On the competitive ratio of evaluating priced functions

Ferdinando Cicalese
AG Genominformatik, Technische Fakultaet, Bielefeld Universität, Bielefeld, Germany
nando@cebitec.uni-bielefeld.de

Coauthors: Eduardo Sany Laber (Dept. Computer Science, PUC-Rio, Rio de Janeiro, Brasil)

We study the fundamental problem of evaluating a function by sequentially selecting a subset of variables whose values uniquely identify the function's value. This basic problem arises in several domains of computer science, e.g., Automatic diagnosis, AI, applied Game Theory, Data Base Query optimization, just to mention a few.

A function f over a set of variables V must be computed and, for many inputs of the domain, not all the variables need to be read in order to determine the value of f on those inputs. A deterministic algorithm for this problem adaptively reads the value of the variables of f until the values read so far uniquely determine the value of f. Classically, each read operation is assumed to incur a unit-cost and the number of variables read (for the worst case input setting) is the measure used to analyze the efficiency of the algorithms. However, it is well known that a large class of functions of interest enjoy the evasiveness property, i.e., in the worst case any deterministic algorithm must read all the variables. Such classes show that the worst case analysis is not generally able to distinguish among the performances of different algorithms for the function evaluation problem. Other metrics that employ probabilistic and competitive analysis have been investigated in the literature (see, e.g., [12,11,2].)

Following Charikar *et. al.* [2], here we address the variant of the function evaluation problem where different variables can incur different reading costs and competitive analysis is employed to measure the performance of the evaluation algorithm.

Competitive Function Evaluation. 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* σ, 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. σ, i.e., $f(\sigma) = f(x_1(\sigma), \ldots, x_n(\sigma))$.

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

An evaluation algorithm $\mathcal{A}$ for f is a decision tree, that 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 σ is the total cost incurred by $\mathcal{A}$ to evaluate f under the assignment σ. Given a cost function $c(\cdot)$, we let $c^f_{\mathcal{A}}(\sigma)$ denote the cost of the algorithm $\mathcal{A}$ for an 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 $\mathcal{A}$.

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.

Our Contributions. We present efficient algorithms for the above models that achieve in general optimal and always very high competitiveness for several classes of functions that have been widely studied in the area. We focus on the class of monotone Boolean function and particularly on the *evasive monotone Boolean functions*, for which any deterministic evaluation algorithm must read all the variables in the worst case. AND/OR trees and threshold tree functions constitute two important representative classes of evasive monotone Boolean functions and they will have a special place in our investigation. The former are tree circuits with both AND and OR gates and with each leaf corresponding to a distinct variable. Threshold trees are a generalization of AND/OR trees where AND and OR gates are replaced by threshold gates. We also consider a non trivial class of monotone Boolean functions that are non-evasive. Our main contributions are as follows (see also [3,4,5]): 1. A simple polynomial time algorithm that achieves optimal competitive ratio γ^f_c for every function f that can be representable by threshold trees. 2. A Linear Programming (based) approach that allows the design of polynomial time algorithms with competitiveness

$\gamma(f)$ (or close to it) for many classes of functions.

In particular, by employing the Linear Programming (based) approach we provide: (a) a $(2\gamma(f) - \sqrt{\gamma(f)})$-competitive polytime algorithm for arbitrary monotone Boolean functions. Here, we assume the existence of an oracle that returns $f(\sigma)$ in polytime, for every possible assignment σ. (b) a $1.618\gamma(f)$-competitive algorithm for the class of *strongly evasive* monotone Boolean functions. A function f is strongly evasive if and only if f and all of its restrictions are evasive functions. (c) a $\gamma(f)$-competitive polytime algorithm for the class of monotone Boolean functions where every variable appears in at most 3 minterms. The interest of this result is that it is optimal in terms of extremal competitiveness for a non-trivial class of monotone Boolean functions that are not included in the class of evasive monotone Boolean functions.

Other Models. Other measures for analyzing the performance of function evaluation algorithms have also been considered in the literature. Both deterministic and randomized algorithms have been investigated. For the former, worst case analysis and probabilistic analysis have been employed to understand the behavior of the proposed algorithms (see [1,11,10,7,8,12,6,9] and references therein quoted.) One can see all these results as different attempts to capture the complexity of evaluating classes of important functions. It is then remarkable that as opposed to the state of the art when the above measures were used, we are able to provide optimal algorithms in terms of the competitive ratio for important classes of functions.

[1] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

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

[3] F. Cicalese and E. S. Laber. A new strategy for querying priced information. In *Proc. of ACM STOC 2005*,pp. 674–683. ACM, 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. Springer, 2005.

[5] F. Cicalese and E. S. Laber. On the competitive ratio of evaluating priced functions. In *Proc. of ACM-SIAM SODA-06*, pp. 944–953, 2006.

[6] R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2006.

[7] R. Heiman and A. Wigderson. Randomized vs. deterministic decision tree complexity for read-once boolean functions. *Computational*

Complexity, 1:311–329, 1991.

[8] T. S. Jayram, R. Kumar, and D. Sivakumar. Two applications of information complexity. In *Proc. of ACM STOC 2003*,pp. 673–682, 2003.

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

[10] M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating game trees. In *Proc. of IEEE-FOCS 1986*, pp. 29–38. IEEE Computer Society, 1986.

[11] M. Snir. Lower bounds on probabilistic linear decision trees. *Theoretical Computer Science*, 38(1):69–82, 1985.

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