# Function Evaluation Via Linear Programming in the Priced Information Model

Ferdinando Cicalese*,[1] and Eduardo Sany Laber[2]

[1] AG Genominformatik, Technical Faculty, Bielefeld University, Germany
[2] Departamento de Informática, PUC – Rio de Janeiro, Brazil

**Abstract.** We determine the complexity of evaluating monotone Boolean functions in a variant of the decision tree model introduced in [Charikar *et al.* 2002]. In this model, reading different variables can incur different costs, and competitive analysis is employed to evaluate the performance of the algorithms. It is known that for a monotone Boolean function $f$, the size of the largest certificate, aka $PROOF(f)$, is a lower bound for $\gamma(f)$, the best possible competitiveness achievable by an algorithm on $f$. This bound has been proved to be achievable for some subclasses of the monotone Boolean functions, e.g., read once formulae, threshold trees. However, determining $\gamma(f)$ for an arbitrary monotone Boolean function has so far remained a major open question, with the best known upper bound being essentially a factor of 2 away from the above lower bound.

We close the gap and prove that *for any monotone Boolean function* $f$, $\gamma(f) = PROOF(f)$. In fact, we prove that $\gamma(f) \leq PROOF(f)$ holds for a class much larger than the set of monotone Boolean functions. This class also contains all Boolean functions.

## 1 Introduction

The decision tree is perhaps the simplest model of computation for studying the complexity of Boolean functions. In the classical variant of this model, a Boolean function is to be evaluated by querying about the initially unknown input. Each query asks the value of a single variable and it is charged a unitary cost. No other computational cost is taken into account. Then, the decision tree complexity $DC(f)$ of a function $f$ is defined as the number of queries that an optimal algorithm for evaluating $f$ needs to make on the most costly input. This is, in fact, the depth of the tree induced by the optimal algorithm. Clearly, $DC(f) \leq n$ for all functions on $n$ variables. Conversely, a large part of the Boolean functions are known to be *evasive*, i.e., to satisfy $DC(f) = n$ [18]. In a certain way, this shows that the model is "too simple" to characterize the complexity of large classes of functions in an algorithmically meaningful way: in fact, for each evasive function, any algorithm has the same "optimal" behavior. There are basically two ways out of here: to restrict the classes of functions

* Supported by the Sofja Kovalevskaja Award 2004 of the Alexander von Humboldt Foundation and the Bundesministerium für Bildung und Forschung.

L. Aceto et al. (Eds.): ICALP 2008, Part I, LNCS 5125, pp. 173–185, 2008.

one focuses on  or to slightly change the model in a way that allows more discriminative power. Both ways have been considered in the literature. Several subclasses of Boolean functions have been studied in the basic and in other variants of the decision tree model by, e.g., considering statistical knowledge on the input together with average case analysis, or by allowing randomization (see, e.g., [21,20,19] and references quoted therein). However, only very few non-trivial classes of functions have been completely characterized so far [19,12]. Conversely, for many classes studied, the gap between the best known bounds is still significant (see, e.g., [13,19], also, [2,10] include comprehensive surveys of the known results).

In this paper, we follow the more recent approach proposed by Charikar *et al.* in [3]. We assume that different variables can incur different reading costs and we employ competitive analysis to measure the performance of the evaluation algorithms. In particular, we study the following model.

**Problem Statement.** A function $f$ over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$ has to be evaluated for a fixed but unknown *assignment* $\sigma$, i.e., a choice of the values for the variables of $V$. Each variable $x_i$ has an associated non-negative cost $c(x_i)$ which is the cost incurred to probe $x_i$, i.e., to read its value $x_i(\sigma)$. For each $i = 1, \ldots, n$, the cost $c(x_i)$ is fixed and known beforehand. The goal is to *adaptively* identify and probe a minimum cost set of variables $U \subseteq V$ whose values uniquely determine the value of $f$ for the given assignment, regardless of the values of the variables not probed. The cost $c(U)$ of $U$ is the sum of the costs of the variables it contains, i.e., $c(U) = \sum_{x \in U} c(x)$. We use $f(\sigma)$ to denote the value of $f$ w.r.t. $\sigma$, i.e., $f(\sigma) = f(x_1(\sigma), \ldots, x_n(\sigma))$.

A set of variables $U \subseteq V$ is a *proof* for $f$ with respect to a given assignment $\sigma$ for the variables of $V$ if the value $f(\sigma)$ is determined by the values that $\sigma$ assigns to the variables of $U$ regardless of the values assigned to the other variables.

An evaluation algorithm $\mathcal{A}$ for $f$ is a rule to adaptively read the variables in $V$ until the set of variables read so far is a proof for the value of $f$. The cost of algorithm $\mathcal{A}$ for an assignment $\sigma$ is the total cost incurred by $\mathcal{A}$ to evaluate $f$ under the assignment $\sigma$. Given a cost function $c(\cdot)$, we let $c^f_{\mathcal{A}}(\sigma)$ denote the cost of the algorithm $\mathcal{A}$ for an assignment $\sigma$ and $c^f(\sigma)$ the cost of the cheapest proof for $f$ under the assignment $\sigma$. We say that $\mathcal{A}$ is $\rho$-competitive if $c^f_{\mathcal{A}}(\sigma) \leq \rho c^f(\sigma)$, for every possible assignment $\sigma$. We use $\gamma^{\mathcal{A}}_c(f)$ to denote the competitive ratio of $\mathcal{A}$, that is, the minimum $\rho$ for which $\mathcal{A}$ is $\rho$-competitive. The best possible competitive ratio for any deterministic algorithm, then, is $\gamma^f_c = \min_{\mathcal{A}} \gamma^{\mathcal{A}}_c(f)$, where the minimum is computed over all possible deterministic algorithms $\mathcal{A}$.

With the aim of evaluating the dependence of the competitive ratio on the structure of $f$, one defines the extremal competitive ratio $\gamma^{\mathcal{A}}(f)$ of an algorithm $\mathcal{A}$ as $\gamma^{\mathcal{A}}(f) = \max_c \gamma^{\mathcal{A}}_c(f)$. The best possible extremal competitive ratio for any deterministic algorithm, then, is $\gamma(f) = \min_{\mathcal{A}} \gamma^{\mathcal{A}}(f)$. This last measure is meant to capture the structural complexity of $f$ independent of a particular cost assignment and algorithm. For instance, consider the Boolean function

$$f = (x_1 \text{ AND } x_2) \text{ OR } (x_2 \text{ AND } x_3) \text{ OR } (x_3 \text{ AND } x_4) \tag{1}$$

together with the costs $c(x_1) = 3, c(x_2) = 5, c(x_3) = 4$ and $c(x_4) = 1$. For the assignment $\sigma_R = (1, 0, 1, 1)$, we have $f(\sigma_R) = 1$ and $U = \{x_3, x_4\}$ as the only proof of minimum cost. Therefore, $c^f(\sigma_R) = 4 + 1$. On the other hand, for the assignment $\sigma_S = (1, 0, 0, 0)$, we have $f(\sigma_S) = 0$ and the cheapest proof is $\{x_2, x_4\}$. Thus, $c^f(\sigma_S) = 5 + 1$. Let now $\mathcal{A}$ be an algorithm that reads first $x_1$, then $x_2$, and so on, just skipping a variable $x_i$ if, due to the values read so far, the value of $x_i$ cannot affect the value of $f$. Thus, it is not hard to verify that $c^f_{\mathcal{A}}(\sigma_R) = 13$, since $\mathcal{A}$ reads the variables $x_1, x_2, x_3, x_4$. Furthermore, $c^f_{\mathcal{A}}(\sigma_S) = 12$, since in this case, $\mathcal{A}$ reads $x_1, x_2$ and $x_3$.

Beside its important theoretical aspects, function evaluation problems arise in several domains of computer science. In automatic diagnosis problems, a given system has to be checked by performing some tests whose costs can be different. This typically means evaluating some (Boolean) function and looking for the cheapest testing procedure (see, e.g., [1] and references therein). Applications are found in a variety of fields, e.g., telecommunications [16], manufacturing [7], computer networks [8], satisficing search problems [10], computer aided medical systems [17]. Also, the function evaluation problem arises in query optimization, a major issue in databases [14].

**Related Work.** The extremal competitive ratio was proposed by Charikar *et al.* in [3]. In the same paper, they present bounds on the extremal competitive ratio for monotone Boolean functions and some non-Boolean functions as the function "minimum of a list" and the function "searching a sorted list". Most of these bounds were improved in subsequent papers [11,4,5]. For some results regarding the measure $\gamma^f_c$ see also [3,6].

For monotone Boolean functions, Charikar *et al.* proved that for every function $f$ representable by an AND/OR tree it holds $\gamma(f) = PROOF(f)$, where $PROOF(f)$ is the size of the largest minimal proof for $f$. In addition, they mention that the inequality $\gamma(f) \leq 2 \times PROOF(f)$ holds for every monotone Boolean function $f$. This follows from the existence of an exponential time algorithm that achieves this bound. In [4], the authors observed that $\gamma(f) \geq PROOF(f)$ holds for any monotone Boolean function and a polytime algorithm[1] was presented showing that $\gamma(f)$ is slightly smaller than $2 \times PROOF(f)$. In [5], $\gamma(f) = PROOF(f)$ is proved for the classes of threshold tree functions (a class that includes AND/OR trees) and monotone Boolean functions for which no variable appears in more than 3 minterms. Determining whether $\gamma(f) = PROOF(f)$ holds for every monotone Boolean function $f$ has so far remained a major open question.

**Our Results.** We prove that $\gamma(f) = PROOF(f)$ for all monotone Boolean functions. Actually, our main result is much more general: we prove $\gamma(f) \leq PROOF(f)$ for any non constant function $f : S_1 \times \cdots \times S_n \to \mathbb{R}$ whose domain is given by the Cartesian product of the domains of the individual variables. The class of such functions, which we denote by $\mathcal{F}^{\times}$, contains the whole class

[1] The time bound assumes that, given a $\sigma$, the algorithm can obtain $f(\sigma)$ in polytime.

of (total) Boolean functions. This seems a rather interesting result, considering how few assumptions are made on the structure of the functions in question.

We achieve the above bound by employing an LP-based approach ($\mathcal{LPA}$) to the design of algorithms for the function evaluation problem. The $\mathcal{LPA}$ is based on the solution of a linear program defined on the set of the minimal proofs of the function under consideration. The same linear program had been used for a variant of the problem in which the costs are unknown beforehand [6]. The optimal solution of this LP is used to capture the impacts of the variables of $f$ in the process of evaluating $f$. The competitiveness of an algorithm obtainable by this approach is directly provided by the cost of the optimal solution for the linear program. Through an elegant geometric argument we are able to prove that this cost is not larger than $PROOF(f)$ for any $f \in \mathcal{F}^{\times}$.

Finding the optimal solution for our linear program can be a hard problem due to its potentially huge number of constraints. Therefore, when, in practical applications, also the cost of the computation, other than just the cost of probing the variables, is to be taken into account, suboptimal feasible solutions that are easy to compute might be preferable. We point out that for AND/OR trees and threshold trees, subclasses of monotone Boolean functions which have been studied in the literature, polynomial time algorithms with competitive ratio $\gamma(f)$ can be obtained through our linear programming based approach. In addition, when monotone Boolean functions are represented by their list of minterms (minimal proofs that guarantee that the function evaluates to 1) we can use our approach to isolate, in polynomial time, up to $n$ strategies such that at least one of them has competitive ratio $\gamma(f)$. These results are presented in Section 3.1

Though we focus mainly on monotone Boolean functions, our linear programming approach has much broader applicability since it does not rely on any structure of the (class of) function(s) under consideration. We discuss how to employ it for other basic problems studied under the priced information framework, such as finding the minimum, sorting and searching [3,11,15]. We show that $\mathcal{LPA}$ allows us to devise algorithms with the best known competitive ratio for finding the minimum and for sorting.

We remark that the $\mathcal{LPA}$ extends the *General Approach* introduced in [4]. In fact, this can be considered a restriction of the $\mathcal{LPA}$, in which the impact of a variable takes values in $\{0, 1/p\}$ for some (implementation dependent) integer $0 < p \leq n$. Due to this constraint, the *General Approach* suffers from some intrinsic limitations [4] that are not inherited by the new methodology we present here. As an example, our main result, $\gamma(f) = PROOF(f)$ for monotone Boolean function $f$, cannot be proved with the *General Approach*.

## 2 Preliminaries

Let $f$ be a function over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$. In this paper we shall always assume that $V$ does not contain redundant variables, i.e., the value of $f$ depends upon the value of all the variables in $V$.

Let $Y \subseteq V$ and let $\sigma$ be an assignment for the variables of $Y$. We use $f_{Y,\sigma}$ to denote the function over $V \setminus Y$ obtained from $f$ by fixing the values of the variables in $Y$ as given by $\sigma$. Consider, e.g., the function $f$ in (1). Let $Y = \{x_2, x_4\}$ and $\sigma = (x_2 = 1, x_4 = 1)$. Then, we have $f_{Y,\sigma} = x_1$ OR $x_3$. In general, throughout this paper, $Y$ will denote the set of variables read so far by the algorithm that evaluates $f$, and $\sigma$ will be the assignment given by the values obtained when the variables of $Y$ are read. We will usually write $f_Y$ instead of $f_{Y,\sigma}$ whenever the assignment $\sigma$ is clear from the context.

Given an assignment $\sigma$ for the variables of $V$ and a set of variables $Y \subseteq V$, we use $\sigma_Y$ to denote the assignment $\sigma$ restricted to the variables of $Y$, i.e., $\sigma_Y$ is the assignment for the variables of $Y$ satisfying $x(\sigma_Y) = x(\sigma)$ for every $x \in Y$.

Let $v$ be a value in the range of $f$. A *$v$-witness* for $f$ is a set of variables $C \subseteq V$ such that there is an assignment $\sigma$, with $f(\sigma) = v$, for which $C$ is a proof for $f$ with respect to $\sigma$.

We say that $C \subseteq V$ is a *$v$-certificate* for $f$ if $C$ is a $v$-witness for $f$ and it is minimal, i.e., for any $x \in C$, the set $C \setminus \{x\}$ is not a $v$-witness for $f$. More generally, we say that a set $C \in V$ is a *certificate* for $f$ if there exists a $v$ such that $C$ is a $v$-certificate for $f$. Note that for all assignments $\sigma$, every proof for $f$ with respect to $\sigma$ contains a certificate. Moreover, we have that $PROOF(f)$ is the size of the largest certificate for $f$.

Recall that $\mathcal{F}^\times$ denotes the class of functions whose domain is given by the Cartesian product of the domains of the single variables. In other words, the class $\mathcal{F}^\times$ is the set of functions whose variables' values can be chosen independently of each other. For instance, a function $g : \{(0,1),(1,0),(1,1)\} \to \{0,1\}$, is not in $\mathcal{F}^\times$. In fact, for $g$, both variables have individually domain $\{0,1\}$ but the function is not defined in $(0,0)$. In the last section, we will cope with some special functions not in $\mathcal{F}^\times$.

**Proposition 1.** *Let $f$ be a non-constant function in $\mathcal{F}^\times$. Let $u$ and $v$ be distinct values in the range of $f$ and $C, D \subseteq V$ be a $u$-witness and $v$-witness for $f$ respectively. Then, $C \cap D \neq \emptyset$.*

The above proposition holds because otherwise we could construct an assignment $\sigma$ for $f$ such that $f(\sigma) = v$ and $f(\sigma) = u$.

**Proposition 2.** *Let $V$ be the set of variables of a function $f$. Then, for every $Y \subset V$ and for every assignment $\sigma$ for the variables of $Y$, we have $PROOF(f_{Y,\sigma}) \leq PROOF(f)$.*

**Monotone Boolean functions.** In this paper, by a Boolean function, we shall classically understand a total Boolean function, i.e., such that, the function is defined over the complete domain $\{0,1\}^n$. A Boolean function $f$ over the set of variables $V = \{x_1, \ldots, x_n\}$ is *monotone* (increasing) iff $f(\sigma) \leq f(\sigma')$, for each pair of assignments $\sigma$ and $\sigma'$ such that $x_i(\sigma) \leq x_i(\sigma')$, for $i = 1, \ldots, n$.

For monotone Boolean functions, 0-certificates are called *maxterms* and 1-certificates are called *minterm*. As an example, in the function presented in (1), $\{x_1, x_2\}$ is a minterm and $\{x_2, x_4\}$ is a maxterm. We use $k(f)$ and $l(f)$ to denote

the size of the largest minterm and the largest maxterm of a monotone Boolean function $f$ respectively. Then, $PROOF(f) = \max\{k(f), l(f)\}$.

The following result was first proved in [3] for the class of AND/OR trees and generalized to arbitrary monotone Boolean functions in [4].

**Theorem 1.** *If $f$ is a monotone Boolean function then $\gamma(f) \geq \max\{k(f), l(f)\}$.*

## 3 The Linear Programming Approach

We shall now describe our general schema for the design of algorithms for evaluating functions. We call this methodology the *Linear Programming Approach* ($\mathcal{LPA}$). As suggested by the name itself, this schema is based on the solution of a linear program defined on the variables of the function under consideration and constrained on its certificates.

The linear program $\mathbf{LP_f}$ (see below) is used in the $\mathcal{LPA}$ to estimate how important a variable is in the process of evaluating $f$, that is, its *impact*. It tries to capture the intuitive idea that the relevance of a variable is proportional to the number of certificates it appears in and inversely proportional to the size of these certificates (small certificates tend to include variables with higher impact).

The linear programming approach consists of reading a variable that minimizes the ratio between its evaluation cost and its impact as estimated by the solution available for the linear program $\mathbf{LP_f}$. The cost function is then updated (scaled) in order to charge to every potential proof a fraction of the cost spent by the method. The procedure is then recursively applied on the new instance obtained by fixing the value of the variable just read and using the scaled cost.

**The Linear Programming Approach.** Let $f$ be the function to evaluate and $V$ its set of variables. Let $\mathcal{P} = \{P \subseteq V \mid P \text{ is a certificate for } f\}$. We define the following linear program $\mathbf{LP_f}$ where we have one non-negative real variable $s(x)$ for each variable $x \in V$ and one constraint for each certificate $P \in \mathcal{P}$.

$$\mathbf{LP_f} : \left\{ \text{Minimize } \sum_{x \in V} s(x) : \sum_{x \in P} s(x) \geq 1, \forall P \in \mathcal{P} \text{ and } s(x) \geq 0, \forall x \in V \right\}$$

The procedure below formalizes the linear programming approach. An *implementation* of this meta-algorithm is then obtained by fixing the rule used to choose at each iteration the feasible solution of $\mathbf{LP_{f_Y}}$, where $Y$ is the set of variables already probed.

$\mathcal{LPA}(f, V, c)$
$Y \leftarrow \emptyset$;
**While** the value of $f$ is unknown
  Let $s_Y$ be a feasible solution for $\mathbf{LP_{f_Y}}$.
  Let $u$ be the unread variable $x$ that minimizes $\frac{c(x)}{s_Y(x)}$
  Read($u$)
  **For** each $v \in V \setminus Y$ **do** $c(v) \leftarrow c(v) - s_Y(v) \times \frac{c(u)}{s_Y(u)}$
  $Y = Y \cup \{u\}$
**End While**
**Return** the value of $f$

We shall now present a lemma that is a key tool for the analysis of the implementations of $\mathcal{LPA}$. More precisely, this lemma allows to straightforwardly give an upper bound on the competitiveness of an implementation of the linear programming approach in terms of the feasible solution selected for the linear program. Therefore, in the different implementations presented, we shall simply verify the feasibility of the solution used for the linear program $\mathbf{LP_f}$ and provide a bound on the corresponding objective function. Then, we shall employ this lemma to state the competitiveness of the resulting algorithm.

**Lemma 1.** *Let $\mathbb{LP}$ be an implementation of $\mathcal{LPA}$ and, for each $Y \subset V$ let $s_Y(\cdot)$ be the feasible solution used by $\mathbb{LP}$ when the set of variables already read is $Y$. Then, $\gamma^{\mathbb{LP}}(f) \leq \max_{Y \subset V} \left\{ \sum_{x \in V \setminus Y} s_Y(x) \right\}$*

*Proof.* If $f$ has only one variable the result holds. We assume as induction hypothesis that the result holds for every function that depends on less than $n$ variables. Let $f$ be a function that depends on $n$ variables and let $c(\cdot)$ be a cost function such that $\gamma_c^{\mathbb{LP}}(f) = \gamma^{\mathbb{LP}}(f)$. Furthermore, let $\sigma$ be an assignment for $f$ which maximizes $c^f_{\mathbb{LP}}(\sigma)/c^f(\sigma)$. Let $u$ be the first variable selected by $\mathbb{LP}$. Let us denote $s_\emptyset(\cdot)$ with $s(\cdot)$. Then,

$$c^f_{\mathbb{LP}}(\sigma) \leq \sum_{v \in V} s(v) \left( \frac{c(u)}{s(u)} \right) + \tilde{c}^{f_{\{u\}}}_{\mathbb{LP}}(\sigma_{V \setminus \{u\}}) = \frac{c(u)}{s(u)} \sum_{v \in V} s(v) + \tilde{c}^{f_{\{u\}}}_{\mathbb{LP}}(\sigma_{V \setminus \{u\}}), \quad (2)$$

where $\tilde{c}$ denotes the new cost function after that the costs of the variables in $V$ have been decreased in the **For** loop of the $\mathcal{LPA}$ pseudo-code.

Let $X$ be the cheapest proof for $f$ w.r.t. cost function $c(\cdot)$ and assignment $\sigma$. Moreover, let $X'$ be the cheapest proof for $f_{\{u\}}$ w.r.t. cost function $\tilde{c}$ and assignment $\sigma_{V \setminus \{u\}}$. Note that $X \setminus \{u\}$ is also a proof for $f_{\{u\}}$ w.r.t. assignment $\sigma_{V \setminus \{u\}}$. By the definition of the linear program $\mathbf{LP_f}$ we have $\sum_{v \in X} s(v) \geq 1$. Then,

$$c(X) = \sum_{v \in X} s(v) \left( \frac{c(u)}{s(u)} \right) + \tilde{c}(X \setminus \{u\}) \geq \frac{c(u)}{s(u)} \sum_{v \in X} s(v) + \tilde{c}(X') \geq \frac{c(u)}{s(u)} + \tilde{c}(X'). \quad (3)$$

Putting together (2) and (3) and noting that $\tilde{c}^{f_{\{u\}}}_{\mathbb{LP}}(\sigma_{V \setminus \{u\}})/\tilde{c}(X') \leq \gamma^{\mathbb{LP}}(f_{\{u\}})$, we have that $\gamma^{\mathbb{LP}}(f) = \gamma_c^{\mathbb{LP}}(f) = \dfrac{c^f_{\mathbb{LP}}(\sigma)}{c(X)} \leq \dfrac{\frac{c(u)}{s(u)} \sum_{v \in V} s(v) + \tilde{c}^{f_{\{u\}}}_{\mathbb{LP}}(\sigma_{V \setminus \{u\}})}{\frac{c(u)}{s(u)} + \tilde{c}(X')} \leq$

$\leq \max \left\{ \sum_{v \in V} s(v), \gamma^{\mathbb{LP}}(f_{\{u\}}) \right\}$. Since $f_{\{u\}}$ depends on less than $n$ variables, the induction hypothesis yields

$$\gamma^{\mathbb{LP}}(f) \leq \max \left\{ \sum_{v \in V} s(v), \max_{Y \subset V \setminus \{u\}} \sum_{x \in V \setminus Y} s_Y(x) \right\} \leq \max_{Y \subset V} \sum_{v \in V \setminus Y} s_Y(v).$$

Note that the previous lemma does not rely on any assumption on the structure of $f$. It also motivates the following definition.

**Definition 1.** *The* fractional cover number *of a function $f$ is defined by* $\Delta(f) = \max_{Y \subset V} \left\{ \sum_{x \in V \setminus Y} s^*_Y(x) \right\}$, *where $s^*_Y(\cdot)$ denotes the optimal solution of* $\mathbf{LP_{f_Y}}$.

Using this definition, Lemma 1 states that for every function $f$, we have $\gamma(f) \leq \Delta(f)$. We shall now prove for any function $f \in \mathcal{F}^{\times}$ an upper bound on the fractional cover number of $f$ in terms of the size of its largest proof.

For a point $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$, we use $\|\mathbf{v}\|_p$ to denote the $\ell_p$-norm of $\mathbf{v}$, i.e., $\|\mathbf{v}\|_p = \sqrt[p]{\sum_{i=1}^n v_i^p}$. Given two points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$ we shall denote with $\mathbf{p} \cdot \mathbf{q}$ their dot product, i.e., $\mathbf{p} \cdot \mathbf{q} = \sum_{j=1}^n p_i q_i$.

We shall need the following simple technical result.

**Proposition 3.** *Let $S \subset \mathbb{R}^n$ be a convex set and let $\mathbf{u}$ be a point in $\mathbb{R}^n$. In addition, let $\mathbf{v}$ be a point in $S$ closest to $\mathbf{u}$ in $\ell_2$-norm. Then, $\|\mathbf{u}-\mathbf{w}\|_2 \geq \|\mathbf{v}-\mathbf{w}\|_2$ for all $\mathbf{w} \in S$ .*

**Lemma 2.** *Let $f \in \mathcal{F}^{\times}$ be a non-constant function. Then, $\Delta(f) \leq PROOF(f)$.*

*Proof.* We shall show that for any function $f \in \mathcal{F}^{\times}$ there exists a feasible solution $\mathbf{s}$ for the linear program $\mathbf{LP_f}$ that has $\ell_1$-norm not larger than $PROOF(f)$. We shall use the following geometric construction. For each value $v$ in the range of $f$ we consider the convex hull of the characteristic vectors of the $v$-certificates of $f$. We take the point $\mathbf{p}$ with the smallest $\ell_2$-norm in the union of these convex hulls. Let $\mathbf{p}$ be in the convex hull of the $v$-certificates and take $w \neq v$ in the range of $f$. We prove that the desired $\mathbf{s}$ is given by the closest point (in $\ell_2$-norm) to $\mathbf{p}$ among the points in the convex hull of the $w$-certificates of $f$.

Let $Q$ be the range of $f$. Note that $|Q| \geq 2$, because $f$ is not constant. For each $v \in Q$, let $\mathcal{P}_v$ denote the set of $v$-certificates for $f$. Thus, $\mathcal{P} = \bigcup_{v \in Q} \mathcal{P}_v$.

For each $P \in \mathcal{P}$, let $\mathbf{p}^P = (p_1^P, \ldots, p_n^P) \in [0,1]^n$ be defined by $p_i^P = 1$ if $x_i \in P$, and $p_i^P = 0$, otherwise.[2] Abusing notation, let us denote with $conv(v)$ the convex hull of the set $\{\mathbf{p}^P \mid P \in \mathcal{P}_v\}$.

*Claim.* Let $v, v' \in Q$ with $v \neq v'$. Then, $\mathbf{y} \cdot \mathbf{p}^{P'} \geq 1$ holds for each $\mathbf{y} \in conv(v)$ and for each proof $P' \in \mathcal{P}_{v'}$.

*Proof of the Claim* For each pair of proofs $P \in \mathcal{P}_v, P' \in \mathcal{P}_{v'}$, Proposition 1 assures that $P \cap P' \neq \emptyset$. Thus, $\mathbf{p}^P \cdot \mathbf{p}^{P'} \geq 1$.

Since $\mathbf{y} \in conv(v)$ we have that $\mathbf{y} = \sum_{P \in \mathcal{P}_v} \lambda_P \mathbf{p}^P$, where $\sum_{P \in \mathcal{P}_v} \lambda_P = 1$ and $\lambda_P \geq 0$, for each $P \in \mathcal{P}_v$. Thus, we have $\mathbf{y} \cdot \mathbf{p}^{P'} = \sum_{P \in \mathcal{P}_v} \lambda_P \mathbf{p}^P \cdot \mathbf{p}^{P'} \geq 1$. The proof of the claim is complete.

Now, let us rewrite $\mathbf{LP_f}$ in the following equivalent way.

$$\mathbf{LP_f} : \left\{ \text{Minimize } \|\mathbf{s}\|_1 : \mathbf{s} \cdot \mathbf{p}^P \geq 1, \text{ for every } P \in \bigcup_v \mathcal{P}_v \text{ and } \mathbf{s} \geq \mathbf{0} \right\}$$

[2] We look at the certificates of $\mathcal{P}_v$ as vectors in $\mathbb{R}^n$ with 0 and 1 coordinates.

Let $\mathbf{z}$ be a point with minimum $\ell_2$ norm among the points in $\bigcup_{v\in Q} conv(v)$, i.e., $\|\mathbf{z}\|_2 \leq \|\mathbf{y}\|_2$, for each $\mathbf{y} \in \bigcup_{v\in Q} conv(v)$. Let $v$ be such that $\mathbf{z} \in conv(v)$. In addition, let $v^*$ be an arbitrarily chosen element in $Q - \{v\}$. Let $\mathbf{z}^*$ be the point in $conv(v^*)$ closest to $\mathbf{z}$ in the $\ell_2$-norm, i.e., $\|\mathbf{z} - \mathbf{z}^*\|_2 \leq \|\mathbf{z} - \mathbf{y}^*\|_2$, for any $\mathbf{y}^* \in conv(v^*)$. We shall prove that $\mathbf{z}^*$ is a feasible solution for $\mathbf{LP_f}$ and $\|\mathbf{z}^*\|_1 \leq PROOF(f)$.

For the latter, it is enough to observe that $\mathbf{z}^* \in conv(v^*)$ implies that $\mathbf{z}^* = \sum_{P\in\mathcal{P}_{v^*}} \lambda_P \mathbf{p}^P$, for some non-negative scalars $\lambda_P$ such that $\sum_{P\in\mathcal{P}_{v^*}} \lambda_P = 1$. Thus, $\|\mathbf{z}^*\|_1 = \|\sum_{P\in\mathcal{P}_{v^*}} \lambda_P \mathbf{p}^P\|_1 = \sum_{P\in\mathcal{P}_{v^*}} \lambda_P |P| \leq PROOF(f)$.

We now prove the feasibility of $\mathbf{z}^*$. By the above claim, we immediately have that $\mathbf{z}^* \cdot \mathbf{p}^P \geq 1$ for any $P \in \mathcal{P}_u$, with $u \in Q$, $u \neq v^*$. It remains to prove that $\mathbf{z}^* \cdot \mathbf{p}^P \geq 1$, for each $P \in \mathcal{P}_{v^*}$.

By Proposition 3, we have $||\mathbf{z}^* - \mathbf{y}^*||_2 \leq ||\mathbf{z} - \mathbf{y}^*||_2$, for each $\mathbf{y}^* \in conv(v^*)$. In particular, for each $P \in \mathcal{P}_{v^*}$, we have

$$(||\mathbf{z}^* - \mathbf{p}^P||_2)^2 \leq (||\mathbf{z} - \mathbf{p}^P||_2)^2. \tag{4}$$

Recall that for any $\mathbf{p}$ and $\mathbf{q}$ it holds that $(\|\mathbf{p} - \mathbf{q}\|_2)^2 = (\|\mathbf{p}\|_2)^2 + (\|\mathbf{q}\|_2)^2 - 2\mathbf{p} \cdot \mathbf{q}$. Thus, (4) becomes

$$(||\mathbf{z}^*||_2)^2 - 2\mathbf{z}^* \cdot \mathbf{p}^P \leq (||\mathbf{z}||_2)^2 - 2\mathbf{z} \cdot \mathbf{p}^P. \tag{5}$$

By the choice of $\mathbf{z}$ we have that $||\mathbf{z}||_2 \leq ||\mathbf{z}^*||_2$, which together with (5) yields $\mathbf{z}^* \cdot \mathbf{p}^P \geq \mathbf{z} \cdot \mathbf{p}^P \geq 1$, where the last inequality follows by the above claim.

Summarizing, we have shown that for any function $f$ satisfying the hypothesis, the optimal solution $s^*$ of $\mathbf{LP_f}$, has cost not greater than $PROOF(f)$. In particular, for every restriction $f_Y$ of $f$, denoting with $s^*_Y$ the optimal solution of $\mathbf{LP_{f_Y}}$, we have $\|s^*_Y\|_1 \leq PROOF(f_Y) \leq PROOF(f)$. This concludes the proof.

Directly from Lemmas 1 and 2 and Theorem 1 we have the following results.

**Theorem 2.** *Let $f = f(x_1, x_2, \ldots, x_n)$ be a non-constant function in $\mathcal{F}^\times$. Then, $\gamma(f) \leq PROOF(f)$.*

**Corollary 1.** *For every monotone Boolean function $\gamma(f) = PROOF(f)$.*

**Some remarks on the issue of efficiency.** We have shown that an optimal implementation of the $\mathcal{LPA}$ can be used to obtain an algorithm for evaluating monotone Boolean functions with the optimal extremal competitive ratio. By an optimal implementation we mean one that always uses the optimal solution to the $\mathbf{LP_f}$. We shall now touch upon the issue of whether or when this can be done efficiently. Due to the space limitation, we shall defer a more formal treatment of the following material to the full version of the paper.

Let $f$ be a monotone Boolean function over $n$ variables. Clearly the existence of a polynomial time implementation of the $\mathcal{LPA}$ for $f$ critically depends on the representation of $f$ which is given. We shall here assume that $f$ is given as the list of its minterms. Let $A$ be the binary matrix whose rows are given by the characteristic vectors of the minterms of $f$, indexed over the set of the $n$ variables

of $f$. Let $B$ be the binary matrix whose rows are the characteristic vectors of the maxterms of $f$, i.e., the minimal hitting sets of the family of the minterms. An optimal implementation of the $\mathcal{LPA}$ has to find the vector of minimum $\ell_1$-norm in the polyhedron $\mathcal{A} = \{\mathbf{p} \mid A\mathbf{p} \geq 1, B\mathbf{p} \geq 1, \mathbf{p} \geq 0\}$. We observe that the the number of maxterms can be exponential in the size of the representation of $f$. Furthermore, the separation problem of $\mathbf{LP_f}$ consists of solving the hitting set problem, a well known NP-complete problem. Therefore, for all practical purposes, rather than trying to directly solve $\mathbf{LP_f}$, it seems reasonable to look for "good" feasible solutions for $\mathbf{LP_f}$ that can be constructed in polynomial time. To this aim, we can try to construct another polyhedron $\mathcal{B}$ with the following properties: (i) it is contained in the polyhedron $\mathcal{A}$ and (ii) linear functions can be optimized over it in polynomial time. Clearly, the point of minimum $\ell_1$-norm in $\mathcal{B}$ gives us a feasible solution for $\mathbf{LP_f}$ in polynomial time. We aim at having such minimum in $\mathcal{B}$ not much larger than the minimum in $\mathcal{A}$.

Let $\mathcal{M} = \{\mathbf{z} \in [0,1]^n \mid A\mathbf{z} \geq \mathbf{1}\}$ and define $\mathcal{B} = \{\mathbf{p} \mid \mathbf{p} \geq \mathbf{0}, A\mathbf{p} \geq \mathbf{1},\ \mathbf{p} \cdot \mathbf{z} \geq 1\, \forall \mathbf{z} \in \mathcal{M}\}$. Since every row $\mathbf{y}$ in the maxterm matrix $B$ satisfies $A\mathbf{y} \geq 1$, we have that $\mathbf{y} \in \mathcal{M}$ and then $\mathcal{B} \subseteq \mathcal{A}$, as desired. Moreover, although $\mathcal{B}$ is defined by an infinite number of constraints (one for any $\mathbf{z} \in \mathcal{M}$) the corresponding separation problem can be solved in polynomial time. In fact, given a point $\mathbf{p} \in \mathbb{R}^n$, we can use the linear program $\mathbf{LP^B} : \{$ Minimize $\mathbf{p} \cdot \mathbf{z} : A\mathbf{z} \geq \mathbf{1},\ \mathbf{z} \geq 0\}$ to verify whether $\mathbf{p}$ is in $\mathcal{B}$ or not. It is easy to see that $\mathbf{p} \in \mathcal{B}$ if and only if the cost of the optimal solution to $\mathbf{LP^B}$ is not smaller than 1 and $A\mathbf{p} \geq 1$. In addition, if $\mathbf{u}$ ($\mathbf{v}$) is the characteristic vector of an arbitrary minterm (maxterm) of $f$ then $\mathbf{z} = \mathbf{u} + \mathbf{v}$ belongs to $\mathcal{B}$ and $||\mathbf{z}||_1 \leq 2PROOF(f)$. By a more refined construction we can prove that there exists a point $\mathbf{p} \in \mathcal{B}$ such that $\|\mathbf{p}\|_1 \leq 2PROOF(f) - \sqrt{k(f), \ell(f)}$. This implies that an implementation of the $\mathcal{LPA}$ that as feasible solution of $\mathbf{LP_f}$ uses the point with minimum $\ell_1$ norm of $\mathcal{B}$ has competitive ratio at most $2PROOF(f) - \sqrt{k(f), \ell(f)}$, and is polytime. This matches the competitiveness of the best known polytime algorithm for evaluating monotone Boolean functions[4]. We do not know if this bound is tight or not, that is, whether a precise estimation of the minimum $\ell_1$-norm over all points in $\mathcal{B}$ could improve the factor of 2 in the competitive ratio.

Optimizing a linear function over the polyhedron $\mathcal{B}$ can be slow for practical applications since one needs to use the ellipsoid method. However, by using the theory of blocking polyhedra (see, e.g., [9, Ch. 30]), it is possible to prove that $\mathcal{B}$ is the projection into $\mathbb{R}^n$ of the polyhedron $\mathcal{C} = \{(\mathbf{y}, \mathbf{w}) \mid \mathbf{y} \in \mathbb{R}^n,\ \mathbf{w} \in \mathbb{R}^m_+,\ \|\mathbf{w}\|_1 = 1,\ A\mathbf{y} \geq \mathbf{1},\ \mathbf{y} \geq \mathbf{w}^T A\}$, defined in $\mathbb{R}^{m+n}$, where $m$ is the number of minterms. Since $\mathcal{C}$ is defined by a polynomial (in fact linear) number of constraints, the point of minimum $\ell_1$-norm in $\mathcal{B}$ can be quickly obtained by projecting in $\mathbb{R}^n$ the point of $\mathcal{C}$ which minimizes the sum of the first $n$ coordinates.

We remark that for functions that have a compact circuit representation we can obtain a polynomial time algorithm if the separation problem for $\mathbf{LP_f}$ is solvable in polynomial time. This is the case, e.g., of previously studied classes of functions like threshold trees and, AND/OR trees (see, e.g., [3,4,5,6]).

The existence of a polytime algorithm with competitive ratio $PROOF(f)$ for evaluating monotone Boolean functions remains an open problem.

## 4 Final Remarks: Beyond Monotone Boolean Functions

In this section we shall give more evidence of the power of the $\mathcal{LPA}$ and its broad applicability. We shall first discuss the case of arbitrary Boolean functions and then outline how the $\mathcal{LPA}$ can be used to obtain efficient and highly competitive algorithms for the functions: finding the minimum and sorting.

**Arbitrary Boolean functions.** Since the class of boolean functions belong to $\mathcal{F}^{\times}$ it follows from Theorem 2 that $\gamma(f) \leq PROOF(f)$ for every boolean function $f$. Then, it it is natural to ask whether Corollary 1 holds without the monotonicity assumption, that is, whether $\gamma(f) = PROOF(f)$ for every boolean function $f$. We now give an example showing that $\gamma(f)$ can be smaller than $PROOF(f)$. Let, e.g., $f = (z \text{ AND } x_1) \text{ OR } (z \text{ AND } x_2) \text{ OR } (\bar{z} \text{ AND } x_3) \text{ OR } (\bar{z} \text{ AND } x_4)$. We have $PROOF(f) \geq 4$ since $\{x_1, x_2, x_3, x_4\}$ is a 0-certificate for the assignment $(x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0, z = 0)$. A careful inspection shows that $\Delta(f) = 3$ which implies, by Lemma 1, that $\gamma(f) < PROOF(f)$.

We shall note however that $\gamma(f) = \Delta(f)$ for the above function. In fact, we do not know any example for which $\gamma(f) < \Delta(f)$. The above example can be generalized to show that that $\gamma(f)$ can be much smalller than $PROOF(f)$.

**Sorting and Finding the minimum.** The $\mathcal{LPA}$ can be implemented to provide very competitive solutions to the problems of "sorting" and "finding the minimum" in the context of function evaluation with costs. We want to compute the functions $f^{sort} \equiv sort(v_1, \ldots, v_n)$ and $f^{min} \equiv \min\{v_1, \ldots, v_n\}$ when $v_1, \ldots, v_n$ are variables taking values in some totally ordered set. As is customary, the performance of algorithms for such problems is in terms of the comparisons that it performs We shall assume that each pair of variables, $v_i, v_j$ has an associated cost $c(v_i, v_j)$ to be paid to compare them. Different comparisons may incur different costs.

As model we use a complete weighted graph in which the set of vertices is given by the set of variables $S = \{v_1, \ldots, v_n\}$ and each edge represents the comparison between the incident variables. The weight of an edge is the cost incurred to execute the comparison it represents. We can view the outcome of the comparison as the operation of disclosing the orientation of the edge, assuming that the orientation goes from the smaller to the larger variable. A feasible assignment $\sigma$ is a transitive orientation of the edges. In the case of $f^{min}$, a proof for the identification of the minimum w.r.t. a feasible assignment $\sigma$ is a set of edges $P$ such that, when oriented according to $\sigma$, there is a vertex $v$ which reaches every other vertex in the graph $G_P = (S, P)$. For $f^{sort}$ a minimal proof is a Hamiltonian path such that at least one of its two possible orientations is compatible with the assignment $\sigma$. The following summarizes our results.

**Proposition 4.** *Let* $f \in \{f^{sort}, f^{min}\}$. *Then,* $\Delta(f) = n - 1$.

This implies the bound $\gamma(f^{min}) \leq n-1$ previously given in [3,11]. The exact value $\gamma(f^{min}) = n-2$ was given in [4] by an *ad hoc* implementation of the General Approach. Moreover, by Proposition 4, it also follows that $\gamma^{\mathcal{LPA}}(f^{sort}) \leq n-1$ which improves (the constant of) the best known result for sorting [11]. The problem of determining the exact value of $\gamma(f^{sort})$ remains open.

**Final open questions.** We have shown that in the class of monotone Boolean functions and also in some special proper superclass of it we have $\gamma(f) = \Delta(f)$. On the other hand, we know that for some functions outside $\mathcal{F}^{\times}$ like $f^{min}$ we can prove a strict inequality, in fact, $n-2 = \gamma(f^{min}) < \Delta(f^{min}) = n-1$. Another negative example is given by the function searching for an element in a sorted list, also considered in [3]. In this case, we have $O(\log n) \leq \gamma(f) << \Delta(f) = n$.

An interesting question is the following. "Does $\gamma(f) = \Delta(f)$ hold for each $f \in \mathcal{F}^{\times}$?" This class already contains all the (total) Boolean functions.

## References


1. Boros, E., Ünlüyurt, T.: Diagnosing double regular systems. Annals of Mathematics and Artificial Intelligence 26(1-4), 171–191 (1999)
2. Buhrman, H., de Wolf, R.: Complexity measures and decision tree complexity: a survey. Theoretical Computer Science 288(1), 21–43 (2002)
3. Charikar, M., Fagin, R., Guruswami, V., Kleinberg, J.M., Raghavan, P., Sahai, A.: Query strategies for priced information. Journal of Computer and System Sciences 64(4), 785–819 (2002)
4. Cicalese, F., Laber, E.S.: A new strategy for querying priced information. In: Proc. of STOC 2005, pp. 674–683. ACM Press, New York (2005)
5. Cicalese, F., Laber, E.S.: An Optimal Algorithm for Querying Priced Information: Monotone Boolean Functions and Game Trees. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 664–676. Springer, Heidelberg (2005)
6. Cicalese, F., Laber, E.S.: On the competitive ratio of evaluating priced functions. In: Proc. of SODA 2006, pp. 944–953 (2006)
7. Duffuaa, S.O., Raouf, A.: An optimal sequence in multicharacteristics inspection. J. Optim. Theory Appl. 67(1), 79–86 (1990)
8. Gillies, D.W.: Algorithms to schedule tasks with and/or precedence constraints. PhD thesis, Champaign, IL, USA (1993)
9. Graham, R.L., Grötschel, M., Lovász, L. (eds.): Handbook of Combinatorics. The Mit Press - North-Holland (1996)
10. Greiner, R., Hayward, R., Jankowska, M., Molloy, M.: Finding optimal satisficing strategies for and-or trees. Artificial Intelligence 170(1), 19–58 (2006)
11. Gupta, A., Kumar, A.: Sorting and selection with structured costs. In: Proc. of FOCS 2001, pp. 416–425. IEEE, Los Alamitos (2001)
12. Heiman, R., Newman, I., Wigderson, A.: On read-once threshold formulae and their randomized decision tree complexity. TCS 107(1), 63–76 (1993)
13. Heiman, R., Wigderson, A.: Randomized vs. deterministic decision tree complexity for read-once boolean functions. Comput. Complexity 1, 311–329 (1991)
14. Hellerstein, J.M.: Optimization techniques for queries with expensive methods. ACM Transactions on Database Systems 23(2), 113–157 (1998)
15. Kannan, S., Khanna, S.: Selection with monotone comparison cost. In: Proc. of SODA 2003, pp. 10–17. ACM/SIAM (2003)

16. Louis AnthonyCox, J., Qiu, Y., Kuehner, W.: Heuristic least-cost computation of discrete classification functions with uncertain argument values. Ann. Oper. Res. 21(1-4), 1–30 (1989)
17. Qiu, Y., Anthony Cox Jr., L., Davis, L.: Guess-and-verify heuristics for reducing uncertainties in expert classification systems. In: Dubois, D., Wellman, M.P. (eds.) UAI, pp. 252–258. Morgan Kaufmann, San Francisco (1992)
18. Rivest, R.L., Vuillemin, J.: On recognizing graph properties from adjacency matrices. TCS 3(3), 371–384 (1976)
19. Saks, M., Wigderson, A.: Probabilistic Boolean decision trees and the complexity of evaluating game trees. In: Proc. of FOCS 1986, pp. 29–38. IEEE, Los Alamitos (1986)
20. Snir, M.: Lower bounds on probabilistic linear decision trees. TCS 38, 69–82 (1985)
21. Tarsi, M.: Optimal search on some game trees. JACM 30(3), 389–396 (1983)