Universität Bielefeld

Technische Fakultät
Abteilung Informationstechnik
Forschungsberichte

Computing with Priced Information: game trees and the value dependent cost model.

Ferdinando Cicalese Martin Milanič

Report 2008-03

Impressum: Herausgeber:
Ellen Baake, Mario Botsch, Robert Giegerich, Ralf Hofestädt, Franz Kummert, Peter Ladkin, Ralf Möller, Tim Nattkemper, Helge Ritter, Gerhard Sagerer, Jens Stoye, Ipke Wachsmuth

Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31,
33501 Bielefeld, Germany

ISSN 0946-7831

Computing with Priced Information: Game Trees and the Value Dependent Cost Model [⋆]

Ferdinando Cicalese and Martin Milanič

AG Genominformatik, Faculty of Technology, Bielefeld University, Germany
{nando, mmilanic}@cebitec.uni-bielefeld.de

Abstract. We study the function evaluation problem in the priced information framework introduced in [Charikar *et al.* 2002]. We characterize the best possible extremal competitive ratio for the class of game tree functions. Moreover, we extend the above result to the case when the cost of reading a variable depends on the value of the variable. In this new value dependent cost variant of the problem, we also exactly evaluate the extremal competitive ratio for the whole class of monotone Boolean functions.

1 Introduction

Problem Statement. A function f over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$ is given and we want to determine the value of f for a fixed but unknown *assignment* σ, i.e., a choice of the values for the variables of V. We are allowed to adaptively read the values of the variables. Each variable x_i has an associated non-negative cost c_{x_i} which is the cost incurred 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 identify and read a minimum cost set of variables $U \subseteq V$ whose values uniquely determine the value $f(\sigma) = f(x_1(\sigma), \ldots, x_n(\sigma))$ of f w.r.t. the given assignment σ, regardless of the values of the variables not probed. We say that such a set $U \subseteq V$ is a *proof* for f with respect to the assignment σ.

An evaluation algorithm $\mathcal{A}$ for f adaptively reads the variables in V until the set of variables read so far is a proof for the value of f. Given a cost assignment $c = (c_{x_1}, \ldots, c_{x_n})$, we let $c^f_{\mathcal{A}}(\sigma)$ denote the total cost incurred by the algorithm $\mathcal{A}$ to evaluate f under the assignment σ and $c^f(\sigma)$ the cost of the cheapest proof for f under the assignment σ. We say that $\mathcal{A}$ is ρ-competitive if $c^f_{\mathcal{A}}(\sigma) \leq \rho c^f(\sigma)$, for every possible assignment σ. We use $\gamma^{\mathcal{A}}_c(f)$ to denote the competitive ratio of $\mathcal{A}$, that is, the minimum ρ for which $\mathcal{A}$ is ρ-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 $\mathcal{A}$.

The extremal competitive ratio $\gamma^{\mathcal{A}}(f)$ of an algorithm $\mathcal{A}$ is defined by $\gamma^{\mathcal{A}}(f) = \max_c \gamma^{\mathcal{A}}_c(f)$. The best possible extremal competitive ratio for any deterministic algorithm is $\gamma(f) = \min_{\mathcal{A}} \gamma^{\mathcal{A}}(f)$. This is a measure of the structural complexity of f independent of a particular cost assignment and algorithm.

⋆ This work was supported by the Sofja Kovalevskaja Award 2004 of the Alexander von Humboldt Foundation and the Bundesministerium für Bildung und Forschung.

Function evaluation problems have been extensively studied in AI, particularly in the theory of computer aided games. The strategic evolution of the game is usually modeled by a so called game tree [7, 18, 16]. In a game tree (formalized in Sect. 4), each node represents a state of the game. The root is the current state. For each node/state ν, its children are the set of possible states reachable from ν given the moves available to the player moving in state ν. The possible moves for the two players are represented by alternating levels of edges. A game tree of a certain depth is built by a computer in order to explore the possible developments of the game from the current position. By assigning to each leaf-state an estimate of the "goodness" of that state for the computer-player, it is possible to evaluate all the inner states. The most fruitful move for the computer is the one corresponding to the edge from the root to its children of maximum value. In general, the evaluation of some leaf state might involve expensive computations. Since, on the other hand, not all the leaf-state evaluations are needed to compute the node of maximum value in the first level, we have here an instance of the problem of evaluating a function by only looking at a *cheap* set of its variables.

In this paper we characterize the extremal competitiveness for the class of game tree functions. Moreover, we also study the function evaluation problem when the cost of reading a variable depends on the value of the variable. The evaluation algorithm knows the cost $c_x(y)$ of reading x when $x(\sigma) = y$, for each variable x and for each value y that the variable x can take.

This above model has applications in several situations. Consider, e.g., the decision making process of a physician—or of a computer aided decision making system—who has to decide the cheapest sequence of tests to perform on a patient in order to reliably diagnose a given disease. Different tests typically involve different costs. In this framework, costs are usually understood in an extended meaning encompassing the actual monetary costs, the distress of the patient, and the possible side-effects of the tests. Also, tests' costs might be dependent on the outcome: a single lab analysis might undergo several phases, some of which are only performed depending on the result of the previous ones. Analogously, there are tests that if positive, are necessarily followed by a sequence of other tests—on which the decision maker has no alternative. In this case, the cost of the "triggering" test can be considered as the sum of the whole set, in case its outcome is positive, and only its own cost if the outcome is negative. It is then natural to consider models in which tests' costs are dependent on the outcome of the test itself. We refer the interested reader to [21] and references quoted therein for a remarkable account of several types of costs to be considered in inference procedures.

Besides the two examples above, function evaluation problems are found in a plethora of different areas both in theoretical and applied computer science like telecommunications [14], manufacturing [8], computer networks [9], satisficing search problems [10]. For more on automatic diagnosis problems and computer aided medical systems see also [1, 15] and references therein. Finally, the function evaluation problem arises in query optimization, a major issue in databases [13].

Our Results. We obtain the tight extremal competitive ratio of monotone Boolean functions in the new value dependent cost model extending the previous result of [6]. This is achieved via an adaptation of the Linear programming based approach of [6].

Outside the Boolean realm, we focus on the class of game tree functions. We obtain the tight extremal competitive ratio for game trees. In particular we show that for any game tree function f, but for special cases that we also characterize, the extremal competitiveness $\gamma(f)$ is equal to the maximum size of a *certificate* for f, i.e., of a minimal set of variables which allow to prove an upper or lower bound on the value of f, for some assignment σ. In fact, we provide a polynomial algorithm with competitiveness $\gamma(f)$ for any game tree function f. We also extend this result to the value dependent cost model. Our result significantly improves the previous best known result in [4], where a polynomial time algorithm was provided which achieves $\gamma(f)$ competitiveness over a restricted set of assignments, namely only those σ's for which exactly one variable has value $f(\sigma)$.

Related Work. Most of the earlier work on function evaluation problems was done in the classical unitary cost model for both deterministic and randomized algorithms or assuming some statistical knowledge on the values of the variables (see, e.g., [20, 19, 17, 12]). The competitive analysis scenario was proposed by Charikar *et al.* in [2] where several classes of functions were studied in this novel framework, including the class of game trees. For game trees, Charikar *et al.* [2] presented a pseudo-polynomial time algorithm with competitiveness $2\gamma_c^f$. The extremal competitiveness for game trees was also studied in [4] where a polynomial time algorithm was provided achieving competitiveness $\gamma(f)$ for any assignment σ such that there exists exactly one variable with value $f(\sigma)$. In [5] the authors showed a polynomial time algorithm with competitiveness $4\gamma_c^f$. However, to date, there was no complete and exact characterization of the optimal competitiveness for the evaluation of game trees.

All the above results are for the case when the cost is independent of the value of the variable. In fact, this is the first paper taking into account the dependency of costs on the values in the competitive analysis scenario. In [1] function evaluation with value dependent costs was also discussed, even though in the probabilistic model considered in [1] the dependency on the values can be absorbed in the distribution assigned to the values of the variables. In [5], the case of unknown costs was also considered. This is an attempt to address cases in which the algorithm has a reduced knowledge on the cost assignment. It is important to notice that the model of [5] cannot be used to solve the type of problems addressed here, and vice versa.

2 Preliminaries: the Linear Programming Approach

Let $f : D \rightarrow \mathbb{R}$ be a real-valued function defined over a set of variables $V = \{x_1, \ldots, x_n\}$, where $D \subseteq \mathbb{R}^n$. For $x \in V$, let $D(x)$ denote the set of possible values that the variable x can take in the elements of the domain of f, that is, $D(x)$ is the projection of the set D on the x coordinate. For $x \in V$ and $y \in D(x)$, let $c_x(y) \geq 0$ denote the cost for querying the variable x, given that the value of x in the (unknown) assignment σ is $x(\sigma) = y$. Furthermore, let $c_x^{min} = \min\{c_x(y) : y \in D(x)\}$ and $c_x^{max} = \max\{c_x(y) : y \in D(x)\}$, for all $x \in V$. We allow that the costs of querying a certain variable are *value dependent*. In other words, the functions $c_x(y)$ are not necessarily constant as functions of y, i.e., it is possible that $c_x^{min} \neq c_x^{max}$.

We assume that a bound $r \geq 1$ is fixed (and known to the algorithm) on the maximum possible ratio between two costs of queries for a single variable. More precisely, we assume that the cost function satisfies, for all $x \in V$, $c_x^{max} \leq r c_x^{min}$. Equivalently, for every $x \in V$, and for every $y_1, y_2 \in D(x)$, we have $0 \leq c_x(y_1) \leq r c_x(y_2)$. The set of all such assignments of cost functions will be denoted by $\mathcal{C}_r(f)$. Note that without such bound, no algorithm could guarantee any competitiveness.

In order to make explicit the dependency of our results on the bound r, we shall now rephrase the definition of the competitive measures.

Definition 1 *Let $r \geq 1$. The r-extremal competitive ratio of a function $f : D \to \mathbb{R}$, where $D \subseteq \mathbb{R}^n$, is defined as $\gamma_r(f) = \min_{\mathbb{A}} \gamma_r^{\mathbb{A}}(f)$ where the minimum is taken over all deterministic algorithms that evaluate f, and where $\gamma_r^{\mathbb{A}}(f) = \max_{c \in \mathcal{C}_r(f)} \max_{\sigma \in D} \frac{c_{\mathbb{A}}^f(\sigma)}{c^f(\sigma)}$.*

It is not hard to see that every ρ-competitive algorithm for the value independent cost model is an $(r \times \rho)$-competitive algorithm in the value dependent cost model. Therefore, $\gamma_r(f) \leq r\gamma(f)$. However, we shall see that this estimate of $\gamma_r(f)$ loses an additive term of $r-1$. For this we devise a variant of the Linear Programming Approach introduced in [6] which is adapted to the value dependent cost model. We denote this new scheme by $\mathcal{LPA}^*$.

In order to describe the $\mathcal{LPA}^*$ we shall need some new notation. Let $\mathcal{P}(f)$ denote the set of inclusion-wise minimal proofs of f, i.e., the family of sets X such that there exists at least one assignment σ with respect to which X is a proof for f, while no subset of X is. Consider the following linear program $\mathbf{LP_f}$:

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

Suppose that the set of variables already read is Y. We shall denote with f_Y the *restriction* of f with respect to Y, that is, the function over $V \backslash Y$ obtained from f by fixing the values of the variables in Y as given by the valued read so far, according to the underlying fixed and unknown assignment σ. Let s_Y be a feasible solution to the linear program $\mathbf{LP_{f_Y}}$. The $\mathcal{LPA}^*$ chooses a variable u that minimizes the value of $\frac{c_x^{min}}{s_Y(x)}$. (For definiteness, we let $\frac{0}{0} := 0$. This assures that the variables of zero cost are always queried before the others.) Then, the cost assignment c is updated to a new cost assignment $\tilde{c}$ defined as follows: For $x \in V \backslash (Y \cup \{u\})$ and $y \in D(x)$, we let

$$\tilde{c}_x(y) = c_x(y) - \delta c_x(y) \quad \text{where} \quad \delta c_x(y) = c_x(y) \cdot \frac{c_u^{min}}{c_x^{min}} \cdot \frac{s_Y(x)}{s_Y(u)}. \tag{1}$$

Note that the quantities $\delta c_x(y)$ are well-defined. More importantly, the values of $\tilde{c}_x(y)$ are chosen so that $\tilde{c} \in \mathcal{C}_r(f_{\{u\}})$. (To see this, observe that equality $\tilde{c}_x(y_1)/\tilde{c}_x(y_2) = c_x(y_1)/c_x(y_2)$ holds for every $x \in V \backslash \{u\}$ and every $y_1, y_2 \in D(x)$.) The above procedure is repeated over $f_{Y \cup \{u\}}$ using the new costs $\tilde{c}$, until the value of f is determined.

The linear programming approach for the value dependent cost model is formally described in Fig. 1, where for the sake of efficiency, for each $x \in V \setminus Y$ only c_x^{min} is actually updated. 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 $\frac{c_x^{min}}{s_Y(x)}$.
        Read($u$)
        **For** each $v \in V \setminus Y$ **do** $c_v^{min} \leftarrow c_v^{min} - s_Y(v) \times \frac{c_u^{min}}{s_Y(u)}$
        $Y = Y \cup \{u\}$
**End While**
**Return** the value of $f$
```

Fig. 1. The "value dependent cost" Linear Programming Approach

Lemma 1 *Let $\mathbb{LP}$ be an implementation of the $\mathcal{LPA}^*$. 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, for every $r \geq 1$,*

$$\gamma_r^{\mathbb{LP}}(f) \leq r \cdot \max_{Y \subset V} \left\{ \sum_{v \in V \setminus Y} s_Y(v) \right\} - r + 1.$$

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. Let $c \in \mathcal{C}_r(f)$ be a cost function such that $\gamma_c^{\mathbb{LP}}(f) = \gamma_r^{\mathbb{LP}}(f)$, and let σ be an assignment for f that maximizes the ratio $c_{\mathbb{LP}}^f(\sigma)/c^f(\sigma)$. For $U \subseteq V$, we denote $c(U) = \sum_{x \in U} c_x(x(\sigma))$. Furthermore, let X be a cheapest proof for f w.r.t. cost function c and assignment σ. Let us denote $s_\emptyset(\cdot)$ with $s(\cdot)$. It is not hard to see that the 0-cost variables do not affect the competitiveness of $\mathbb{LP}$. Then, let u be the first variable selected by $\mathbb{LP}$ with $c_u^{min} > 0$. Therefore, in particular, $c_u(u(\sigma)) > 0$ and $c_x^{min} > 0$ for all variables $x \in V$. (Here and throughout the proof, c_x^{min} denotes the value before the update.) For $x \in V \setminus \{u\}$ and $y \in D(x)$, we define the new cost function $\tilde{c}(\cdot)$ as in (1).

The total amount that the algorithm spends on f to prove the value of σ is at most the total amount of change in the costs, summed over all the variables, plus the amount that the algorithm spends on the remaining iterations, that is, the cost spent on $f_{\{u\}}$ to prove the value of $\sigma_{V \setminus \{u\}}$ with respect to the new costs $\tilde{c}(\cdot)$. In formulae:

$$c_{\mathbb{LP}}^f(\sigma) \leq \sum_{v \in V} \delta c_v(v(\sigma)) + \tilde{c}_{\mathbb{LP}}^{f_{\{u\}}}(\sigma_{V \setminus \{u\}}) = \frac{c_u^{min}}{s(u)} \cdot \sum_{v \in V} \frac{c_v(v(\sigma))}{c_v^{min}} \cdot s(v) + \tilde{c}_{\mathbb{LP}}^{f_{\{u\}}}(\sigma_{V \setminus \{u\}}), \tag{2}$$

where $\tilde{c}$ is the cost function defined above.

Let X' be a cheapest proof for $f_{\{u\}}$ w.r.t. cost function $\tilde{c}$ and assignment $\sigma_{V \setminus \{u\}}$. Recall that X is a cheapest proof for f w.r.t. cost function c and assignment σ. Note that $X \setminus \{u\}$ is also a proof for $f_{\{u\}}$ w.r.t. assignment $\sigma_{V \setminus \{u\}}$. Then,

$$c(X) = \sum_{v \in X} \delta c_v(v(\sigma)) + \tilde{c}(X \setminus \{u\}) \geq \frac{c_u^{min}}{s(u)} \cdot \sum_{v \in X} \frac{c_v(v(\sigma))}{c_v^{min}} \cdot s(v) + \tilde{c}(X'). \tag{3}$$

Putting together the inequalities (2) and (3) and noting that $\tilde{c}_{\mathbb{LP}}^{f_{\{u\}}}(\sigma_{V\setminus\{u\}})/\tilde{c}(X') \leq \gamma_r^{\mathbb{LP}}(f_{\{u\}})$, we have

$$\gamma_r^{\mathbb{LP}}(f) = \gamma_c^{\mathbb{LP}}(f) = \frac{c_{\mathbb{LP}}^f(\sigma)}{c(X)} \leq \max\left\{\frac{\sum_{v\in V}\frac{c_v(v(\sigma))}{c_v^{min}}\cdot s(v)}{\sum_{v\in X}\frac{c_v(v(\sigma))}{c_v^{min}}\cdot s(v)}, \gamma_r^{\mathbb{LP}}(f_{\{u\}})\right\}.$$

We shall now bound the first term in the maximum. In order to simplify formulas, let us write τ_v for $c_v(v(\sigma))/c_v^{min}$. We have that the first term in the maximum becomes

$$\frac{\sum_{v\notin X}\tau_v s(v) + \sum_{v\in X}\tau_v s(v)}{\sum_{v\in X}\tau_v s(v)} \leq \frac{\sum_{v\notin X} r s(v)}{\sum_{v\in X} s(v)} + 1 = \frac{r\sum_{v\in V} s(v) - r\sum_{v\in X} s(v)}{\sum_{v\in X} s(v)} + 1$$
$$\leq r\sum_{v\in V} s(v) - r + 1,$$

where the first inequality follows by $1 \leq \tau_v \leq r$; the equality by writing the summation over $V \setminus X$ as the difference between the summation over V and the one over X; the second inequality follows because $\sum_{v\in X} s(v) \geq 1$ by definition of the linear program $\mathbf{LP_f}$ and the fact that X is a minimal proof for f. Therefore we have

$$\gamma_r^{\mathbb{LP}}(f) \leq \max\left\{r\sum_{v\in V} s(v) - r + 1, \gamma_r^{\mathbb{LP}}(f_{\{u\}})\right\}.$$

and since $f_{\{u\}}$ depends on less than n variables, the induction hypothesis yields the desired result. $\square$

3 Monotone Boolean Functions

By virtue of the above result, it is not hard to provide an upper bound on the extremal competitiveness for monotone Boolean functions in the value dependent cost model. Let $\Delta(f) = \max_{Y,\sigma}\left\{\sum_{v\in V\setminus Y} s^*_{Y,\sigma}(v)\right\}$, where the maximum is taken over all possible restrictions $f_{Y,\sigma}$ of f (i.e., $f_{Y,\sigma}$ is defined by an assignment σ of the values to the variables in $Y \subset V$), and where $s^*_{Y,\sigma}(\cdot)$ denotes an optimal solution of $\mathbf{LP_{f_{Y,\sigma}}}$. Recently, Cicalese and Laber have proved in [6] that for a large class of functions, which includes all Boolean functions, $\Delta(f)$ is bounded above by $PROOF(f)$, the size of a largest minimal proof of f. In particular, in conjunction with Lemma 1 this implies that for every $r \geq 1$ and for every Boolean function f, it holds that $\gamma_r(f) \leq r \cdot PROOF(f) - r + 1$.

We shall now provide a lower bound that matches the above upper bound. For monotone Boolean functions, minimal proofs are usually referred to as *maxterms* and *minterms*. A maxterm (minterm) can be defined as a minimal set of variables such that for any σ that sets their value to 0 (1) we have $f(\sigma) = 0$ ($f(\sigma) = 1$). This is used in the following lemma which provides the matching lower bound by generalizing a construction of [2] and [3]. We use $k(f)$ and $l(f)$ to denote the size of the largest minterm and the largest maxterm of f respectively. Thus, $PROOF(f) = \max\{k(f), l(f)\}$.

Theorem 1 *Let f be a monotone Boolean function. Then $\gamma_r(f) \geq r \cdot PROOF(f) - r + 1$.*

Proof. Consider an algorithm $\mathbb{A}$ for evaluating f. We construct an assignment $\sigma^{\mathbb{A}}$ which is 'bad' for $\mathbb{A}$. Let C be a largest minterm of f, i.e., $|C| = k(f)$. For $x \in C$, we set $c_x(1) = r$, and $c_x(0) = 1$. For $x \notin C$, we set $c_x(1) = c_x(0) = 0$. For all variables in C but the last one read by $\mathbb{A}$ we let $x(\sigma^{\mathbb{A}}) = 1$. All the other variables are set to 0.

The algorithm spends $r(|C| - 1) + 1$ to prove that $f(\sigma^{\mathbb{A}}) = 0$. In fact, since C is a minterm, $\mathbb{A}$ cannot conclude that f evaluates to 0 before reading all variables in C. On the other hand, the cheapest proof costs exactly 1 since there is a maxterm of f whose intersection with C is exactly the last variable read by $\mathbb{A}$. Thus, $\gamma_r(f) \geq r(k(f)-1)+1$. By an analogous argument, one can prove that $\gamma_r(f) \geq r(l(f) - 1) + 1$ yielding the desired result. □

Combining this result with the above upper bound gives the exact value of $\gamma_r(f)$ for monotone Boolean functions.

Theorem 2 *For every $r \geq 1$ and for every monotone Boolean function f, we have*

$$\gamma_r(f) = r \cdot \max\{k(f), l(f)\} - r + 1 \,.$$

4 Game Trees

A game tree T is a tree, rooted at a node r, where every internal node has either a MIN or a MAX label and the parent of every MIN (MAX) node is a MAX (MIN) node. Let V be the set of leaves of T. Every leaf of V is associated with a real number, its value. The value of a MIN (MAX) node is the minimum (maximum) of the values of its children. The function computed by T maps the values of the leaves to the value of the root. We shall identify T with the function it computes. Thus, if f is the function computed by the game tree T, we shall also write T for f and T_Y for f_Y.

By a *minterm* (*maxterm*) of a game tree we shall understand a minimal set of leaves whose values allow to state a lower (upper) bound on the value of the game tree. More precisely, a minterm (maxterm) for a game tree T rooted at r is a minimal set C of leaves of T such that if $x(\sigma) \geq \ell$ ($x(\sigma) \leq \ell$), for each $x \in C$ then $T(\sigma) \geq \ell$ ($T(\sigma) \leq \ell$) regardless of the values of the leaves $y \notin C$. We shall use the more general term *certificate* to either refer to a minterm or to a maxterm. We shall use $\mathcal{F}^L_T$ and $\mathcal{F}^U_T$ to denote the family of all minterms and the family of all maxterms of T, respectively.

As an example, for the game tree function

$$T = \max\{\min\{x_1, x_2, x_3\}, \min\{\max\{x_4, x_5\}, x_6\}\} \,,$$

we have $\mathcal{F}^U_T = \{\{x_1, x_6\}, \{x_2, x_6\}, \{x_3, x_6\}, \{x_1, x_4, x_5\}, \{x_2, x_4, x_5\}, \{x_3, x_4, x_5\}\}$ and $\mathcal{F}^L_T = \{\{x_1, x_2, x_3\}, \{x_4, x_6\}, \{x_5, x_6\}\}$.

These families can be obtained by the following recursive procedure:

• if r is a leaf then $\mathcal{F}^L_T = \mathcal{F}^U_T = \{\{r\}\}$,

• otherwise, let $T_1, \ldots, T_p$ be the subtrees rooted at the children of r. If r is a MIN node then $\mathcal{F}^U_T = \bigcup_{i=1}^p \mathcal{F}^U_{T_i}$ and[1] $\mathcal{F}^L_T = \prod_{i=1}^p \mathcal{F}^L_{T_i}$. If r is a MAX node, $\mathcal{F}^L_T = \bigcup_{i=1}^p \mathcal{F}^L_{T_i}$ and $\mathcal{F}^U_T = \prod_{i=1}^p \mathcal{F}^U_{T_i}$. For the ease of notation, when the function/tree T is clear from the context we shall simply write $\mathcal{F}^U$ and $\mathcal{F}^L$ for $\mathcal{F}^U_T$ and $\mathcal{F}^L_T$.

By the above recursion, it can be verified that every maxterm and every minterm have a unique variable in common. Notice that the number of certificates of a game tree can in general be exponential in the number of leaves. Therefore, an efficient algorithm will never explicitly construct the whole families of certificates.

We shall use $k(T)$ and $l(T)$ to denote the largest minterm and maxterm of T, respectively. These quantities play a critical role in the following lower bound on the extremal competitiveness of every algorithm that evaluates a game tree (in the value dependent cost model).

Theorem 3 *Let T be a game tree. If each certificate of T has size at least 2 then $\gamma_r(T) \geq r \cdot \max\{k(T), l(T)\} - r + 1$.*

Proof. Consider an algorithm $\mathbb{A}$ for evaluating T. Mimicking the proof of Theorem 1, fix a largest minterm C and consider the assignment $\sigma^{\mathbb{A}}$ which sets to 1 all variables in C but the last one queried by $\mathbb{A}$ and sets to 0 all other variables. Also, for $x \in C$, we set $c_x(1) = r$, and $c_x(0) = 1$. For $x \notin C$, we set $c_x(1) = c_x(0) = 0$.

Let x denote the last variable in C queried by $\mathbb{A}$ (the existence of such a variable follows from equation (4) and the fact that every maxterm intersects C). Let $X = (V \setminus C) \cup \{x\}$.

Claim. The set X contains a minterm.

Proof of claim. It is enough to show that X intersects every maxterm C'. If $C' \cap C = \{x\}$, the statement holds. Otherwise, since C' intersects C in precisely one variable and $|C'| \geq 2$ by assumption, C' must contain a variable from X, which again implies the desired conclusion.

Consider now the set $C^U \cup C^L$ where C^U is a maxterm of T such that $C^U \cap C = \{x\}$, and C^L is a minterm contained in X. Then, $C^U \cup C^L$ is a proof for $T(\sigma^{\mathbb{A}}) = 0$; moreover, the cost of $C^U \cup C^L$ is 1. Since every proof must contain a maxterm, and every maxterm intersects C, we conclude that the cheapest proof for $\sigma^{\mathbb{A}}$ costs exactly 1.

On the other hand, since C is a minterm, $\mathbb{A}$ cannot conclude that $T(\sigma) < 1$ before reading all variables in C. Thus, $\gamma_r(T) \geq r(k(T) - 1) + 1$. By an analogous argument, one can prove that $\gamma_r(T) \geq r(l(T) - 1) + 1$ yielding the desired result. □

Upper Bound. We shall now employ the Linear Programming Approach for obtaining an upper bound on the (r-)extremal competitive ratio for game trees that matches the above lower bound.

We need to introduce some more notation. Let T be a game tree on V. Consider a run of an algorithm $\mathbb{A}$ for evaluating T. Let $Y \subseteq V$ denote the set of variables read by $\mathbb{A}$ at some point during its run and let σ_Y be the assignment of real numbers to the leaves in Y corresponding to the variables read. Suppose that the restriction T_Y

[1] For all families of sets $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$ we define $\prod_i \mathcal{F}_i$ as follows: $\prod_{i=1}^k \mathcal{F}_i = \{X | X = \bigcup_{i=1}^k X_i, X_i \in \mathcal{F}_i, X_i \neq \emptyset\}$

of T according to the assignment given by σ_Y is non-constant. Let C be a minterm (maxterm) of T. We define the (current) *value* of C as the minimum (maximum) value in σ_Y of the leaves in $Y \cap C$. We say that a minterm (maxterm) is *completely evaluated* if it is entirely contained in Y. Let LB denote the maximum value of a completely evaluated minterm (or $-\infty$, if no minterm has been completely evaluated), and let UB denote the minimum value of a completely evaluated maxterm (or ∞, if no maxterm has been completely evaluated). Note that if UB (LB) is finite, then every minterm (maxterm) has a well-defined value.

In order to study the structure of a proof for T, it is useful to express the function computed by T in terms of its certificates as follows. For every $\sigma \in \mathbb{R}^V$, we have:

$$T(\sigma) = \max_{C^L \in \mathcal{F}^L} \min\{x(\sigma) : x \in C^L\} = \min_{C^U \in \mathcal{F}^U} \max\{x(\sigma) : x \in C^U\}\,. \tag{4}$$

It follows that LB (UB) is the lower (upper) bound on the value of $T(\sigma)$ for any assignment that extends σ_Y. Moreover, since T_Y is assumed to be non-constant, we have $LB < UB$.

As we show next, T can evaluate to any value between the two bounds.

Lemma 2 *Let $y \in \mathbb{R}$ such that $LB \leq y \leq UB$. Then:*

(i) *The set $V^{\leq y} := (V \backslash Y) \cup \{x \in Y : x(\sigma_Y) \leq y\}$ contains a maxterm of T.*
The set $V^{\geq y} := (V \backslash Y) \cup \{x \in Y : x(\sigma_Y) \geq y\}$ contains a minterm of T.
(ii) *There is an assignment $\sigma \in \mathbb{R}^V$ that extends the current partial assignment σ_Y such that $T(\sigma) = y$.*

Proof. For part (i), we shall only prove that the set $V^{\leq y}$ contains a maxterm of T. The other statement can be proved similarly.

Since the maxterms of T are precisely the minimal hitting sets of the minterms of T, it suffices to show that the set $V^{\leq y}$ intersects every minterm of T. Suppose, for the sake of contradiction, that there is a minterm C^L of T disjoint from $V^{\leq y}$. Then, C^L is contained in the set $V \backslash V^{\leq y} = \{x \in Y : x(\sigma_Y) > y\}$. In particular, this implies that C^L has been completely evaluated. Moreover, the value of C^L is $\min\{x(\sigma_Y) : x \in C^L\} > y \geq LB$, contradicting the definition of LB.

To see (ii), let C^L (C^U) be a minterm (maxterm) of T contained in the set $V^{\geq y}$ ($V^{\leq y}$), and let σ be the assignment of values to the leaves of T that extends the current assignment σ_Y and satisfies $x(\sigma) = y$ for all $x \in V \backslash Y$. Then, the value of both C^L and C^U with respect to σ is y. Therefore, $T(\sigma) = y$. □

We say that a minterm (maxterm) C is *active* if for each leaf $x \in C \cap Y$ we have $x(\sigma_Y) > LB$ ($x(\sigma_Y) < UB$). In words, a minterm (maxterm) C is active if the evaluation of its unevaluated leaves can still lead to an improvement in the lower bound LB (upper bound UB), i.e., can provide information on the value of the game tree. Note that if all leaves of a certificate C have already been read, then C is non-active.

The following lemma characterizes the proofs of the restricted game tree T_Y. By saying that a set of variables P is a proof of (a value) y for (a function) f we mean here that P is a proof for f w.r.t. an assignment σ s.t. $f(\sigma) = y$.

Lemma 3 (Proofs of a restricted game tree) *Let $P \subseteq V \backslash Y$. Then:*

(1) **[minterm proofs]** *Suppose that UB is finite. P is a proof of UB for T_Y if and only if there is an active minterm C^L of value at least UB such that $C^L \backslash Y \subseteq P$.*
(2) **[maxterm proofs]** *Suppose that LB is finite. P is a proof of LB for T_Y if and only if there is an active maxterm C^U of value at most LB such that $C^U \backslash Y \subseteq P$.*
(3) **[combined proofs]** *Let $y \in (LB, UB)$. P is a proof of y for T_Y if and only if there is an active minterm C^L of value y^L and an active maxterm C^U of value y^U such that $LB < y^U \leq y \leq y^L < UB$ and such that $(C^L \backslash Y) \cup (C^U \backslash Y) \subseteq P$.*
If $UB = \infty$ then $y^L = \infty$ is allowed. Similarly, if $LB = -\infty$ then $y^U = -\infty$ is allowed.

Proof. We shall prove (1) and (3). Item (2) can be proved similarly as (1).

(1): First, suppose that P is a proof of UB for T_Y w.r.t. an assignment σ_P of values to the variables in P. Let $LB \leq y' < UB$, and consider the assignment σ' that agrees with σ_P on the variables in P and assigns y' to the variables in $V \backslash (Y \cup P)$. By the assumption on P, the restricted game tree T_Y evaluates to UB on σ', or, equivalently, T evaluates to UB on the assignment σ composed of σ_Y and σ'. By equation (4), there is a minterm C^L such that $\min\{x(\sigma) : x \in C^L\} = UB$. In particular, C^L is an active minterm of value UB, with $C^L \backslash Y \subseteq P$ (by the choice of y).

The other direction is considerably simpler. If there is an active minterm C^L of value at least UB such that $C^L \backslash Y \subseteq P$ then assigning UB to each variable in $P \supseteq C^L \backslash Y$ makes C^L evaluate to UB, which in turn raises the lower bound to UB, thus forcing the game tree to evaluate to UB.

(3): Let $LB < y < UB$ and suppose that P is a proof of y for T_Y w.r.t. an assignment σ_P of values to the variables in P. Similarly as above, let $LB \leq y' < y$, and consider the assignment σ' that agrees with σ_P on the variables in P and assigns y' to the variables in $V \backslash (Y \cup P)$. By the assumption on P, T_Y evaluates to y on σ', or, equivalently, T evaluates to y on the assignment σ composed of σ_Y and σ'. By equation (4), there is a minterm C^L such that $\min\{x(\sigma) : x \in C^L\} = y$. Then, C^L is an active minterm of value $y^L \geq y$, with $C^L \backslash Y \subseteq P$. Similarly, there is an active maxterm C^U of value $y^U \leq y$, with $C^U \backslash Y \subseteq P$.

For the converse direction, suppose that there is an active minterm C^L of value y^L and an active maxterm C^U of value y^U such that $LB < y^U \leq y \leq y^L < UB$ and such that $(C^L \cup C^U) \backslash Y \subseteq P$. Since C^L and C^U are active, the sets $C^L \backslash Y$ and $C^U \backslash Y$ are nonempty. Consider the assignment σ_P that assigns y to the variables in $P \supseteq (C^L \backslash Y) \cup (C^U \backslash Y)$. This makes C^L evaluate to y, which implies $T(\sigma) \geq y$ for every assignment σ that simultaneously extends σ_Y and σ_P. At the same time, C^U gets evaluated to y, which implies $T(\sigma) \leq y$. Therefore, the value of the restricted game tree $T_{Y \cup P}$ is constantly equal to y, proving that P is a proof of y for T_Y. □

For $z \in \mathbb{R}^V$, we denote $\|z\|_1 = \sum_{x \in V} |z(x)|$.

Lemma 4 *There is a solution s_Y to the $\mathbf{LP_{T_Y}}$ such that $\|s_Y\|_1 \leq \max\{k(T), l(T)\}$. Moreover, such a solution can be found in polynomial time.*

Proof. We split the proof into two cases, according to the value of UB.

Case 1. *No maxterm has been completely evaluated yet ($UB = \infty$).* In particular, all the maxterms are active, and there are no minterm proofs. Let H^U be a minimal hitting set of the family $\{C^U \backslash Y : C^U$ is an active maxterm$\}$, and let s_Y be the characteristic vector of H^U. We claim that this s_Y is a solution with the desired properties. Indeed, since there are no minterm proofs, all the minimal proofs contain a member of the family $\{C^U \backslash Y : C^U$ is an active maxterm$\}$, which implies that s_Y is a feasible solution to the linear program $\mathbf{LP_{T_Y}}$. Furthermore, it was shown in [3] that every minimal hitting set of the family $\{C^U \backslash Y : C^U$ is an active maxterm$\}$ is contained in a minterm of T. Hence $\|s_Y\|_1 = |H^U| \leq k(T)$.

Case 2. *There is a completely evaluated maxterm ($UB < \infty$).* In this case, let $\mathcal{P}_1$ denote the family of all minimal minterm proofs, and let $\mathcal{P}_2$ denote the family of all $(C^U \backslash Y)$-parts of the other (i.e., maxterm and combined) minimal proofs. By Lemma 2, the families $\mathcal{P}_1$ and $\mathcal{P}_2$ are nonempty.

Claim.

(i) $\max\{|P| : P \in \mathcal{P}_1 \cup \mathcal{P}_2\} \leq \max\{k(T), l(T)\}$.

(ii) Every member of $\mathcal{P}_1$ intersects every member of $\mathcal{P}_2$.

Proof of Claim. Part (i) follows from the observations that every element of $\mathcal{P}_1$ is contained in a minterm, and every element of $\mathcal{P}_2$ is contained in a maxterm.

We prove (ii) by contradiction. Suppose that there is a minimal minterm proof $C_0^L \backslash Y$ and a minimal non-minterm proof $(C_1^L \backslash Y) \cup (C_1^U \backslash Y)$, with $(C_1^U \backslash Y)$ nonempty and $(C_1^L \backslash Y)$ possibly empty, such that $(C_0^L \backslash Y) \cap (C_1^U \backslash Y) = \emptyset$. Let y be the value of C_1^U. Then, by the above characterization of minimal proofs, $y < UB$.

Consider the partial assignment σ that extends the current assignment σ_Y by setting all the leaves of $C_0^L \backslash Y$ to UB, and all the leaves of $C_1^U \backslash Y$ to y. Then, the minterm C_0^L proves that the value of T at σ is at least UB, while the maxterm C_1^U proves that the value of T at σ is at most $y < UB$. This is a contradiction, and the proof of the claim is complete.

We recall the following result implicitly contained in [6].

Theorem 4 ([6]) *Let $\mathcal{A}_1, \mathcal{A}_2$ be two nonempty set families over V such that $X \cap Y \neq \emptyset$, for each $X \in \mathcal{A}_1$ and each $Y \in \mathcal{A}_2$. Then, there is a feasible solution s to the linear program* $\left\{ \textbf{Minimize } \|s\|_1 \textit{ s.t. } \sum_{x \in A} s(x) \geq 1 \ \forall A \in \mathcal{A}_1 \cup \mathcal{A}_2, \textit{ and } s(x) \geq 0 \ \forall x \in V \right\}$ *such that* $\|s\|_1 \leq \max\{|A| : A \in \mathcal{A}_1 \cup \mathcal{A}_2\}$.

In conjunction with the above claim, this theorem implies that there is a feasible solution s_Y to the linear program

$$\left\{ \textbf{Minimize } \|s_Y\|_1 \text{ s.t. } \sum_{x \in P} s_Y(x) \geq 1 \ \forall P \in \mathcal{P}_1 \cup \mathcal{P}_2, \text{ and } s_Y(x) \geq 0 \ \forall x \in V \backslash Y \right\}$$

such that $\|s_Y\|_1 \leq \max\{k(T), l(T)\}$.

It remains to show that s_Y is a feasible solution to $\mathbf{LP_{T_Y}}$. But this follows from the fact that every minimal proof of T_Y contains a member of $\mathcal{P}_1 \cup \mathcal{P}_2$.

This concludes Case 2 and completes the proof of the existence of the desired solution s_Y.

We now present a polynomial-time algorithm that computes the solution s_Y described in Lemma 4.

Since game trees are monotone functions, the value of UB can be computed in linear time by a single bottom-up traversal of the tree, in the same manner as the game tree T would be evaluated—with the exception that infinite values are allowed. More precisely, we have $UB = T(\sigma^U) \in \mathbb{R} \cup \{\infty\}$, where σ^U is the assignment that extends σ_Y by assigning to every unevaluated leaf the value ∞.

If we are in **Case 1**, then we compute the set H^U in linear time by a single bottom-up traversal of the tree, as follows. We associate to each node x of T a set $H(x) \subseteq V$.

If x is a leaf, then let $H(x) = \{x\}\backslash Y$. For an internal node x, let $C(x)$ denote the set of children of x.

If x is a MAX node, then let

$$H(x) = \begin{cases} \emptyset, & \text{if } H(y) = \emptyset \text{ for all } y \in C(x); \\ H(y), \text{where } y \in C(x) \text{ such that } H(y) \neq \emptyset & \text{otherwise.} \end{cases}$$

If x is a MIN node, then let

$$H(x) = \begin{cases} \emptyset, & \text{if } \exists y \in C(x) \text{ such that } H(y) = \emptyset; \\ \cup\{H(y) : y \in C(x)\}, & \text{otherwise.} \end{cases}$$

For every node x of the game tree, let $\mathcal{F}_x^U$ denote the set of maxterms of the game tree defined by the subtree of T rooted at x. Moreover, let $\mathcal{F}_Y^U(x)$ denote the set of all inclusion-wise minimal sets in the set $\{C^U\backslash Y : C^U \in \mathcal{F}_x^U\}$. It can be proved by induction on the height of the subtree rooted at x that for each node x of T,

$$H(x) = \begin{cases} \text{a minimal hitting set of } \mathcal{F}_Y^U(x), & \text{if } \mathcal{F}_Y^U(x) \neq \{\emptyset\}; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Clearly, the desired set H^U is then given by $H(r)$ where r is the root of T .

If we are in **Case 2**, then we will show how to compute in polynomial time an optimal solution s_Y to the linear program

$$\left\{ \textbf{Minimize } \|s_Y\|_1 \text{ s.t. } \sum_{x \in P} s_Y(x) \geq 1 \ \ \forall P \in \mathcal{P}_1 \cup \mathcal{P}_2, \text{ and } s_Y(x) \geq 0 \ \ \forall x \in V\backslash Y \right\} .$$

As shown above, such a solution will be feasible for $\mathbf{LP_{T_Y}}$ and will satisfy $\|s_Y\|_1 \leq \max\{k(T), l(T)\}$.

There could be exponentially many constraints in this linear program. Nevertheless, we will show that the separation problem can be solved in polynomial time. Using the ellipsoid method, an optimal solution to the above linear program can be found with only polynomially many calls to the separation oracle [11].

Let s_Y be a rational vector in $\mathbb{R}^{V\backslash Y}$. We may assume that $s_Y(x) \geq 0$ for every $x \in V\backslash Y$, for otherwise we have a separating hyperplane.

To verify whether $\sum_{x \in P} s_Y(x) \geq 1$ for every $P \in \mathcal{P}_1 \cup \mathcal{P}_2$, we proceed as follows.

First, we compute the value of $\alpha := \min\{\sum_{x \in P} s_Y(x) : P \in \mathcal{P}_1\}$, together with a $P \in \mathcal{P}_1$ such that $\sum_{x \in P} s_Y(x) = \alpha$. If $\alpha < 1$, we output the characteristic vector of P as a separating hyperplane. Otherwise, we compute the value of $\beta := \min\{\sum_{x \in P} s_Y(x) : P \in \mathcal{P}_2\}$, and proceed similarly.

The following observations show that these computations can be carried out in polynomial time.

Let $\mathcal{F}^L_{\geq UB}$ denote the set of minterms of T of value at least UB. Similarly, let $\mathcal{F}^U_{<UB}$ denote the set of maxterms of T of value less than UB. Then:

- $\alpha = \min\{\sum_{x \in C^L \setminus Y} s_Y(x) : C^L \in \mathcal{F}^L_{\geq UB}\}$;
  $\beta = \min\{\sum_{x \in C^U \setminus Y} s_Y(x) : C^U \in \mathcal{F}^U_{<UB}\}$.
  This follows directly from the definitions of α, β and the two families $\mathcal{P}_1$, $\mathcal{P}_2$.
- *For every $w : V \to \mathbb{R} \cup \{\infty\}$, a minterm $C \in \mathcal{F}^L$ minimizing $w(C) := \sum_{x \in C} w(x)$ over all minterms can be computed in polynomial time. Similarly, a maxterm $C \in \mathcal{F}^U$ minimizing $w(C)$ over all maxterms can be computed in polynomial time.*
  Using the recursive structure of the minterms (maxterms), it is easy to see that such a minterm (maxterm) can be computed in linear time by a single bottom-up traversal of the tree T.
- *A minterm $C^L \in \mathcal{F}^L_{\geq UB}$ of T that minimizes the quantity $\sum_{x \in C^L \setminus Y} s_Y(x)$ over all $C \in \mathcal{F}^L_{\geq UB}$ can be computed in polynomial time.*
  This follows from the previous observation, by defining $w : V \to \mathbb{R} \cup \{\infty\}$ as follows:
  $$w(x) = \begin{cases} s_Y(x), & \text{if } x \in V \setminus Y \\ 0, & \text{if } x \in Y \text{ and } \sigma_Y(x) \geq UB; \\ \infty, & \text{otherwise.} \end{cases}$$
  Let C^L be a minterm minimizing $w(C^L)$. Since the set $\mathcal{F}^L_{\geq UB}$ is nonempty, and $w(C)$ is finite precisely for $C \in \mathcal{F}^L_{\geq UB}$, the minterm C^L must belong to $\mathcal{F}^L_{\geq UB}$. But clearly, for all $C \in \mathcal{F}^L_{\geq UB}$, we have $w(C) = \sum_{x \in C} w(x) = \sum_{x \in C \setminus Y} s_Y(x)$.
- *A maxterm $C^U \in \mathcal{F}^U_{<UB}$ of T that minimizes the quantity $\sum_{x \in C^U \setminus Y} s_Y(x)$ over all $C \in \mathcal{F}^U_{<UB}$ can be computed in polynomial time.*
  This statement can be proved in a similar manner.

This concludes the description of the polynomial-time separation procedure, and with it the proof of the lemma. □

The following result follows from Lemmas 1 and 4 and Theorem 3.

Corollary 1 *Let T be a game tree, and let $r \geq 1$. If each certificate of T has size at least 2 then $\gamma_r(T) = r \cdot \max\{k(T), l(T)\} - r + 1$. Moreover, there is a polynomial time algorithm for evaluating game trees each certificate of which has size at least 2 with optimal r-extremal competitiveness, for each $r \geq 1$.*

In the case when not all the certificates of T are of size at least 2, it is possible to improve the upper bound. We let $p(T)$ ($q(T)$) denote the number of minterms (maxterms) of T of size 1.

Theorem 5 *Let T be a game tree with at least two leaves. If $p(T) \geq 1$, then $\gamma_r(T) = r \cdot \max\{k(T), l(T) - p(T)\} - r + 1$. Similarly, if $q(T) \geq 1$, then $\gamma_r(T) = r \cdot \max\{k(T) - q(T), l(T)\} - r + 1$.*

The following two lemmas provide the matching bounds.

Lemma 5 *Let T be a game tree with at least two leaves. If $p(T) \geq 1$ (and then $q(T) = 0$), then $\gamma_r(T) \geq r \cdot \max\{k(T), l(T) - p(T)\} - r + 1$. Similarly, if $q(T) \geq 1$ (and then $p(T) = 0$), then $\gamma_r(T) \geq r \cdot \max\{k(T) - q(T), l(T)\} - r + 1$.*

Proof. We shall only prove that if $p(T) \geq 1$, then $\gamma_r(T) \geq r \cdot \max\{k(T), l(T) - p(T)\} - r + 1$. The proof of the other inequality is similar.

For simplicity, let us write $p = p(T)$ and $l = l(T)$. Let $p \geq 1$. First, note that the inequality $\gamma_r(T) \geq r(k(T) - 1) + 1$ is proved exactly the same as in the proof of Theorem 1.

It remains to prove $\gamma_r(T) \geq r(l - p - 1) + 1$. Note that, since every maxterm intersects every minterm, we have $l \geq p$. If $l = p$, the inequality holds true. Assume now that $l > p$. Let $X \subseteq V$ denote the set of leaves of T that correspond to the minterms of size 1, and let C be a largest maxterm of T. Then X is a proper subset of C.

Consider an algorithm $\mathbb{A}$ for evaluating T. We construct an assignment $\sigma^{\mathbb{A}}$ which is 'bad' for $\mathbb{A}$. For $x \in C \backslash X$, we set $c_1(x) = 1$, and $c_0(x) = r$. For $x \notin C \backslash X$, we set $c_1(x) = c_0(x) = 0$. For all variables in $C \backslash X$ but the last one read by $\mathbb{A}$ we set $\sigma^{\mathbb{A}}(x) = 0$. All the other variables are set to 1. The algorithm spends $r(|C| - p - 1) + 1$ to prove that $T(\sigma^{\mathbb{A}}) = 1$. In fact, since C is a maxterm, $\mathbb{A}$ cannot conclude that T evaluates to 1 before reading all variables in C. On the other hand, it is easy to see that the cheapest proof costs exactly 1. Thus, $\gamma_r(T) \geq r(l - p - 1) + 1$ and the proof is complete. □

Lemma 6 *Let T be a game tree with at least two leaves. If $p(T) \geq 1$, then $\gamma_r(T) \leq r \cdot \max\{k(T), l(T) - p(T)\} - r + 1$. Similarly, if $q(T) \geq 1$, then $\gamma_r(T) \leq r \cdot \max\{k(T) - q(T), l(T)\} - r + 1$.*

Proof. We shall show the first statement only: if $p := p(T) \geq 1$, then $\gamma_r(T) \leq r \cdot \max\{k(T), l(T) - p\} - r + 1$. The other statement can be proved similarly.

First, consider the case when T is of the form $T = \max\{x_1, \ldots, x_p\}$. Then, every algorithm must query all the variables in order to evaluate T, and $\gamma_r(T) = 1$ (independently of r).

Otherwise, T can be written in the form $T = \max\{x_1, \ldots, x_p, T'\}$ where T' is a game tree over $V' := V \backslash \{x_1, \ldots, x_p\}$. By Lemma 1, it is sufficient to show that there is an implementation $\mathbb{LP}$ of the $\mathcal{LPA}^*$ such that, for each subset $Y \subset V$ of queried variables, the solution $s_Y(\cdot)$ to the $\mathbf{LP_{T_Y}}$ chosen by the $\mathbb{LP}$ satisfies $\|s_Y\|_1 \leq \max\{k(T), l(T) - p\}$.

For $Y = \emptyset$, we let $\mathbb{LP}$ choose $s_\emptyset$ as the characteristic vector of the singleton $\{x_1\}$. It is easy to see that $s_\emptyset$ is a feasible solution to $\mathbf{LP_T}$ such that $\|s_\emptyset\|_1 = 1$. Then, $\mathbb{LP}$ reads the variable x_1 (setting $Y = \{x_1\}$), and proceeds in the same manner: when

$Y = \{x_1, \ldots, x_i\}$, for $1 \leq i < p$, we let $\mathbb{LP}$ choose $s_Y \in \mathbb{R}^{V \setminus Y}$ as the characteristic vector of the singleton $\{x_{i+1}\}$.

When $Y = \{x_1, \ldots, x_p\}$, the restricted function is of the form

$$T_Y = \max\{C, T'\}\,,$$

where $C = \max_{1 \leq i \leq p}\{x_i(\sigma)\}$. Moreover, C is a lower bound on the function's value.

It is easy to see that, for every $Y' \subseteq V$ such that $\{x_1, \ldots, x_p\} \subseteq Y'$, and every corresponding assignment $\sigma_{Y'}$, any active minterm of T' is an active minterm of T, and vice versa, and any active maxterm of T' is the intersection of an active maxterm of T with the set V', and vice versa. Thus, one can use the same approach as in the proof of Lemma 4 to show that there is a solution $s_{Y'}$ to the $\mathbf{LP_{T_{Y'}}}$ such that

$$\|s_{Y'}\|_1 \leq \max\{k(T'), l(T')\} = \max\{k(T), l(T) - p\}\,.$$

This concludes the proof. (The proof assumed that all costs were positive. The proof could easily be modified to take care of the zero costs as well. The variables with zero costs would be queried before the others, and the solutions s_Y could be chosen so to assure that any variable with positive cost from the set $\{x_1, \ldots, x_p\}$ is queried before any variable with positive cost from the set $V \setminus \{x_1, \ldots, x_p\}$. Then, the same arguments as above would apply.) $\square$

The following theorem summarizes our findings on the (r-)extremal competitiveness for game trees.

Theorem 6 *Let T be a game tree. Then*

$$\gamma(T) = \begin{cases} \max\{k(T), l(T)\}, & \text{if } p(T) = q(T) = 1; \\ \max\{k(T) - q(T), l(T) - p(T)\}, & \text{otherwise.} \end{cases}$$

Furthermore, for each $r \geq 1$, we have $\gamma_r(T) = r \cdot \gamma(T) - r + 1$, and there is a polynomial time algorithm for evaluating game trees with optimal r-extremal competitiveness, for each $r \geq 1$.

5 Concluding Remarks

We believe that the value dependent cost model deserves further investigation, as called by its applications in several situations, particularly in the medical setting. The study of this model with respect to the γ_c competitiveness is a main direction for continued research. Remarkably, already the situation of AND/OR tree functions, whose certificates have a simpler structure than those of the game trees, seems to be challenging. We also remark that the existence of an optimal γ_c-competitive algorithm for game tree function is still an open problem even in the more classical value independent cost model.

Acknowledgment. We are grateful to Mike Paterson for suggesting to us the idea of studying the value dependent cost model.

References

1. E. Boros and T. Ünlüyurt. Diagnosing double regular systems. *Annals of Mathematics and Artificial Intelligence*, 26(1-4):171–191, 1999.
2. M. Charikar, R. Fagin, V. Guruswami, J. M. Kleinberg, P. Raghavan, and A. Sahai. Query strategies for priced information. *Journal of Comp. and Syst. Sc.*, 64:785–819, 2002.
3. F. Cicalese and E. S. Laber. A new strategy for querying priced information. In *Proc. of the 37th Annual ACM Symposium on Theory of Computing*, pages 674–683. 2005.
4. F. Cicalese and E. S. Laber. An optimal algorithm for querying priced information: Monotone Boolean functions and game trees. In *Proc. of ESA 2005*, *LNCS* **3669**, pp. 664–676. 2005.
5. F. Cicalese and E. S. Laber. On the competitive ratio of evaluating priced functions. In *Proc. of SODA'06*, pages 944–953, 2006.
6. F. Cicalese and E. S. Laber. Function evaluation via linear programming in the priced information model. In *Proc. of ICALP 2008*, LNCS **5125**, pp. 173–185. 2008.
7. C. G. Diderich. A bibliography on minimax trees. *SIGACT News*, 24(4):82–89, 1993.
8. S. O. Duffuaa and A. Raouf. An optimal sequence in multicharacteristics inspection. *J. Optim. Theory Appl.*, 67(1):79–86, 1990.
9. D. W. Gillies. *Algorithms to schedule tasks with and/or precedence constraints*. PhD thesis, Champaign, IL, USA, 1993.
10. R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2006.
11. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.
12. R. Heiman and A. Wigderson. Randomized vs. deterministic decision tree complexity for read-once boolean functions. *Computational Complexity*, 1:311–329, 1991.
13. J. M. Hellerstein. Optimization techniques for queries with expensive methods. *ACM Transactions on Database Systems*, 23(2):113–157, 1998.
14. J. Louis Anthony Cox, Y. Qiu, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Ann. Oper. Res.*, 21(1-4):1–30, 1989.
15. Y. Qiu, L. A. Cox Jr., and L. Davis. Guess-and-verify heuristics for reducing uncertainties in expert classification systems. In D. Dubois and M. P. Wellman, editors, *UAI*, pages 252–258. Morgan Kaufmann, 1992.
16. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
17. M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In *Proc. of FOCS 1986*, pp. 29–38. 1986.
18. P. P. Shenoy. Game trees for decision analysis. *Theory and Decision*, 44:149–171, 1998.
19. M. Snir. Lower bounds on probabilistic linear decision trees. *TCS*, 38(1):69–82, 1985.
20. M. Tarsi. Optimal search on some game trees. *Journal of the ACM*, 30(3):389–396, 1983.
21. P. Turney. Types of cost in inductive concept learning. In *Proceedings of Workshop Cost-Sensitive Learning at the 17th International Conference on Machine Learning*, 2000.

Bisher erschienene Reports an der Technischen Fakultät
Stand: 2008-09-05

94-01 Modular Properties of Composable Term Rewriting Systems
(Enno Ohlebusch)

94-02 Analysis and Applications of the Direct Cascade Architecture
(Enno Littmann, Helge Ritter)

94-03 From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction
(Robert Giegerich, Stefan Kurtz)

94-04 Die Verwendung unscharfer Maße zur Korrespondenzanalyse in Stereo Farbbildern
(André Wolfram, Alois Knoll)

94-05 Searching Correspondences in Colour Stereo Images – Recent Results Using the Fuzzy Integral
(André Wolfram, Alois Knoll)

94-06 A Basic Semantics for Computer Arithmetic
(Markus Freericks, A. Fauth, Alois Knoll)

94-07 Reverse Restructuring: Another Method of Solving Algebraic Equations
(Bernd Bütow, Stephan Thesing)

95-01 PaNaMa User Manual V1.3
(Bernd Bütow, Stephan Thesing)

95-02 Computer Based Training-Software: ein interaktiver Sequenzierkurs
(Frank Meier, Garrit Skrock, Robert Giegerich)

95-03 Fundamental Algorithms for a Declarative Pattern Matching System
(Stefan Kurtz)

95-04 On the Equivalence of E-Pattern Languages
(Enno Ohlebusch, Esko Ukkonen)

96-01 Static and Dynamic Filtering Methods for Approximate String Matching
(Robert Giegerich, Frank Hischke, Stefan Kurtz, Enno Ohlebusch)

96-02 Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language
(Alois Knoll, Bernd Hildebrand, Jianwei Zhang)

96-03 Correctness in System Engineering
(Peter Ladkin)

96-04 An Algebraic Approach to General Boolean Constraint Problems
(Hans-Werner Güsgen, Peter Ladkin)

96-05 Future University Computing Resources
(Peter Ladkin)

96-06 Lazy Cache Implements Complete Cache
(Peter Ladkin)

96-07 Formal but Lively Buffers in TLA+
(Peter Ladkin)

96-08 The X-31 and A320 Warsaw Crashes: Whodunnit?
(Peter Ladkin)

96-09 Reasons and Causes
(Peter Ladkin)

96-10 Comments on Confusing Conversation at Cali
(Dafydd Gibbon, Peter Ladkin)

96-11 On Needing Models
(Peter Ladkin)

96-12 Formalism Helps in Describing Accidents
(Peter Ladkin)

96-13 Explaining Failure with Tense Logic
(Peter Ladkin)

96-14 Some Dubious Theses in the Tense Logic of Accidents
(Peter Ladkin)

96-15 A Note on a Note on a Lemma of Ladkin
(Peter Ladkin)

96-16 News and Comment on the AeroPeru B757 Accident
(Peter Ladkin)

97-01 Analysing the Cali Accident With a WB-Graph
(Peter Ladkin)

97-02 Divide-and-Conquer Multiple Sequence Alignment
(Jens Stoye)

97-03 A System for the Content-Based Retrieval of Textual and Non-Textual Documents Based on Natural Language Queries
(Alois Knoll, Ingo Glöckner, Hermann Helbig, Sven Hartrumpf)

97-04	Rose: Generating Sequence Families (Jens Stoye, Dirk Evers, Folker Meyer)
97-05	Fuzzy Quantifiers for Processing Natural Language Queries in Content-Based Multimedia Retrieval Systems (Ingo Glöckner, Alois Knoll)
97-06	DFS – An Axiomatic Approach to Fuzzy Quantification (Ingo Glöckner)
98-01	Kognitive Aspekte bei der Realisierung eines robusten Robotersystems für Konstruktionsaufgaben (Alois Knoll, Bernd Hildebrandt)
98-02	A Declarative Approach to the Development of Dynamic Programming Algorithms, applied to RNA Folding (Robert Giegerich)
98-03	Reducing the Space Requirement of Suffix Trees (Stefan Kurtz)
99-01	Entscheidungskalküle (Axel Saalbach, Christian Lange, Sascha Wendt, Mathias Katzer, Guillaume Dubois, Michael Höhl, Oliver Kuhn, Sven Wachsmuth, Gerhard Sagerer)
99-02	Transforming Conditional Rewrite Systems with Extra Variables into Unconditional Systems (Enno Ohlebusch)
99-03	A Framework for Evaluating Approaches to Fuzzy Quantification (Ingo Glöckner)
99-04	Towards Evaluation of Docking Hypotheses using elastic Matching (Steffen Neumann, Stefan Posch, Gerhard Sagerer)
99-05	A Systematic Approach to Dynamic Programming in Bioinformatics. Part 1 and 2: Sequence Comparison and RNA Folding (Robert Giegerich)
99-06	Autonomie für situierte Robotersysteme – Stand und Entwicklungslinien (Alois Knoll)
2000-01	Advances in DFS Theory (Ingo Glöckner)
2000-02	A Broad Class of DFS Models (Ingo Glöckner)

2000-03	An Axiomatic Theory of Fuzzy Quantifiers in Natural Languages (Ingo Glöckner)
2000-04	Affix Trees (Jens Stoye)
2000-05	Computergestützte Auswertung von Spektren organischer Verbindungen (Annika Büscher, Michaela Hohenner, Sascha Wendt, Markus Wiesecke, Frank Zöllner, Arne Wegener, Frank Bettenworth, Thorsten Twellmann, Jan Kleinlützum, Mathias Katzer, Sven Wachsmuth, Gerhard Sagerer)
2000-06	The Syntax and Semantics of a Language for Describing Complex Patterns in Biological Sequences (Dirk Strothmann, Stefan Kurtz, Stefan Gräf, Gerhard Steger)
2000-07	Systematic Dynamic Programming in Bioinformatics (ISMB 2000 Tutorial Notes) (Dirk J. Evers, Robert Giegerich)
2000-08	Difficulties when Aligning Structure Based RNAs with the Standard Edit Distance Method (Christian Büschking)
2001-01	Standard Models of Fuzzy Quantification (Ingo Glöckner)
2001-02	Causal System Analysis (Peter B. Ladkin)
2001-03	A Rotamer Library for Protein-Protein Docking Using Energy Calculations and Statistics (Kerstin Koch, Frank Zöllner, Gerhard Sagerer)
2001-04	Eine asynchrone Implementierung eines Microprozessors auf einem FPGA (Marco Balke, Thomas Dettbarn, Robert Homann, Sebastian Jaenicke, Tim Köhler, Henning Mersch, Holger Weiss)
2001-05	Hierarchical Termination Revisited (Enno Ohlebusch)
2002-01	Persistent Objects with O2DBI (Jörn Clausen)
2002-02	Simulation von Phasenübergängen in Proteinmonoschichten (Johanna Alichniewicz, Gabriele Holzschneider, Morris Michael, Ulf Schiller, Jan Stallkamp)
2002-03	Lecture Notes on Algebraic Dynamic Programming 2002 (Robert Giegerich)

2002-04	Side chain flexibility for 1:n protein-protein docking (Kerstin Koch, Steffen Neumann, Frank Zöllner, Gerhard Sagerer)
2002-05	ElMaR: A Protein Docking System using Flexibility Information (Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)
2002-06	Calculating Residue Flexibility Information from Statistics and Energy based Prediction (Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)
2002-07	Fundamentals of Fuzzy Quantification: Plausible Models, Constructive Principles, and Efficient Implementation (Ingo Glöckner)
2002-08	Branching of Fuzzy Quantifiers and Multiple Variable Binding: An Extension of DFS Theory (Ingo Glöckner)
2003-01	On the Similarity of Sets of Permutations and its Applications to Genome Comparison (Anne Bergeron, Jens Stoye)
2003-02	SNP and mutation discovery using base-specific cleavage and MALDI-TOF mass spectrometry (Sebastian Böcker)
2003-03	From RNA Folding to Thermodynamic Matching, including Pseudoknots (Robert Giegerich, Jens Reeder)
2003-04	Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt (Sebastian Böcker)
2003-05	Systematic Investigation of Jumping Alignments (Constantin Bannert)
2003-06	Suffix Tree Construction and Storage with Limited Main Memory (Klaus-Bernd Schürmann, Jens Stoye)
2003-07	Sequencing from compomers in thepresence of false negative peaks (Sebastian Böcker)
2003-08	Genalyzer: An Interactive Visualisation Tool for Large-Scale Sequence Matching – Biological Applications and User Manual (Jomuna V. Choudhuri, Chris Schleiermacher)

2004-01 Sequencing From Compomers is NP-hard
(Sebastian Böcker)

2004-02 The Money Changing Problem revisited: Computing the Frobenius number in time
$O(k\,a_1)$
(Sebastian Böcker, Zsuzsanna Lipták)

2004-03 Accelerating the Evaluation of Profile HMMs by Pruning Techniques
(Thomas Plötz, Gernot A. Fink)

2004-04 Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection
(Ferdinando Cicalese, Peter Damaschke, Ugo Vaccaro)

2004-05 Compressed Representation of Sequences and Full-Text Indexes
(Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, Gonzalo Navarro)

2005-01 Overlaps Help: Improved Bounds for Group Testing with Interval Queries
(Ferdinando Cicalese, Peter Damaschke, Libertad Tansini, Sören Werth)

2005-02 Two batch Fault-tolerant search with error cost constraints: An application to learning
(Ferdinando Cicalese)

2005-03 Searching for the Shortest Common Supersequence
(Sergio A. de Carvalho Jr., Sven Rahmann)

2005-04 Counting Suffix Arrays and Strings
(Klaus-Bernd Schürmann, Jens Stoye)

2005-05 Alignment of Tandem Repeats with Excision, Duplication, Substitution and Indels (EDSI)
(Michael Sammeth, Jens Stoye)

2005-06 Statistics of Cleavage Fragments in Random Weighted Strings
(Hans-Michael Kaltenbach, Henner Sudek, Sebastian Böcker, Sven Rahmann)

2006-01 Decomposing metabolomic isotope patterns
(Sebastian Böcker, Zsuzsanna Lipták, Anton Pervukhin)

2006-02 On Common Intervals with Errors
(Cedric Chauve, Yoan Diekmann, Steffen Heber, Julia Mixtacki, Sven Rahmann, Jens Stoye)

2007-01 Identifying metabolites with integer decomposition techniques, using only their mass spectrometric isotope patterns
(Sebastian Böcker, Matthias C. Letzel, Zsuzsanna Lipták, Anton Pervukhin)

2007-02 2-Stage Fault Tolerant Interval Group Testing
(Ferdinando Cicalese, José Augusto Amgarten Quitzau)

2008-01 Online Abelian Pattern Matching
(Tahir Ejaz, Sven Rahmann, Jens Stoye)

2008-02 A Space Efficient Representation for Sparse de Bruijn Subgraphs
(José Augusto Amgarten Quitzau, Jens Stoye)