Electronic Colloquium on Computational Complexity, Report No. 69 (2002)

A Note of Deterministic Approximate Counting for k-DNF

[Draft]

Luca Trevisan*

November 14, 2002

Abstract

We describe a deterministic algorithm that, for constant k, given a k-DNF or k-CNF formula φ and a parameter ε, runs in time linear in the size of φ and polynomial in $1/\varepsilon$ and returns an estimate of the fraction of satisfying assignments for φ up to an additive error ε.

For k-DNF, a multiplicative approximation is also achievable in time polynomial in $1/\varepsilon$ and linear in the size of φ.

Previous algorithms achieved polynomial (but not linear) dependency on the size of φ and on $1/\varepsilon$; their dependency on k, however, was much better than ours. Unlike previous algorithms, our algorithm is not based on derandomization techniques, and it is quite similar to an algorithm by Hirsch for the related problem of solving k-SAT under the promise that an ε-fraction of the assignments are satisfying. Our analysis is different from (and somewhat simpler than) Hirsch's.

1 Introduction

We consider the following problem: given a k-CNF formula φ and a parameter ε, approximate within an additive factor ε the fraction of satisfying assignments for φ.[1]

The problem is easy to solve using randomization: just generate $O(1/\varepsilon^2)$ assignments at random and then output the fraction of assignments in the sample that satisfies φ. We are interested in deterministic algorithms. We also consider the related problem of finding a satisfying assignment for φ under the promise that an ε fraction of assignments are satisfying. Again, we are interested in deterministic algorithms, and the problem is easy to solve probabilistically, since after picking $O(1/\varepsilon)$ assignments at random it is likely that one of them satisfies the formula.

These problems were first studied by Ajtai and Wigderson [AW89]. Using derandomization techniques (k-wise independence) they give an algorithm for the counting problem running in time $O(n^{k^2}+2^{(\log(1/\varepsilon))^{2^k}})$ and an algorithm for the satisfiability problem running

*luca@cs.berkeley.edu. Computer Science Division, U.C. Berkeley.

[1]Note that an algorithm achieving additive approximation ε for k-CNF immediately implies an algorithm achieving the same additive approximation for k-DNF. Also, achieving multiplicative approximation $(1+\varepsilon)$ for k-DNF reduces to achieving additive approximation $\varepsilon 2^{-k}$, since a satisfiable k-DNF is satisfied by at least a $1/2^k$ fraction of assignments.

ISSN 1433-8092

in time $O(n^{k2^k \log(1/\varepsilon)})$. They also give sub-exponential time algorithm for the counting problem for functions computed by constant-depth circuits (CNF and DNF are depth-2 circuits).

Nisan [Nis91] and Nisan and Wigderson [NW94] construct a pseudorandom generator that fools constant-depth circuits and that has poly-logarithmic seed length. As a consequence, they achieve $n^{(\log n)^{O(1)}}$ time algorithms for the counting and satisfiability problems for constant-depth circuits.

Luby, Velickovic and Wigderson [LVW93] optimize the constructions of Nisan and Wigderson [Nis91, NW94] to the case of depth-2 circuits, thus solving the counting and satisfiability problem in time $n^{O((\log n)^3)}$ for general CNF and DNF. Luby and Velickovic [LV91] show how to reduce arbitrary CNF and DNF to formula in a simplified format, and show that the counting and satisfiability problems can be solved in polynomial time for k-CNF even if $k = O((\log n)^{1/8})$ is more than a constant.

Hirsch [Hir98], in seemingly independent work, shows how to solve the satisfiability problem for kCNF in time $O(Lk(2/\varepsilon)^{B(k)})$, where L is the size of the formula and $B(k)$ is a function for which a closed formula is not given, but that seems to grow exponentially in k.

In this paper, we show how to solve the counting and the satisfiability problem in time $O(L(1/\varepsilon)^{(\ln 4)k2^k})$.

Our algorithm is based on the following simple observation: given a k-CNF φ, then for every fixed c, either we can efficiently find a set of $\leq kc$ variables that hits all the clauses, or we can efficiently find $> c$ clauses over disjoint sets of variables. In the former case, we can try all assignments to those variables, and recurse on each assignment, thus reducing our problem to 2^{kc} problems on $(k-1)$-CNF instances; in the latter case, less than a $(1-1/2^k)^c$ fraction of assignments can satisfy φ, and thus 0 is an approximation to an additive factor $(1-1/2^k)^c$ of the fraction of satisfying assignments for φ. Fixing c to be $2^k \ln 1/\varepsilon$ gives us the main result.

2 The Algorithm

We describe the algorithm only for the case of k-CNF. As discussed in the introduction, an algorithm for k-DNF is an immediate corollary.

The algorithm works as follows: given φ and ε,

- If φ is a 1-CNF, that is, it is just an AND of literals, then we output 0 if there are two inconsistent literals and 2^{-c} where c is the number of distinct literals, otherwise. This procedure is exact and can be implemented in linear time.

- Otherwise, we find a maximal set of disjoint clauses, that is, a set $C_1, \ldots, C_t$ of clauses over disjoint sets of variables and such that every other clause in φ shares at least a variable with some of the C_i. If $(1-1/2^k)^t < \varepsilon$ then we output 0. Otherwise, let V the set of $tk \leq k2^k(\ln 1/\varepsilon)$ variables that occurr in the clauses $C_1, \ldots, C_k$. For every assignment a to the variables V, let φ_a be the formula obtained from φ by substituting the assigment into the variables, then φ_a is a $(k-1)$-CNF formula. We recursively call the algorithm on each of the φ_a with parameter ε, and take the average of the results. Assuming that each recursive call returns an ε additive approximation, the algorithm returns an ε additive approximation.

If we denote by $T(L,k)$ the running time of the algorithm for a k-CNF instance of size L, then we have

$$T(L,1) = O(L)$$

and

$$T(L,k) \leq O(L) + 2^{k(\ln(1/\varepsilon))2^k} T(L,k-1)$$

which solves to $T(L,k) = O(L \cdot 2^{2k(\ln 1/\varepsilon)2^k}) = O(L(1/\varepsilon)^{(\ln 4)k2^k})$.

For the promise problem of finding a satisfying assignment under the promise that an ε fraction of assignments are satisfiable, we essentially use the same recursive algorithm. When we are down to 1-CNF, we find a satisfying assignment or fail if the instance is unsatisfiable. (Indeed, we can stop at 2-CNF.) In the recursive step, we fail if t is such that $(1 - 1/2^k)^t < \varepsilon$. The analysis of the running time is the same, and it is clear that at least one of the recursive branches produces a satisfying assignment.

Hirsch's algorithm is basically the same as the above sketch of the algorithm for the satisfiability promise problem, except that a different greedy strategy is used to pick the variables in V. The analysis is slightly tighter but more difficult.

References

[AW89] Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constand-depth circuits. *Advances in Computing Research - Randomness and Computation*, 5:199–223, 1989. Preliminary version in *Proc. of FOCS'85*.

[Hir98] Edward A. Hirsch. A fast deterministic algorithm for formulas that have many satisfying assignments. *Journal of the IGPL*, 6(1):59–71, 1998.

[LV91] Michael Luby and Boban Velickovic. On deterministic approximation of DNF. In *Proceedings of STOC'91*, pages 430–438, 1991.

[LVW93] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the 2nd ISTCS*, pages 18–24, 1993.

[Nis91] Noam Nisan. Pseudo random bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'