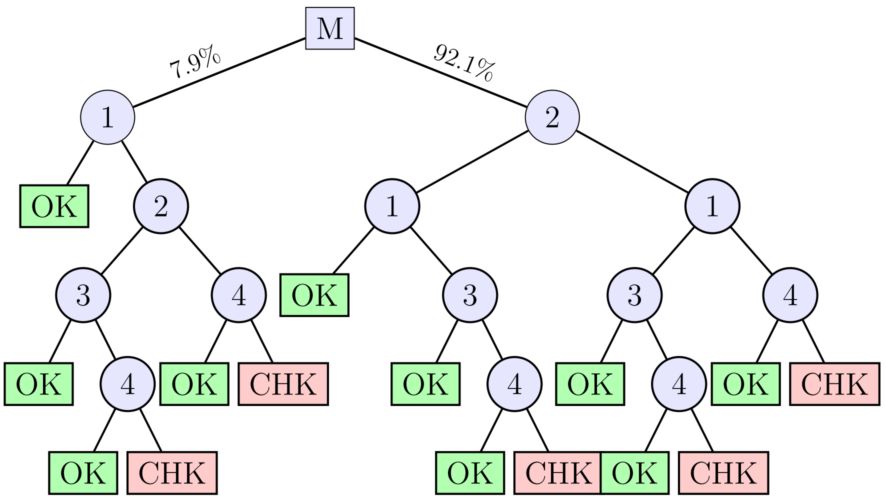

Figure 8.1: Optimal mixed decision tree.

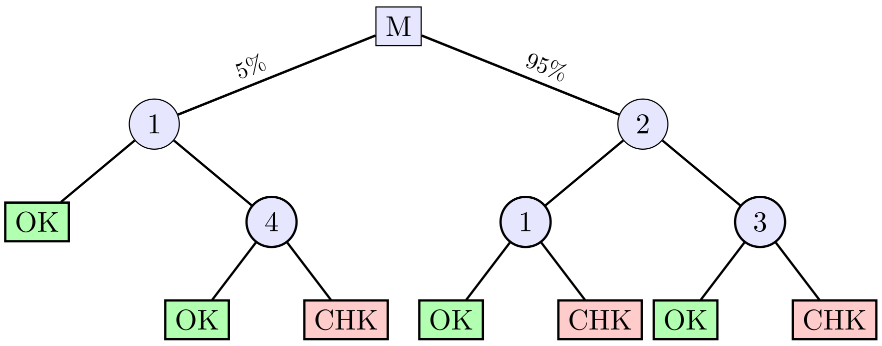

Figure 8.2: Optimal decision tree of depth 2.

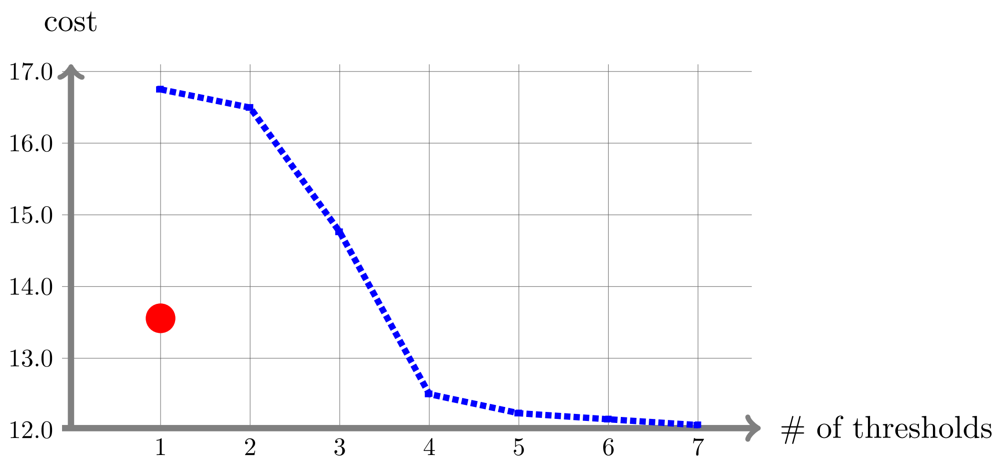

Figure 8.3: Optimal cost for different number of thresholds.

Number of					CPU time	
ranges per test	constraints	variables	nonzeros	iterations	setup (sec)	solver (sec)
2	318	1264	6960	17	0.05	0.06
3	1810	5424	50640	25	0.20	0.58
4	5918	15776	209056	32	3.66	3.58
5	14658	36640	630240	36	30.81	23.59
6	30622	73489	1518794	45	147.88	160.22
7	56978	132945	3276170	53	487.38	1975.30

Table 8.1: Problem sizes and computational times for different number of ranges.

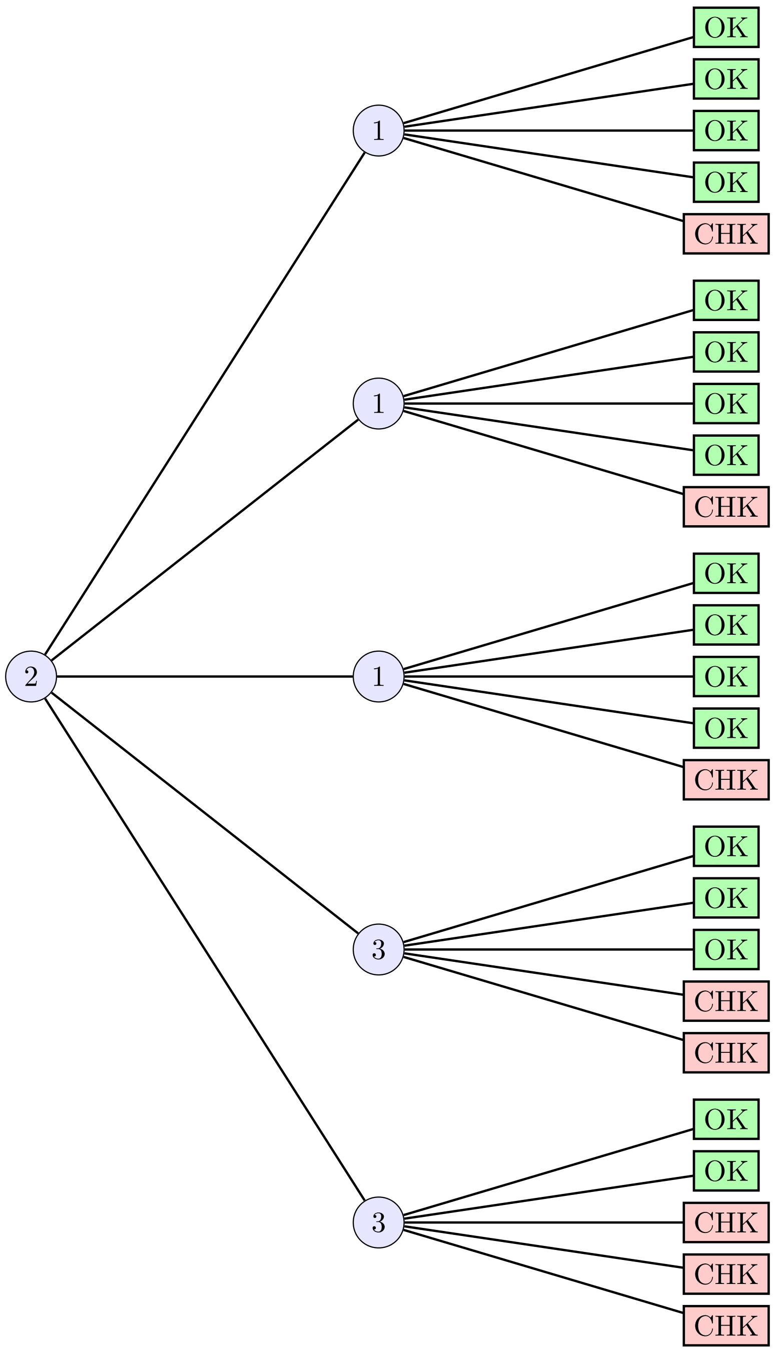

Figure 8.4: 24% of the containers follow this decision tree as a part of optimal inspection strategy for decision trees of depth 2 and 5 ranges.

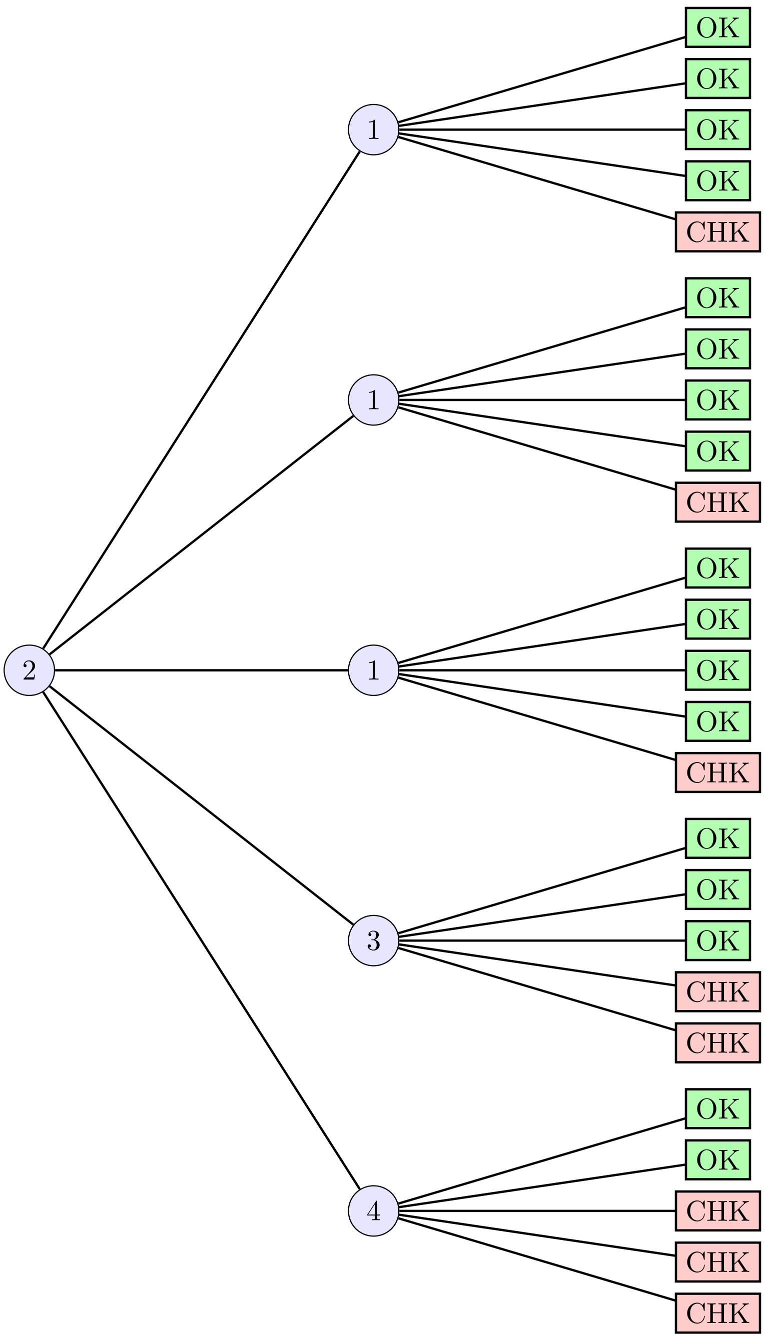

Figure 8.5: 76% of the containers follow this decision tree as a part of optimal inspection strategy for decision trees of depth 2 and 5 ranges.

first to test 4, it costs at least \$176 per container to inspect all the containers, whereas for our solution it costs only \$28 per container (six times less). Also, although misclassification costs can be specified in Weka, the current version of Weka does not take them into account.

8.2 Classification Results for Real-Life Data

Linear programming can be applied to the traditional learning model[1] and tested using cross-validation, where the input (distributions) can be derived from classical machine learning input (values of attributes).

We apply our model to Wisconsin Breast Cancer, Liver Disorders, and Pima Indians Diabetes real-life databases from UC Irvine Machine Learning Repository ([6]) and compare our results with J48 decision tree algorithm in Weka. The Credit Approval database has 15 attributes, and the details on attributes for two other databases are listed in Tables 8.2 and 8.3.

We have 345 observations for Liver Disorders and 768 observations for Pima Indians Diabetes databases. Wisconsin Breast Cancer database has 699 observations, and some observations have missing data. We delete from the study all the observations with missing data, which results in 683 observations for the Breast Cancer database. To compare the results we use our model to maximize the classification power, where for every strategy y *classification power* $\kappa(y)$ is the fraction of correctly classified observations; that is,

$$\kappa(y) = \sum_{\substack{\pi \in \mathcal{P} \\ CHK \in \pi}} \frac{y_\pi p_0 b(\pi)}{p_0 b(\pi) + (1-p_0) g(\pi)} + \sum_{\substack{\pi \in \mathcal{P} \\ OK \in \pi}} \frac{y_\pi (1-p_0) g(\pi)}{p_0 b(\pi) + (1-p_0) g(\pi)},$$

where p_0 is the fraction of positive observations.

Let us note that at least two different formulas have been used in the literature to define the classification power (see, e.g., [40] for an alternative formula). Because we compare our results with the results in Weka, we use the formula that is used in Weka.

One of the assumptions in our model is that the test results are independent of each other. To test whether this assumption holds, we calculate the correlation matrices (see,

[1]The application of this method to classification was also suggested by Noam Goldberg.

Attribute	Description
1	Mean corpuscular volume (blood test)
2	Alkaline phosphotase (blood test)
3	Alamine aminotransferase (blood test)
4	Aspartate aminotransferase (blood test)
5	Gamma-glutamyl transpeptidase (blood test)
6	Number of half-pint equivalents of alcoholic beverages drunk per day

Table 8.2: Description of Liver Disorder database attributes.

Attribute	Description
1	Number of times pregnant
2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3	Diastolic blood pressure (mm Hg)
4	Triceps skin fold thickness (mm)
5	2-Hour serum insulin (mu U/ml)
6	Body mass index (weight in kg/(height in m)^2)
7	Diabetes pedigree function
8	Age (years)

Table 8.3: Description of Pima Indians Diabetes database attributes.

Attribute	1	2	3	4	5	6
1	1	0.044103	0.147695	0.187765	0.222314	0.31268
2	0.044103	1	0.076208	0.146057	0.13314	0.100796
3	0.147695	0.076208	1	0.739675	0.503435	0.206848
4	0.187765	0.146057	0.739675	1	0.527626	0.279588
5	0.222314	0.13314	0.503435	0.527626	1	0.341224
6	0.31268	0.100796	0.206848	0.279588	0.341224	1

Table 8.4: Correlation matrix for Liver Disorders data.

e.g., Table 8.4 for Liver Disorders data). Because some of the test results are highly correlated, for example, the results of tests 3 and 4 for Liver Disorders, we modify our model to incorporate the dependency of the test results. When independency assumption holds we use the formula $Pr(\cap A_i) = \Pi Pr(A_i)$ for all events A_i, or in our case test results. In case of dependency, the above formula does not always hold, and we calculate distributions taking into account all the test results simultaneously (see example below).

To obtain a distribution (histogram) for our model from the data for each attribute, we select the smallest interval that contains all the values for positive and negative observations. Then, depending on the number of thresholds that we use in the model, we select one by one a threshold that divides the largest interval by half starting from the left. To obtain a histogram, we calculate the percentage of observation for each combination of ranges, attributes, and type of observations.

8.2.1 Example

Consider a simple example with two attributes and one threshold for each attribute. Given the data in Table 8.5 (two negative and three positive observations), we calculate

$$\begin{aligned}
g((1,1),0) &= 1,\\
g((1,2),0) &= 0,\\
g((2,1),0) &= 1/2,\\
g((2,2),0) &= 1/2,\\
b((1,1),1) &= 1/3,\\
b((1,2),1) &= 2/3,\\
b((2,1),1) &= 1/3,\\
b((2,2),1) &= 2/3,\\
g((1,1),(2,1),0) &= g((2,1),(1,1),0) = 1/2,\\
g((1,1),(2,2),0) &= g((2,2),(1,1),0) = 1/2,\\
g((1,2),(2,1),0) &= g((2,1),(1,2),0) = 0,\\
g((1,2),(2,2),0) &= g((2,2),(1,2),0) = 0,\\
b((1,1),(2,1),1) &= b((2,1),(1,1),1) = 1/3,
\end{aligned}$$

$$\begin{aligned} b((1,1),(2,2),1) &= b((2,2),(1,1),1) &= 0, \\ b((1,2),(2,1),1) &= b((2,1),(1,2),1) &= 0, \\ b((1,2),(2,2),1) &= b((2,2),(1,2),1) &= 2/3, \end{aligned}$$

where, for example, $b((1,2),(2,2),1)$ is the fraction of positive observations with the first and second attributes in the second range. We used the dependency assumption here, for example, $b((1,2),(2,1),1) = 0$, while $b((1,2),1) = 2/3$ and $b((2,1),1) = 1/3$; in other words $b((1,2),(2,1),1) \neq b((1,2),1)b((2,1),1)$.

8.2.2 Numerical results

We apply our model using 10-folding cross-validation and compare with the result in Weka following these five steps.

Step 1. We set a maximum decision tree depth that is less than or equal to the number of attributes; that is, $k \leq n$. In our study, we choose $k = 3$ for every dataset. We divide the dataset into 10 subsets.

Step 2. We set aside one tenth of the dataset for testing, and use the rest of the dataset for training. Using only training data we select the smallest interval that contains all the values for positive and negative observations. We set a number of thresholds for each dataset to divide the attribute values into a small number of ranges; for example, we choose two thresholds for Pima Indians Diabetes (PID) dataset, thus dividing all the attribute values into three ranges. We select the thresholds one by one to divide the interval of the largest size by half starting from the left. For example, Table 8.6 shows the thresholds for attribute 4 from the PID dataset for up to seven thresholds, where the first 691 out of 768 observations were used as a training set, and $[0, 99]$ is the smallest interval that contains all the values of the fourth attribute for the training set. Because we choose two thresholds for PID dataset in our study, we choose ranges $[0, 24.75]$, $[24.75, 49.5)$ and $(49.5, 99]$ for attribute 4. Note that this threshold selection procedure is expected to be robust, because there is a high probability that the selected thresholds are always the same. We calculate the distribution values $b(\pi)$ and $g(\pi)$ for each possible path of up to length three for positive and negative observations.

Step 3. We maximize the classification power for the training set over the polytope of at most k-depth decision trees using the data generated in Step 2 and find an optimal decision tree:

$$\max_{\mathbf{y}\in P} \sum_{\substack{\pi\in\mathcal{P} \\ CHK\in\pi}} \frac{y_\pi p_0 b(\pi)}{p_0 b(\pi) + (1-p_0)g(\pi)} + \sum_{\substack{\pi\in\mathcal{P} \\ OK\in\pi}} \frac{y_\pi (1-p_0) g(\pi)}{p_0 b(\pi) + (1-p_0)g(\pi)}.$$

Step 4. We apply the optimal decision tree to the testing set and calculate the number of correctly classified instances for testing set. For example, for LDD dataset optimal decision tree for the training set was applied to the testing set consisting of the first 34 observations of the whole dataset and 25 observations were correctly classified.

Step 5. We repeat Steps 2-4 for every testing set (having a different testing set in every run, so that after 10 runs every observation is once a part of testing set and nine times in training set). We calculate the testing classification power as a fraction of correctly classified observations adding the numbers from the 10 runs and dividing by the number of observations. For example, for LDD dataset we report in Table 8.7 the training and testing classification power, which results in 70.14% of correctly classified observation. We randomly reshuffle the data and repeat the 10-folding cross-validation (Steps 1-5) several times and calculate the average classification power.

We report the results for maximum decision tree depths 2 and 3 in Table 8.8 along with the classification power obtained by J48 decision tree algorithm in Weka (see [40]) using the same datasets. We have used the Xpress-MP package formulating our model in the Mosel modeling language, and solving the large scale linear programming problems by the Newton-barrier method. We ran the experiments on a Dell Inspiron 1200, with a 1.30GHz Intel(R) Celeron(R) processor and 248MB memory.

Optimal decision trees built in Weka are all binary and usually larger, and for two out of three datasets even larger than the number of attributes, because J48 algorithm allows us to revisit the same test (obviously with a different threshold). Although the classification power for our model is lower than classification power obtained by J48 for all three datasets when maximum decision tree depth is 2, it is higher when decision tree depth is 3. Note that we avoided a potential problem of overfitting by limiting the depth of considered decision trees. We based our application on Theorem 3 on polyhedral characterization of decision trees, that applies even if we consider shape restrictions, such as depth of the tree.

Observation/Attribute	1	2	Type
1	1	2	0
2	1	1	0
3	1	1	1
4	2	2	1
5	2	2	1

Table 8.5: Example data for two attributes and two ranges.

Number of thresholds	Thresholds
1	49.5
2	24.75, 49.5
3	24.75, 49.5, 74.25
4	24.75, 37.125, 49.5, 74.26
5	24.75, 37.125, 49.5, 61.875, 74.27
6	12.375, 24.75, 37.125, 49.5, 61.875, 74.28
7	12.375, 24.75, 37.125, 49.5, 61.875, 74.29, 86.625

Table 8.6: The thresholds for attribute 4 from the PID dataset for up to seven thresholds.

Testing set	Classification power for training set	Classification power for testing set
1	76.43%	73.53%
2	76.61%	74.29%
3	76.99%	73.53%
4	75.80%	82.85%
5	77.84%	64.71%
6	77.31%	54.29%
7	75.37%	76.47%
8	77.76%	65.71%
9	75.91%	82.35%
10	77.91%	54.29%

Table 8.7: Classification power for training and testing sets for one of the 10-folding cross-validation runs for LDD.

Dataset	Breast Cancer		Liver Disorders		Pima Indians Diabetes	
Number of attributes	9		6		8	
Decision tree depth (J48)	5		8		9	
Classification power, % (J48)	**93.79**		**63.65**		**76.13**	
Classification power, % (LP)	**91.12**	**94.61**	**60.34**	**70.26**	**70.29**	**77.17**
Decision tree depth (LP)	2	3	2	3	2	3
Number of ranges (LP)	3	3	3	3	3	3
Number of constraints (LP)	6760	142840	2890	35290	5290	96010
Number of variables (LP)	13520	285680	5780	70580	10580	192020
Number of nonzeros (LP)	92502	2759860	39164	681628	72484	1872860
Number of iterations (LP)	7	11	7	9	7	10
Computational time, sec (LP)	85	3300337	7	1399	35	36300

Table 8.8: Classification results for real-life data for Linear Programming model (LP) when maximum decision tree depths are 2 and 3, and the classification power obtained by J48 decision tree algorithm in Weka.

Chapter 9
Conclusions

The results reported herein illustrate several important features of the test scheduling problem. We have shown that formulating the problem in terms of a polyhedron in a very high dimensional space, whose points represent combinations of possible paths of an object through the test network, leads to a large scale linear programming problem that can be solved efficiently.

Because the choice of the thresholds is subjective, there are many different methods for choosing them. One method is based on the distributions of the test results. Another method is based on the values of the ratio of the two conditional distributions. Also, the thresholds may be randomly generated. We expect that some methods may increase the strategy performance.

Also, increasing the number of the thresholds results in better overall performance, and it is presumably asymptotic to the result that would be obtained with infinite number of the thresholds.

Although we assumed conditional independence of the test results, the analysis can be extended to situations in which the test results are stochastically dependent, for both the good and bad objects.

The cost of false negative and of false positive observations can be included in the formulation of the problem. We considered in this study only the cost of operation, because usually they are concrete and easily documented. The costs of various types of errors are typically matters of debate and conjecture.

In this study we considered two types of the observations (positive and negative), and two types of decisions (values of the function), but the analysis can be extended for any reasonable number of different types of objects and decisions.

Chapter 10

VC-dimension of Horn Functions

10.1 Introduction

Let us consider a set of n Boolean variables $V = \{x_1, ..., x_n\}$, where x_i is a *Boolean variable* if $x_i \in \{0, 1\},\ i = 1, ..., n$. Let $L = \{x_1, \overline{x}_1, ..., x_n, \overline{x}_n\}$ denote the set of corresponding *literals*, where $\overline{x}_i = 1 - x_i,\ i = 1, ..., n$ are the *complemented* (or *negated*) variables. A *term* (or *monomial*) is the conjunction of a subset of the literals, e.g. $\phi = x_1 x_3 \overline{x}_8$. A *disjunctive normal form* (or DNF in short) is a disjunction of terms, e.g. $\psi = x_1 x_3 \overline{x}_8 \vee x_2 \vee \overline{x}_7 \overline{x}_3$. It is well-known that any Boolean function f: $\mathbb{B}^n \rightarrow \mathbb{B}$ has a DNF representation, where $\mathbb{B} = \{0, 1\}$.

A point $x = (x_1, ..., x_n) \in \mathbb{B}^n$ is a *true point* of the Boolean function f if $f(x) = 1$, and if $f(x) = 0$ we call x a *false* point. Let us denote by $T(f)$ the set of true points of Boolean function f, and by $F(f)$ the set of false points of f. Let G be a class of Boolean functions $g : \mathbb{B}^n \rightarrow \mathbb{B}$. A set $S \subseteq \mathbb{B}^n$ is said to be *shattered* by G if for any partition of S into two subsets T and F there exists function $g \in G$ for which $T \subseteq T(g)$ and $F \subseteq F(g)$. The *Vapnik-Chervonenkis dimension* of G, denoted by $VCdim(G)$ is defined as the maximum cardinality of a subset $S \subseteq \mathbb{B}^n$ that is shattered by G.

The notion of VC-dimension was introduced in [75] and can be defined for any class of functions analogously. It is used in the evaluation of the bounds of the sample size required for learning in PAC model (see, e.g., [5, 9, 74]). For some classes of Boolean functions VC-dimension is already known, for example, for monomials and threshold functions [4]. The complexity of the problem of finding the VC-dimension for given class of functions is addressed in [66].

Example 3

$$d_P = VCdim(P) = \binom{n}{\lfloor \frac{n+1}{2} \rfloor}, \tag{10.1}$$

where P is a class of positive functions. A Boolean function is positive if it has a DNF representation in which each variable appears only uncomplemented. To find the VC-dimension of positive functions we use the condition that a Boolean function f is positive if and only if for all $x, y \in \mathbb{B}^n$ the implication $x \geq y \Rightarrow f(x) \geq f(y)$ holds. Let us denote by S any subset that is shattered by P. If S contains any two points that can be compared, for example, $x, y \in S$ such that $x \geq y$, then no such positive function exists for which x is a false point and y is a true point. Therefore S consists of such a set of points that no two of them can be compared. Hence the maximum possible cardinality of S (known as Sperner's Theorem) is $\binom{n}{\lfloor \frac{n+1}{2} \rfloor}$, implying that

$$VCdim(P) \leq \binom{n}{\lfloor \frac{n+1}{2} \rfloor}. \tag{10.2}$$

To show that

$$VCdim(P) = \binom{n}{\lfloor \frac{n+1}{2} \rfloor} \tag{10.3}$$

let us consider a set S consisting of vectors $x \in \mathbb{B}^n$ such that x has exactly $\lfloor \frac{n+1}{2} \rfloor$ components equal to 1. For any partition of S into two subsets T and F we construct a DNF expression ϕ such that the number of terms in ϕ equals to the number of vectors in T and there is a one-to-one correspondence between the terms of ϕ and the vectors from T. For each vector x from T the corresponding term of ϕ is the conjunction of $\binom{n}{\lfloor \frac{n+1}{2} \rfloor}$ variables x_j such that $x_j = 1$. Finally, let us observe that T is a subset of true points of ϕ and F is a subset of false points of ϕ.

10.2 Main results

Let us denote by H the class of Horn functions $h : \mathbb{B}^n \to \mathbb{B}$, where a Boolean function is *Horn* if it has a DNF representation with at most one complemented variable in each term. Horn functions were introduced in [42] and are extensively used in artificial intelligence (see, e.g., [3]). Let us denote by d_H the VC-dimension of H.

Our main result is the following.

Theorem 4 *$d_H = \Theta(d_P)$, where $d_P = VCdim(P) = \binom{n}{\lfloor \frac{n+1}{2} \rfloor}$ (see Example above).*

To state even stronger results let us consider the class H_k of k-quasi-Horn functions (see e.g. [3]). For any nonnegative integer constant k let us define *k-quasi-Horn functions* h: $\mathbb{B}^n \to \mathbb{B}$ as a class of Boolean functions that have a DNF representation with at most k complemented variables in each term. The above definition implies that the class of k-quasi-Horn functions includes positive ($k = 0$) and Horn ($k = 1$) functions. Let us denote by d_k the VC-dimension of H_k. Theorem 5 is a generalization of the Theorem 4.

Theorem 5 *$d_k = \Theta(d_P)$, when $k \ll n$.*

Among other generalizations of Horn functions the most popular is the class G_q of q-Horn functions, introduced in [11], but we will use the definition given in [10]. For every subset of the literals $A \subseteq L$ we denote by $\overline{A}$ a set of complemented literals of the set A. A Boolean function h: $\mathbb{B}^n \to \mathbb{B}$ is *q-Horn* if it has a DNF representation ϕ for which there is a subset of the literals $A \subseteq L$ such that $A \cap \overline{A} = \emptyset$ and for any term $t \in \phi$ the inequality

$$\frac{1}{2}\left|Y \cap (L \setminus (A \cup \overline{A}))\right| + |Y \cap A| \leq 1 \tag{10.4}$$

holds, where Y is a set of literals that that are involved in term $t \in \phi$. Our first observation is that if $A = \overline{V}$, or, in other words, if A includes all possible complemented literals then h is Horn. Our second observation is that if we denote by $G_{\overline{q}}$ those q-Horn functions for which A contains only complemented literals then any function from $G_{\overline{q}}$ will be 2-quasi-Horn. Hence we can conclude that $H_1 \subseteq G_{\overline{q}} \subseteq H_2$, implying the following Corollary of the Theorem 5.

Corollary 1 *$VCdim(G_{\overline{q}}) = \Theta(d_P)$.*

10.3 Proof of Theorem 4

The proof of Theorem 4 is based on the following observations.

Given two Boolean vectors $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ we define their conjunction $x \wedge y = (x_1 \wedge y_1, ..., x_n \wedge y_n)$ as the component-wise conjunction. We say that a set $S \subseteq \mathbb{B}^n$ is closed under conjunction if for any $x, y \in S$ we have $x \wedge y \in S$.

Let us recall a famous McKinsey's theorem.

Theorem 6 *([58]) A Boolean function h is Horn if and only if the set of its false points is closed under conjunction.*

For every set $Q \subseteq \mathbb{B}^n$ we denote by $x(Q)$ the intersection of set Q:

$$x(Q) = \bigwedge_{y \in Q} y. \tag{10.5}$$

We say that a set $W \subseteq \mathbb{B}^n$ satisfies the *Non-Intersection Condition* if for any $Q \subseteq W$ such that $Q \neq \emptyset$ we have $x(Q) \notin W \setminus Q$. We say that a set $W \subseteq \mathbb{B}^n$ satisfies the *Designated Zero Condition* if for all $x \in W$ either $x = (1, ..., 1)$ or there exists $k \in \{1, ..., n\}$ such that $x_k = 0$ and for all $y \in W,\ x \neq y$ the inequality

$$y_k + \sum_{j:x_j=1} (1 - y_j) > 0 \tag{10.6}$$

holds. In other words, any vector from the set W either has all the coordinates equal to 1 or has a "designated" zero and the combination of this zero and all the ones of this vector at exactly those positions is unique among all vectors from W.

Lemma 1 *Let $W \subseteq \mathbb{B}^n$. Then W satisfies the Non-Intersection Condition if and only if W satisfies the Designated Zero Condition.*

Proof. ($\Rightarrow$) Let us observe that if the Designated Zero Condition does not hold for W then there is a vector x in W such that some components of x are zeros and any combination of any zero and all ones of x can be found in at least one other vector of W. Let us denote by W_x the set of vectors in which such combinations can be observed. We can observe that

$$x = \bigwedge_{y \in W_x} y \in W \setminus W_x, \tag{10.7}$$

implying that the Non-Intersection Condition does not hold.

($\Leftarrow$) If the Non-Intersection Condition does not hold for W then there is a vector $y \in W$ such that y equals to the conjunction of some other vectors from W. Because any combination of any zero and all ones of y can be found in one of the vectors that participate in conjunction it is impossible to find a designated zero for y. $\square$

Now we are ready to prove our main theorem.

Proof of Theorem 4. Since any positive function is Horn we have $d_H \geq d_P$. To complete the proof we must show that there exists a constant $c > 0$ such that $d_H \leq cd_P$.

Let us denote by $S_H \subseteq \mathbb{B}^n$ a set of maximum cardinality that is shattered by the family of Horn functions H. By Theorem 6 it follows that S_H satisfies the Non-Intersection Condition. To see this let us observe that if there exists $Q \subseteq S_H$ such that

$$x(Q) := \bigwedge_{y \in Q} y \in S_H \setminus Q, \tag{10.8}$$

then according to Theorem 6 no such Horn function exists for which all vectors $y \in Q$ are false points and $x(Q)$ is a true point.

Since by Lemma 1 S_H satisfies the Non-Intersection Condition it follows that each vector from S_H is either $(1, ..., 1)$ or has its designated zero. Let $S = S_H$ if S_H does not include $(1, ..., 1)$; otherwise let $S = S_H \setminus \{(1, ..., 1)\}$. W.l.o.g. let us assume that n is divisible by 2 (if not, we can introduce additional variable $x_{n+1} = 1$ to proceed with our analysis and then delete it). Let us divide S into two sets SL and SR depending whether designated zero of the vector is in its left or right half. Let us forget for a moment about designated zeros and note that the maximum number of distinct left halves of vectors in SL is at most $2^{n/2}$. Consider a collection U of any vectors that have the same left half. Lemma 1 implies that the combination of all ones and designated zero of any vector should be unique. Because the left halves of the vectors from U are the same, to be unique their right halves should form an antichain, or in other words should be noncomparable. The same analysis can be repeated for SR. Thus, using Stirling's approximation we can conclude from the above that for some constants $c_1,\ c_2 > 0$ we have

$$d_H \leq 1 + 2^{n/2} \binom{n/2}{\lfloor \frac{n+2}{4} \rfloor} \leq 1 + 2 \cdot 2^{n/2} c_1 \frac{2^{n/2}}{\sqrt{n/2}} = 1 + 2\sqrt{2} c_1 \frac{2^n}{\sqrt{n}} \leq c_2 d_P. \tag{10.9}$$

□

10.4 Proof of Theorem 5

To prove Theorem 5 we generalize the results of Section 10.3.

For any $x \in \mathbb{B}^n$ let us denote by z_x the number of zero components of x, or in other words

$$z_x = n - \sum_{j=1}^{n} x_j. \tag{10.10}$$

For any $W \subseteq \mathbb{B}^n$ we say that W satisfies the *k-Designated Zeros Condition* if for all $x \in W$ there exists a set of at most k distinct indices $I \subseteq \{1, ..., n\},\ |I| = \min\{k, z_x\}$ such that for all $i \in I$ $x_i = 0$ and for all $y \in W,\ y \neq x$ the inequality

$$\sum_{j:j \in I} y_j + \sum_{m:x_m=1} (1 - y_m) > 0 \tag{10.11}$$

holds. In other words, each vector x from W has $\min\{k, z_x\}$ designated zeros and the combination of all these zeros (if any) and all the ones of this vector at exactly these positions is unique among all vectors from W.

Lemma 2 *Let $W \subseteq \mathbb{B}^n$. Then W is shattered by the class H_k if and only if W satisfies the k-Designated Zeros Condition.*

Proof. ($\Rightarrow$) If the k-Designated Zeros Condition does not hold for W, then there is a vector $x \in W$ such that any combination of any at most k zeros and all ones of this vector can be found in at least one other vector of W. Let us denote by W_x the set of vectors in which such combinations can be observed and assign $T = \{x\}$ and $F = W \setminus T$. It is impossible to find $h \in H_k$ for which T is a subset of its true points, because for any term t in a DNF of such a function $h \in H_k$ if $t(x) = 1$ then we also have that $t(w) = 1$ for $w \in W_x \subseteq F$.

($\Leftarrow$) By construction. For any partition of W into two subsets T and F let us denote by ϕ a DNF representation of $h \in H_k$. The number of terms in ϕ equals to the number of vectors in T, and there is a one-to-one correspondence between the terms of ϕ and the vectors from T. For each vector x from T the corresponding term of ϕ is the conjunction of uncomplemented and complemented variables such that a variable x_j is uncomplemented if $x_j = 1$ and a variable x_j is complemented if x_j is a designated zero. To complete our proof let us observe that T is a subset of true points of ϕ and F is a subset of false points of ϕ.□

Proof of Theorem 5. Because for any nonnegative integer constant k any positive function is k-quasi-Horn, we have $d_k \geq d_P$. To complete the proof we must show that there exists a constant $c > 0$ such that $d_k \leq c d_P$.

Let $S(k) \subseteq \mathbb{B}^n$ be a set of maximum size that is shattered by H_k. According to Lemma 2, each vector from $S(k)$ has at most k designated zeros. Let us consider a set $S = S(k) \backslash S_0$, where $S_0 \subseteq S(k)$ is a set of vectors that have less than k zero components. Note that

$$|S_0| \leq 1 + \binom{n}{2} + ... + \binom{n}{k-1} \leq c_0 n^{k-1} \tag{10.12}$$

for some constant $c_0 > 0$.

W.l.o.g. let us assume that n is divisible by $k+1$ (if not, we can introduce m additional variables $x_{n+1} = ... = x_{n+m} = 1$, where $0 < m \leq k$ and $n+m$ is divisible by $k+1$, proceed with the analysis, and then delete them). Let us divide each vector $x \in S$ by $k+1$ parts. Let the sets $S_1, ..., S_k \subseteq S$ be a partition of S, where $x \in S_i$ if the designated zeros of x are located in exactly i out of $k+1$ different parts of x. Those parts of vectors that contain (do not contain) any designated zeros we will call designated (nondesignated) parts respectively. Subvectors obtained by deletion of nondesignated (designated) parts will be called designated (nondesignated) subvectors respectively. For any $i = 1, ..., k$ there are $\binom{k+1}{i}$ different ways of locating designated zeros in i different parts of vector. The number of distinct designated subvectors of particular i designated parts is at most $2^{in/(k+1)}$. And finally, what if a collection U of vectors has the same i designated parts with the same designated subvectors? According to Lemma 2 the combination of all ones and designated zero of any vector should be unique, and the only way to be unique is that the nondesignated subvectors of their nondesignated parts forms an antichain. Thus, using Stirling's approximation we can conclude from the above that for some constants $c_0,\ c_1,\ c_2 > 0$ we have

$$d_k \leq c_0 n^{k-1} + \sum_{i=1}^{k} \binom{k+1}{i} 2^{in/(k+1)} \binom{n(1-\frac{i}{k+1})}{\frac{n}{2}(1-\frac{i}{k+1})} \leq c_0 n^{k-1} + c_1 \frac{2^n}{\sqrt{n}} \leq c_2 d_P.$$

$\square$

Chapter 11
On k-quasi-Horn Functions

11.1 Introduction

Recall that for any nonnegative integer constant k a Boolean function h: $\mathbb{B}^n \to \mathbb{B}$ is *k-quasi-Horn* if it has a DNF representation with at most k negative literals in each term. We denote by H_k the class of k-quasi-Horn functions, our main interest in this chapter. Note that, for example, the class of 2-quasi-Horn functions includes both positive ($k = 0$) and Horn ($k = 1$) functions. For any Boolean function f its dual f^d is defined as $f^d(x) = \overline{f(\overline{x})}$ for all $x \in \mathbb{B}^n$, where $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$. A function f is *bidual Horn*, if both f and f^d are Horn.

Usually, any given class of Boolean functions can be described in many different ways. For instance, we can define Boolean linear functions as the class of functions that have a DNF representation with at most one literal in each term, or we can say that a Boolean function f is linear if for any $x,\ y \in \mathbb{B}^n$ the equality $f(x) \vee f(y) = f(xy) \vee f(x \vee y)$ holds (see, e.g., [32]). The description of many important classes of Boolean functions by algebraic identities is considered, for example, in [26, 25, 31, 32], and a criterion for the classes of Boolean functions that can be characterized by algebraic identities is derived in [32].

In this chapter we consider the classes of k-quasi-Horn functions, which generalize the class of Horn functions. The importance of Horn functions and McKinsey's theorem, the criterion of Horn functions, cannot be overstated in the theory of Boolean functions and artificial intelligence. In particular, McKinsey's theorem was one of the crucial factors that allowed us to obtain a good polynomial time algorithm for learning a Horn function in [3]. One of the most popular generalizations of the Horn functions is the class of q-Horn functions, introduced in [11]. Algebraic characterization of q-Horn functions is addressed in [43].

Section 11.2 provides definitions and notations that we use in the chapter. In Section 11.3, we state and prove a criterion for k-quasi-Horn functions that generalizes McKinsey's theorem. In Section 11.4, we state and prove necessary and sufficient conditions for function to be bidual k-quasi-Horn.

11.2 Definitions and Notations

For each Boolean vector $x \in \mathbb{B}^n$ we denote $\mathcal{O}(x) = \{i \mid x_i = 1\}$ and $\mathcal{Z}(x) = \{i \mid x_i = 0\}$. Given two Boolean vectors $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ we denote their component-wise conjunction by $x \wedge y = (x_1 \wedge y_1, \dots, x_n \wedge y_n)$, or for short $xy = (x_1y_1, \dots, x_ny_n)$. We say that a set $S \subseteq \mathbb{B}^n$ is *closed under conjunction* if for any $x, y \in S$ we have $x \wedge y \in S$.

For any set $S \subseteq \mathbb{B}^n$ such that $S \neq \emptyset$ let us denote by

$$c(S) = \bigwedge_{y \in S} y$$

the component-wise conjunction of all vectors of S. We will say that a set $S \subseteq \mathbb{B}^n$ satisfies the *k-Zeros Condition* if for any $Q \subseteq \mathcal{Z}(c(S))$ such that $|Q| = k$ there exists $y \in S$ such that $Q \subseteq \mathcal{Z}(y)$. If a set $S \subseteq \mathbb{B}^n$ satisfies the k-Zeros Condition, we say that $c(S)$ is a *k-zeros-conjunction* (or *k-zeros-intersection*) of set S. For example, vector $x = (0, 0, 0)$ is not a 2-zeros-conjunction of vectors $u = (1, 0, 0)$ and $v = (0, 1, 0)$, since $x_1 = x_2 = 0$ and $u_1 + u_2 = v_1 + v_2 = 1$. Let us remark that 1-zeros-conjunction of vectors coincides with conjunction of vectors. Note that if S satisfies the k-Zeros Condition then there exists a set $S' \subseteq S$, $|S'| < n^k$ such that $c(S') = c(S)$ and S' also satisfies k-Zeros Condition. Furthermore, we can check whether a given set S satisfies a k-Zeros Condition and find a minimal $S' \subseteq S$ with $c(S') = c(S)$ if it does in time $n^k|S|$, which is polynomial for a constant k. Finally, we say that a set $S \subseteq \mathbb{B}^n$ is *closed under k-zeros-conjunctions* if for all sets $Q \subseteq S$ which satisfy the k-Zeros Condition we have $c(Q) \in S$.

By analogy, similar notions are formulated for disjunction of vectors. The component-wise disjunction of two Boolean vectors $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ is $x \vee y = (x_1 \vee y_1, \dots, x_n \vee y_n)$. A set $S \subseteq \mathbb{B}^n$ is *closed under disjunction* if for any $x, y \in S$ we have $x \vee y \in S$.

For any set $S \subseteq \mathbb{B}^n$ such that $S \neq \emptyset$ we denote by

$$d(S) = \bigvee_{y \in S} y$$

the componentwise disjunction of all vectors of S. We will say that a set $S \subseteq \mathbb{B}^n$ satisfies the *k-Ones Condition* if for any $Q \subseteq \mathcal{O}(d(S))$ such that $|Q| = k$ there exists $y \in S$ such that $Q \subseteq \mathcal{O}(y)$. If a set $S \subseteq \mathbb{B}^n$ satisfies the k-Ones Condition, we say that $d(S)$ is a *k-ones-disjunction* (or *k-ones-union*) of set S. For example, vector $x = (1,1,1)$ is not a 2-ones-disjunction of vectors $u = (1,0,0)$ and $v = (0,1,1)$, since $x_1 = x_2 = 1$ and $u_1u_2 = v_1v_2 = 0$. Note that a 1-ones-disjunction of vectors defines a disjunction of vectors. A set $S \subseteq \mathbb{B}^n$ is *closed under k-ones-disjunctions* if for all sets $Q \subseteq S$ that satisfy the k-Ones Condition we have $d(Q) \in S$.

Let us denote by $P(t)$ and $N(t)$ respectively the set of indices of positive and negative literals in the term t, and by $V(t) = P(t) \cup N(t)$ the set of all variable indices in the term t. For any $s \geq 2$ terms $t_{i_1}, \ldots, t_{i_s}$ let us denote by $t^+_{i_1,i_2,\ldots,i_s}$ the positive term such that

$$P(t^+_{i_1,i_2,\ldots,i_s}) = P(t_{i_1}) \cup P(t_{i_2}) \cup \ldots \cup P(t_{i_s}),$$

and by $t^\pm_{i_1,i_2,\ldots,i_s}$ the positive term such that

$$P(t^\pm_{i_1,i_2,\ldots,i_s}) = V(t_{i_1}) \cup V(t_{i_2}) \cup \ldots \cup V(t_{i_s}).$$

For example, if $t_1 = x_1\overline{x}_2$ and $t_2 = x_2\overline{x}_3$, then $t^+_{1,2} = x_1x_2$ and $t^\pm_{1,2} = x_1x_2x_3$.

Let us denote by

$$\tilde{N}_{i_2,i_3,\ldots,i_s}(t_{i_1}) = N(t_{i_1}) \setminus (N(t_{i_2}) \cup N(t_{i_3}) \cup \ldots \cup N(t_{i_s}))$$

the set of indices of the unique (with respect to the terms t_{i_2}, t_{i_3}, $\ldots$, t_{i_s}) negated literals in the term t_{i_1}. To simplify the notations we will use $\tilde{N}(t_{i_1})$ instead of $\tilde{N}_{i_2,i_3,\ldots,i_s}(t_{i_1})$ when all the terms are defined without any ambiguity. We define a positive DNF

$$\hat{t}^\pm_{i_1,i_2,\ldots,i_s} = \bigwedge_{q \in \{1,2,\ldots,s\}} \Big(\Big(\bigwedge_{l \in P(t_{i_q})} x_l \Big) \wedge \Big(\bigvee_{l \in \tilde{N}(t_{i_q})} x_l \Big) \Big).$$

For example, if $t_1 = x_1\overline{x}_2\overline{x}_3\overline{x}_4$ and $t_2 = x_2\overline{x}_3\overline{x}_5\overline{x}_6$, then $\hat{t}^\pm_{1,2} = x_1x_2(x_2 \vee x_4)(x_5 \vee x_6)$.

Finally, a vector $x \in \mathbb{B}^n$ is a *true point of the Boolean function f* if $f(x) = 1$, and if $f(x) = 0$ we call x a *false point*. We denote by $T(f)$ the set of true points of Boolean function f, and by $F(f)$ the set of false points of f.

11.3 A Generalization of McKinsey's Theorem for k-quasi-Horn Functions

Let us recall McKinsey's theorem ([58]) from Section 10.3 (Theorem 6), where a Boolean function h is Horn if and only if the set of its false points $F(h)$ is closed under conjunction.

Let us formulate our main result for 2-quasi-Horn functions, and an easy generalization will follow.

Theorem 7 *A Boolean function h is 2-quasi-Horn if and only if the set of its false points is closed under 2-zeros-conjunctions.*

Proof. ($\Rightarrow$) Assume that the set of false points of a Boolean function h is not closed under 2-zeros-conjunctions or, in other words there exists a set $S \subseteq F(h)$ that satisfies the 2-Zeros Condition and $c(S)$ is a true point of h. In any DNF expression of h we must have a term t for which $t(c(S)) = 1$ and $t(y) = 0$ for all $y \in S$. Because S satisfies 2-Zeros Condition a term t has at least three negative literals, because otherwise there exists $y \in S$ such that $t(y) = 1$. Hence h cannot be 2-quasi-Horn.

($\Leftarrow$) Assume that the set of false points of a Boolean function h is closed under 2-zeros-conjunctions. We show that h is then 2-quasi-Horn. Consider an arbitrary true point $x \in T(h)$. Let us distinguish two cases:

Case (i) If x has at most two zero components. In this case define a term t_x as a minterm of x. Clearly, we have $t_x \leq h$ and t_x has at most two negative literals.

Case (ii) If x has at least three zero components. Let us denote by

$$S := \{y \in F(h) \mid y > x\}$$

the set of false points of h that are greater than x. We claim that there exist two indices $i \neq j$ such that $x_i = x_j = 0$ and for any vector $y \in S$ we have $y_i + y_j \geq 1$. Otherwise, the 2-Zeros Condition would hold for S contradicting the fact that $c(S)$ is a true point. Then define a term

$$t_x = \overline{w}_i \overline{w}_j \bigwedge_{k: x_k = 1} w_k.$$

Clearly, we have $t_x \leq h$.

To complete our proof we can observe that the same analysis can be repeated for any true point of h, implying that h can be represented by a DNF

$$h = \bigvee_{x \in T(h)} t_x,$$

which is a 2-quasi-Horn DNF. $\square$

Theorem 8 *A Boolean function h is k-quasi-Horn if and only if the set of its false points is closed under k-zeros-conjunctions.*

Proof. The proof of Theorem 8 is an easy generalization of the proof of Theorem 2.
($\Rightarrow$) Assume that the set of false points of a Boolean function h is not closed under k-zeros-conjunctions or, in other words there exists a set $S \subseteq F(h)$ that satisfies the k-Zeros Condition and $c(S)$ is a true point of h. In any DNF expression of h we must have a term t for which $t(c(S)) = 1$ and $t(y) = 0$ for all $y \in S$. Because S satisfies k-Zeros Condition a term t has more than k negative literals, because otherwise there exists $y \in S$ such that $t(y) = 1$. Hence h cannot be k-quasi-Horn.

($\Leftarrow$) Assume that the set of false points of a Boolean function h is closed under k-zeros-conjunctions. We show that h is then k-quasi-Horn. Consider an arbitrary true point $x \in T(h)$. Let us distinguish two cases:

Case (i) If x has at most k zero components. In this case define a term t_x as a minterm of x. Clearly, we have $t_x \leq h$ and t_x has at most k negative literals.

Case (ii) If x has at least $k+1$ zero components. Let us denote by

$$S := \{y \in F(h) \mid y > x\}$$

the set of false points of h that are greater than x. We claim that there exist k pairwise distinct indices $i_1, \ldots, i_k$ such that

$$x_{i_1} = \ldots = x_{i_k} = 0$$

and for any vector $y \in S$ we have

$$y_{i_1} + \ldots + y_{i_k} \geq 1.$$

Otherwise the k-Zeros Condition would hold for S contradicting the fact that $c(S)$ is a true point. Then define a term

$$t_x = \overline{w}_{i_1} \wedge \ldots \wedge \overline{w}_{i_k} \bigwedge_{j:x_j=1} w_j.$$

Clearly, we have $t_x \leq h$.

To complete our proof we can observe that the same analysis can be repeated for any true point of h, implying that h can be represented by a DNF

$$h = \bigvee_{x \in T(h)} t_x,$$

which is a k-quasi-Horn DNF. □

Let us remark that the class of all Boolean functions is partitioned by $n+1$ classes of k-Horn-functions, and we have necessary and sufficient conditions to recognize a k-quasi-Horn function.

11.4 Bidual k-quasi-Horn Functions

From the definition of bidual function and McKinsey's theorem we have the following corollary.

Corollary 2 *A Boolean function f is bidual k-quasi-Horn if and only if $F(f)$ is closed under k-zeros-conjunctions and $T(f)$ is closed under k-ones-disjunctions.*

The criteria for bidual Horn functions (Lemma 3) and DNFs (Lemma 4) formulated below can be found in [31]. We generalize these results and formulate necessary (Subsection 11.4.1) and sufficient (Subsection 11.4.2) conditions for bidual k-quasi-Horn functions and DNFs. All proofs are considered for $k = 2$, and proofs for $k > 2$ are analogous.

Lemma 3 ([31]) *Let f be a Horn function. Then the following statements are equivalent: (i) f is bidual; (ii) For every pair of Horn implicants t_i and t_j of f that have different negative literals, for example, $|N(t_i) \cup N(t_j)| = 2$, it holds that $t^{+}_{i,j} \leq f$; (iii) For every pair of Horn implicants t_i and t_j of f such that $|N(t_i) \cup N(t_j)| = 2$, it holds that $t^{\pm}_{i,j} \leq f$.*

Lemma 4 ([31]) *Let φ be a Horn DNF. Then φ represents a bidual Horn function if and only if*

$$t^{+}_{i,j} \leq \varphi \text{ (equivalently, } t^{\pm}_{i,j} \leq \varphi)$$

holds for all pairs of Horn terms t_i and t_j in ϕ such that $|N(t_i) \cup N(t_j)| = 2$. (Remark: This condition is different from Lemma 3 (ii) in that only terms t_i and t_j in φ are considered here.)

11.4.1 Necessary conditions

Lemma 5 *Let f be a 2-quasi-Horn function. If f is a bidual 2-quasi-Horn function, then for every three 2-quasi-Horn implicants t_i, t_j and t_m of f such that*

$$|N(t_q)| = |\tilde{N}(t_q)| \geq 1, \ q = i, j, m, \tag{11.1}$$

it holds that

$$t^{+}_{i,j,m} \leq f \text{ and } \hat{t}^{\pm}_{i,j,m} \leq f.$$

Proof. Assume that there is a vector b such that $t^{+}_{i,j,m}(b) = 1$ and $f(b) = 0$ for some 2-quasi-Horn implicants t_i, t_j and t_m of f such that condition (11.1) holds. Because $t^{+}_{i,j,m}(b) = 1$ we have that

$$\mathcal{O}(b) \supseteq P(t_i) \cup P(t_j) \cup P(t_m).$$

Furthermore,

$$N(t_q) \cap \mathcal{O}(b) \neq \emptyset, \ q = i, j, m, \tag{11.2}$$

because otherwise we have $t_q(b) = 1$, and hence $f(b) = 1$, which is a contradiction. Now let us take four vectors b, $b^{(i)}$, $b^{(j)}$ and $b^{(m)}$, where $b^{(q)}$ denotes the vector such that

$$\mathcal{O}(b^{(q)}) = \mathcal{O}(b) \setminus N(t_q) \supseteq P(t_q). \tag{11.3}$$

Because of (11.1) we have

$$b = b^{(i)} \vee b^{(j)} \vee b^{(m)}. \tag{11.4}$$

Because

$$t_i(b^{(i)}) = t_j(b^{(j)}) = t_m(b^{(m)}) = 1,$$

we have

$$f(b^{(i)}) = f(b^{(j)}) = f(b^{(m)}) = 1,$$

and $f(b) = 0$ holds by assumption. From (11.1)-(11.4) we conclude that b is a 2-ones-disjunction of $b^{(i)}$, $b^{(j)}$ and $b^{(m)}$, and, consequently, $T(f)$ is not closed under 2-ones-disjunctions, that is, f is not bidual 2-quasi-Horn. □

Lemma 6 *Let φ be a 2-quasi-Horn DNF. If φ represents a bidual 2-quasi-Horn function then*

$$t^{+}_{i,j,m} \leq \varphi \text{ and } \hat{t}^{\pm}_{i,j,m} \leq \varphi$$

holds for every three 2-quasi-Horn terms t_i, t_j and t_m in φ such that

$$|N(t_q)| = |\tilde{N}(t_q)| \geq 1, \;\; q = i, j, m.$$

Proof. Special case of Lemma 5. □

We extend the above results in the next two Lemmas.

Lemma 7 *Let f be a k-quasi-Horn function. If f is a bidual k-quasi-Horn function, then for every $k+1$ k-quasi-Horn implicants t_{i_1}, t_{i_2}, $\ldots$, $t_{i_{k+1}}$ of f such that*

$$|N(t_s)| = |\tilde{N}(t_s)| \geq 1, \;\; s = 1, 2, \ldots, k+1,$$

it holds that

$$t^{+}_{i_1,i_2,\ldots,i_{k+1}} \leq f \text{ and } \hat{t}^{\pm}_{i_1,i_2,\ldots,i_{k+1}} \leq f.$$

Lemma 8 *Let φ be a k-quasi-Horn DNF. If φ represents a bidual k-quasi-Horn function, then*

$$t^{+}_{i_1,i_2,\ldots,i_{k+1}} \leq \varphi \text{ and } \hat{t}^{\pm}_{i_1,i_2,\ldots,i_{k+1}} \leq \varphi$$

holds for every $k+1$ k-quasi-Horn terms t_{i_1}, t_{i_2}, $\ldots$, $t_{i_{k+1}}$ in φ such that

$$|N(t_s)| = |\tilde{N}(t_s)| \geq 1, \;\; s = 1, 2, \ldots, k+1.$$

11.4.2 Sufficient conditions

Lemma 9 *Let f be a 2-quasi-Horn function. If for every three 2-quasi-Horn implicants t_i, t_j and t_m of f such that*

$$|\tilde{N}(t_q)| \geq 1, q = i, j, m, \tag{11.5}$$

it holds that

$$t^{+}_{i,j,m} \leq f \text{ and } \hat{t}^{\pm}_{i,j,m} \leq f, \tag{11.6}$$

then f is a bidual 2-quasi-Horn function.

Proof. Assume that f is not bidual, or in other words its dual f^d is not 2-quasi-Horn. Because f^d is not 2-quasi-Horn, $T(f)$ is not closed under 2-ones-disjunctions. Hence, there exist four vectors, u, $v^{(i)}$, $v^{(j)}$ and $v^{(m)}$, such that

$$u = v^{(i)} \vee v^{(j)} \vee v^{(m)}, \tag{11.7}$$

u is a 2-ones-disjunction of $v^{(i)}$, $v^{(j)}$ and $v^{(m)}$, $f(u) = 0$ and $f(v^{(q)}) = 1$, for $q = i, j, m$. The fact that $f(v^{(q)}) = 1$ implies that there is a 2-quasi-Horn implicant

$$t_q = \Big(\bigwedge_{l \in P(t_q)} x_l\Big) \wedge \Big(\bigwedge_{l \in N(t_q)} \overline{x}_l\Big)$$

of f such that

$$t_q(v^{(q)}) = 1, \ q = i, j, m. \tag{11.8}$$

We will show that condition (11.5) holds, $t^{+}_{i,j,m} = 1$ and $\hat{t}^{\pm}_{i,j,m} = 1$. Because $f(u) = 0$ this will contradict condition (11.6).

(a) $N(t_q) \neq \emptyset$ for $q = i, j, m$. Otherwise, (11.7)-(11.8) imply $t_q(u) = 1$, that is, $f(u) = 1$, which is a contradiction.

(b) $N(t_i) \cap \mathcal{O}(v^{(j)} \vee v^{(m)}) \neq \emptyset$. Assume that $N(t_i) \subseteq \mathcal{Z}(u)$ holds. Then (11.7)-(11.8) imply that $t_i(u) = 1$, that is, $f(u) = 1$, which is a contradiction. The cases $N(t_j) \subseteq \mathcal{Z}(u)$ and $N(t_m) \subseteq \mathcal{Z}(u)$ are analogous.

(c) $\tilde{N}(t_i) \neq \emptyset$. Assume $\tilde{N}(t_i) = \emptyset$, or in other words $N(t_i) \subseteq N(t_j) \cup N(t_m)$. If there exists

$$p \in \bigcap_{q \in \{i,j,m\}} N(t_q),$$

then $u_p \neq 1$, because otherwise there exists $q \in \{i, j, m\}$ such that $t_q(v^{(q)}) = 0$. From (b) assume that $h \in N(t_i) \cap \mathcal{O}(v^{(j)} \vee v^{(m)})$, $h \notin N(t_m)$ and $g \in N(t_m) \cap \mathcal{O}(v^{(i)} \vee v^{(j)})$. Then $u_h = u_g = 1$, and for every $q = i, j, m$ we have that $v_h^{(q)} v_g^{(q)} = 0$, because from (11.8)

$$v_h^{(i)} = v_h^{(j)} = v_g^{(m)} = 0.$$

Hence, u is not a 2-ones-disjunction of $v^{(i)}$, $v^{(j)}$ and $v^{(m)}$, contradiction. The cases $\tilde{N}(t_j) = \emptyset$ and $\tilde{N}(t_m) = \emptyset$ are analogous.

From (c) we conclude that condition (11.5) holds and $t^{+}_{i,j,m}(u) = 1$, contradiction to condition (11.6) since $f(u) = 0$. Now let us show that $\hat{t}^{\pm}_{i,j,m}(u) = 1$.

(d) $\tilde{N}(t_i) \cap \mathcal{O}(v^{(j)} \vee v^{(m)}) \neq \emptyset$. Assume that $\tilde{N}(t_i) \subseteq \mathcal{Z}(u)$. Then using the same arguments as in (c) we have that 2-Ones Condition does not hold, contradiction. The cases $\tilde{N}(t_j) \subseteq \mathcal{Z}(u)$ and $\tilde{N}(t_m) \subseteq \mathcal{Z}(u)$ are analogous. Hence we conclude that $\hat{t}^{\pm}_{i,j,m}(u) = 1$. $\square$

Lemma 10 *Let φ be a 2-quasi-Horn DNF. Then φ represents a bidual 2-quasi-Horn function f_φ if*

$$t^{+}_{i,j,m} \leq \varphi \text{ and } \hat{t}^{\pm}_{i,j,m} \leq \varphi \tag{11.9}$$

holds for every three 2-quasi-Horn terms t_i, t_j and t_m in φ such that

$$|\tilde{N}(t_q)| \geq 1, \; q = i, j, m.$$

Proof. Let us assume that (11.9) holds, but f_φ is not bidual. Hence, f^d_φ has a prime implicant r such that $|N(r)| \geq 3$, and each term in φ has a common literal with r (it is easy to see by dualizing φ). Furthermore, because r is prime, for each literal ℓ in r there is a term in φ having only literal ℓ in common with r. Otherwise a proper subterm of r would be an implicant of f^d_φ, which is a contradiction, because r is prime. Let t_1, t_2 and t_3 be respectively such terms in φ for different negative literals in r, that is,

$$P(t_i) \cap P(r) = \emptyset. \tag{11.10}$$

By the assumption $t^{+}_{1,2,3}$ is an implicant of f_φ, hence it has a common literal with r, that is

$$P(t^{+}_{1,2,3}) \cap P(r) \neq \emptyset,$$

which is a contradiction to (11.10). The case $\hat{t}^{\pm}_{i,j,m} \leq \varphi$ is analogous. $\square$

The above results can be generalized as follows.

Lemma 11 *Let f be a k-quasi-Horn function. If for every $k+1$ k-quasi-Horn implicants $t_{i_1}, t_{i_2}, \ldots, t_{i_{k+1}}$ of f such that*

$$|\tilde{N}(t_s)| \geq 1, s = 1, 2, \ldots, k+1,$$

it holds that

$$t^{+}_{i_1,i_2,\ldots,i_{k+1}} \leq f \text{ and } \hat{t}^{\pm}_{i_1,i_2,\ldots,i_{k+1}} \leq f,$$

then f is a bidual k-quasi-Horn function.

Lemma 12 *Let φ be a k-quasi-Horn DNF. Then φ represents a bidual k-quasi-Horn function f_φ if*

$$t^{+}_{i_1,i_2,\ldots,i_{k+1}} \leq \varphi \text{ and } \hat{t}^{\pm}_{i_1,i_2,\ldots,i_{k+1}} \leq \varphi$$

holds for every $k+1$ k-quasi-Horn terms $t_{i_1}, t_{i_2}, \ldots, t_{i_{k+1}}$ in φ such that

$$|\tilde{N}(t_s)| \geq 1, \; s = 1, 2, \ldots, k+1.$$

References

[1] B. Alidaee. Optimal ordering policy of a sequential model, *Journal of Optimization Theory and Applications*, vol. 83, pp.199-205, 1994.

[2] S. Anand, D. Madigan, R. Mammone, S. Pathak, and F. Roberts. Experimental analysis of sequential decision making algorithms for port of entry inspection procedures, in S. Mehrotra, D. Zeng, H. Chen, B. Thuraisingham, and F-X Wang (eds.), Intelligence and Security Informatics, Proceedings of ISI-2006, Lecture Notes in Computer Science 3975, Springer-Verlag, New York, 2006, 319-330.

[3] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9, 1992: 147-164.

[4] M. Anthony. The sample complexity and computational complexity of Boolean function learning. CDAM Research Report CDAM-LSE-2002-13, December 2002.

[5] M. Anthony and N. L. Biggs. Computational Learning Theory: An Introduction. Cambridge Tracts in Theoretical Computer Science, 30, 1992. Cambridge University Press, Cambridge, UK.

[6] A. Asuncion and D. J. Newman, (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[7] Y. Ben-Dov. Optimal testing procedures for special structures of coherent systems, *Management Science*, vol. 27, no.12, pp. 1410-1420, 1981.

[8] D. Blackwell and M. A. Girshik, Theory of games and statistical decisions, Wiley and Sons, 1954.

[9] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4), 1989: 929-965.

[10] E. Boros. Maximum renamable Horn sub-CNFs. *Discrete Applied Mathematics*, 96-97(1-3), 1999: 29-40.

[11] E. Boros, Y. Crama, and P. L. Hammer. Polynomial-time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, 1, 1990: 21-32.

[12] E. Boros, P. L. Hammer, T. Ibaraki, and A. Kogan. Logical Analysis of Numerical Data, Mathematical Programming, 79, 163-190, 1997.

[13] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An Implementation of the Logical Analysis of Data, *IEEE Transactions on Knowledge and Data Engineering*, 12, No.2, 292-306, 2000.

[14] E. Boros and T. Unluyurt. Diagnosing double regular systems, In: Artificial Intelligence and Mathematics – AIM'98, Electronic Proceedings of the Fifth International Symposium on Artificial Intelligence and Mathematics, January , 1998, E. Boros and R. Greiner, eds.

[15] E. Boros and T. Unluyurt. Diagnosing double regular systems, *Annals of Mathematics and Artificial Intelligence*, 26, (1-4):171-191,1999.

[16] E. Boros and T. Unluyurt. Sequential testing of series-parallel systems of small depth, In: Computing Tools for Modeling, Optimization and Simulation, 39-74, 2000, Laguna and Velarde eds., Kluwer Academic Publishers, Boston.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

[18] S. D. Buluswer and B. A. Draper. Non-parametric classification of pixels under varying illumination. SPIE: The Int. Society for Optical Eng., 2353:529-536, November 1994.

[19] R. Butterworth. Some reliability fault testing models, *Operations Research*, vol. 20, pp.335-343, 1972.

[20] C. L. Chang and J. R. Slagle. An admissible and optimal algorithm for searching and-or graphs, *Artificial Intelligence* 2 117-128, 1971.

[21] M. Chang, W. Shi, and W. K. Fuchs. Optimal diagnosis procedures for k-out-of-n structures, *IEEE Trans. Comput.* 39, No.4, 1990. 559564.

[22] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search, *Journal of the American Statistical Association*, 93, 935-960, 1998.

[23] H. A. Chipman, E. I. George, and R. E. McCulloch. Extracting representative tree models from a forest, working paper 98-07, 1998, Department of Statistics and Actuarial Science, University of Waterloo.

[24] V. Chvatal. Linear Programming. Freeman, 1983.

[25] M. Couceiro and S. Foldes. On affine constraints satisfied by Boolean functions. *RUTCOR Research Report*, 3, 2003. Rutgers University, New Brunswick, NJ, USA. http://rutcor.rutgers.edu/∼rrr/2003.html

[26] M. Couceiro and S. Foldes. Definability of Boolean function classes by linear equations over GF(2). *Discrete Applied Mathematics*, 142, 2004: 29-34.

[27] L. A. Cox Jr., Y. Qiu, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values, *Annals of Operations Research* 21. 1-21, 1989.

[28] S. K. Das and S. Bhambri. A decision tree approach for selecting between demand based, reorder and JIT/kanban methods for material procurement. Production Planning and Control, 5(4):342, 1994.

[29] M. Dubrovsky. The Binary Decision Tree: The Growing Algorithm and Application to Thunderstorm Forecasting. Studia geophysicae et geophysica et geodaetica, 39, No.1, 84-100, 1995.

[30] S. O. Duffuaa and A. Raouf. An optimal sequence in multicharacteristics inspection, *J. Optim. Theory Appl.* 67(1) 79-87, 1990.

[31] T. Eiter, T. Ibaraki and K. Makino. Bidual Horn Functions and Extensions. *Discrete Applied Mathematics*, 96-97, 1999: 55-88.

[32] O. Ekin, S. Foldes, P. L. Hammer, and L. Hellerstein. Equational characterizations of Boolean function classes. *Discrete Mathematics*, 211, 2000, 27-51.

[33] A. Ercil. Classification trees prove useful in nondestructive testing of spotweld quality. J. Welding, 72(9):59, September 1993. Issue Title: Special emphasis: Rebuilding America's roads, railways and bridges.

[34] B. Evans and D. Fisher. Overcoming process delays with decision tree induction. IEEE Expert, pages 60-66, February 1994.

[35] J. A. Falconer, B. J. Naughton, D. D. Dunlop, E. J. Roth, and D. C. Strasser. Predicting stroke inpatient rehabilitation outcome using a classification tree approach. *Archives of Physical Medicine and Rehabilitation*, 75(6):619, June 1994.

[36] A. Famili. Use of decision tree induction for process optimization and knowledge refinement of an industrial process. Artificial Intelligence for Eng. Design, Analysis and Manufacturing (AI EDAM), 8(1):63-75, Winter 1994.

[37] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem, Part I. Operations Research 9 (1961), pp. 849-859.

[38] P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem, Part II. Operations Research 11 (1963), pp. 863-888.

[39] R. Greiner. Finding optimal derivation strategies in redundant knowledge bases, Artif. Intell. 50, 95-115, 1990.

[40] P. L. Hammer and I. I. Lozina. Boolean Separators and Approximate Boolean Classifiers. *RUTCOR Research Report*, 14, 2006. Rutgers University, New Brunswick, NJ, USA. http://rutcor.rutgers.edu/∼rrr/2006.html

[41] K. Haraguchi, T. Ibaraki, and E. Boros. Classifiers based on iterative compositions of features, Proc. 1st Intl. Conf. Knowledge Engineering and Decision Support, 143-150, Porto, Portugal, Aug. 2004

[42] A. Horn. On sentences which are true of direct unions of algebras. Journal of Symbolic Logic, 16(1), 1951: 14-21.

[43] T. Ibaraki, A. Kogan, and K. Makino. On Functional Dependencies in q-Horn Theories. *Artificial Intelligence*, 131(1-2), 2001: 171-187.

[44] W. B. Joyce. Organizations of unsuccessful R&D projects, IEEE Transactions on Engineering Management, vol.18, no.2, pp 57-65, 1971.

[45] J. Judmaier, P. Meyersbach, G.Weiss, H.Wachter, and G. Reibnegger. The role of Neopterin in assessing disease activity in Crohn's disease: Classification and regression trees. The American J. of Gastroenterology, 88(5):706, May 1993.

[46] J. B. Kadane. Quiz show problems, J. Math. Anal. Appl. 27, 609–623, 1969.

[47] L. G. Khachiyan. A polynomial algorithm in linear programming, (in Russian). Doklady Akedamii Nauk SSSR, 244 (1979), pp. 10931096.

[48] P. Kokol, M. Mernik, J. Zavrsnik, and K. Kancler. Decision trees based on automatic learning and their use in cardiology. Journal of Medical Systems, 18(4):201, 1994.

[49] R. Kowalski. Search strategies for theorem proving, in: Machine Intelligence, Vol. 5, eds. B. Meltzer and D. Mitchie, (Edinburgh University Press, Edinburgh, 1969) pp. 181-201.

[50] R. Kowalski. And-or graphs, theorem proving graphs and bi-directional search, in: Machine Intelligence, Vol. 7, eds. B. Meltzer and D. Mitchie (Edinburgh University Press, Edinburgh, 1972) pp. 167- 194.

[51] H. Kushner and A. Pakut. A Simulation Study of Decentralized Detection Problem, IEEE Trans. On Automatic Control, vol. AC-27, No. 5, pp.1116-1119, 1982.

[52] W. Lehnert, S. Soderland, D. Aronow, F. Feng, and A. Shmueli. Inductive text classification for medical applications. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):49-80, January-March 1995.

[53] W. J. Long, J. L. Grifith, H. P. Selker, and R. B. D'Agostino. A comparison of logistic regression to decision tree induction in a medical domain. Comp. and Biomedical Research, 26(1):74-97, February 1993.

[54] J. M. Loy and R. G. Ross, (2002), Global Trade: Americas Achilles Heel, Defense Horizon, Center for Technology and National Security University.

[55] R. D. Luce and H. Raiffa. Games and decisions: introduction and critical survey. New York: Wiley, 1957. 509 p.

[56] D. Madigan, S. Mittal, and F. Roberts. Sequential decision making algorithms for port of entry inspection: Overcoming computational challenges, in G. Muresan, T. Altiok, B. Melamed, and D. Zeng (eds.), Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI-2007), IEEE Press, Piscataway, NJ, May 2007, 1-7.

[57] D. P. McKenzie, P. D. McGorry, C. S.Wallace, L. H. Low, D. L. Copolov, and B. S. Singh. Constructing a minimal diagnostic decision tree. *Methods of Information in Medicine*, 32(2):161-166, April 1993.

[58] J. C. C. McKinsey. The decision problem for some classes of sentences without quantifiers. Journal of Symbolic Logic, 8(3), 1943: 61-76.

[59] R. Miglio and G. Soffritti. The comparison between classification trees through proximity measures, *Computational Statistics and Data Analysis*, 45, 577-593, 2004.

[60] T. Mitchell. Decision Tree Learning, in T. Mitchell, Machine Learning, The McGraw-Hill Companies, Inc., 1997, pp. 52-78.

[61] L. G. Mitten. An analytic solution to the least cost testing sequence problem, *The Journal of Industrial Engineering*, Jan-Feb 1960, pp.17.

[62] J. Nash. Equilibrium points in n-person games, *Proceedings of the National Academy of the USA*, 36(1):48-49, 1950.

[63] J. Nash. Non-Cooperative Games, *The Annals of Mathematics* 54(2):286-295, 1951.

[64] K. S. Natarajan. Optimizing depth-first search of AND-OR trees, Technical report, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1986.

[65] N. J. Nilsson. Problem-Solving Methods in Artificial Intelligence (McGraw-Hill, New York, 1971).

[66] C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V-C Dimension. *Journal of Computer and System Sciences*, 53(2), 1996: 161-170.

[67] P. Perner, T. B. Belikova, and N. I. Yashunskaya. Knowledge acquisition by symbolic decision tree induction for interpretation of digital images in radiology. Lecture Notes in Computer Science, 1121:208, 1996.

[68] I. Pohl. Bi-directional search, in: Machine Intelligence, Vol. 6, eds. B. Meltzer and D. Mitchie (Edinburgh University Press, Edinburgh, 1971) pp. 127-140.

[69] F. Roberts. Decision support algorithms for port-of-entry inspection, in Working Together: Research and Development Partnerships in Homeland Security, Proceedings of DHS/IEEE Conference, Boston, 2005.

[70] S. Salzberg, R. Chandar, H. Ford, S. Murthy, and R. White. Decision trees for automated identification of cosmic-ray hits in Hubble Space Telescope images. *Publications of the Astronomical Society of the Pacific*, 107:1-10, March 1995.

[71] P. D. Stroud and K. J. Saeger. Enumeration of increasing Boolean expressions and alternative digraph implementations for diagnostic applications. Proceedings of Computer, Communication and Control Technologies (volume IV, H. Chu, J. Ferrer, T. Nguyen and Y. Yu, eds.) International Institute of Informatics and Systematics, Orlando, FL, 2003, pp. 328-333.

[72] T. Unluyurt. Sequential testing of complex systems: A review, *Discrete Applied Mathematics*, 142, (1-3):189-205, 2004.

[73] T. Unluyurt. Testing systems of identical components, Journal of Combinatorial Optimization, 10, (3):261-282, 2005.

[74] L. G. Valiant. A theory of learnable. Communications of the ACM, 27(11), 1984: 1134-1142.

[75] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of the relative frequencies of events to their probabilities. Theory of Probability and its Applications, 16(2), 1971: 264-280.

[76] J. von Neumann and O. Morgenstern. Theory of Games and Economic Behavior, Princeton University Press, Princeton NJ, 1947.

[77] A. Wald. Sequential Analysis, John Wiley and Sons, 1947.

[78] A. Wald. Statistical Decision Functions, John Wiley and Sons, 1950.

[79] P. Winston. Learning by Building Identification Trees, in P. Winston, Artificial Intelligence, Addison-Wesley Publishing Company, 1992, pp. 423-442.

[80] I. H. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[81] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora. Decision tree regression for soft classification of remote sensing data, Remote Sens. Environ., vol. 97, pp. 322-336, Aug. 2005.

[82] H. Zhang, C. Schroepfer, and E. A. Elsayed. Sensor Thresholds in Port-of-Entry Inspection Systems, Proceedings of the 12th ISSAT International Conference on Reliability and Quality in Design, Chicago, Illinois, USA, August 3-5, 2006, pp. 172-176.

Vita

Liliya Fedzhora

2001-2008	Ph.D. in Operations Research, Rutgers, The State University of New Jersey
1998-2000	M.A. in Economics, Kyiv-Mohyla Academy, Ukraine
1993-1998	B.S. in Mathematics, Lviv National University, Ukraine

2007-pres	Business Analyst, Virginia Department of Transportation
2005-2007	Research Assistant, Department of Operations Research, Rutgers University