Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 69 (2002)

A Note on Approximate Counting for k-DNF

LUCA TREVISAN*

June 11, 2004

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 $1/\varepsilon$ (but doubly exponential in k) and returns an estimate of the fraction of satisfying assignments for φ up to an additive error ε. This improves over previous polynomial (but super-linear) time algorithms. The algorithm uses a simple recursive procedure and it is not based on derandomization techniques. It is 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.

We also show that every k-CNF is "foolead" by every δ-biased distribution, with $\delta = 1/2^{O(k2^k)}$. A result of Ajtai and Wigderson implied that the same was true with the weaker bound $1/2^{k^{O(k^2)}}$.

1 Introduction

We consider the following problem: given a k-CNF formula φ and a parameter ε, approximate within an *additive* error 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, and the question is whether efficient *deterministic* algorithms exist.

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.

One can consider the approximate counting problem as the problem of *derandomizing two-sided* error algorithms implemented by *depth-two* circuits. The problem of finding a satisfying assignment for φ under the promise that that there is a large number of such assignments can be seen as the problem of derandomizing *one-sided* error algorithms implemented by depth-two circuits.

*luca@cs.berkeley.edu. Computer Science Division, U.C. Berkeley. Supported by NSF grant CCR-9984703, a Sloan Research Fellowship and an Okawa Foundation Grant.

[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

These problems were first studied by Ajtai and Wigderson [AW89]. Using derandomization techniques (specifically, t-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 in time $O(n^{k2^k \log(1/\varepsilon)})$. They also give sub-exponential time algorithm for the counting problem for functions computed by AC^0 circuits.[2]

The algorithm of Ajtai and Wigderson for k-CNF could be improved by using *almost* t-wise independent distributions, for example the small bias distributions of [NN93], instead of distributions that are perfectly t-wise independent. For constant ε, this would improve the running time to roughly $n \cdot (\log n)^{O(1)} \cdot 2^{k^{O(k^2)}}$ for both the approximate counting problem and the satisfiability problem. Almost t-wise independent distributions were introduced after the publication of [AW89].

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 AC^0 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 [LV96] 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. The reduction in [LV96] also gives an improved derandomization of general CNF and DNF that runs in slightly super-polynomial time $n^{O(2^{\sqrt{\log\log n}})}$.

Hirsch [Hir98] shows how to solve the satisfiability problem for k-CNF in time $O(Lk(2/\varepsilon)^{B(k)})$, where $L \leq nk$ 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. Hirsch's algorithm does not use derandomization techniques.

In this paper, we show how to solve the approximate counting problem 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 within an additive error $(1 - 1/2^k)^c$ of the fraction of satisfying assignments for φ. Fixing c to be $2^k \ln 1/\varepsilon$ gives us the main result.

We also revisit the relation between almost t-wise independent distributions and k-CNF. Using the same recursive approach adopted in our algorithm, we show that every k-CNF is well approximated, in a certain technical sense, by a decision tree of depth $t = O(k2^k)$, and it is well known that functions that are well approximated (in the above technical sense) by a decision tree of depth t cannot distinguish the uniform distribution from a distribution that is approximately t-wise independent. This leads to the proof that no k-CNF can distinguish an approximately $O(k2^k)$-wise independent distribution from

[2]An AC^0 circuit is a circuit of constant depth and polynomial size with unbounded fan-in AND and OR gates. A CNF formula is a depth-two AC^0 circuit, and so is a DNF formula.

the uniform distribution.

2 The Recursive 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 of the algorithm for k-CNF.

We use the following simple fact.

Lemma 1 *There is an algorithm that, on input a k-CNF formula φ and a parameter t, runs in time linear in the size of φ and then it returns either a set of t clauses over disjoint sets of variables, or a set S of at most $k(t-1)$ variables such that every clause in the formula contains at least one variable from S.*

PROOF: (Sketch) Consider the k-uniform hypergraph H that has a vertex for every variable and an hyperedge for every clause. It is easy to find a maximal matching in H in linear time, that is, a set S of clauses over disjoint sets of variables and such that every other clause in φ shares some variables with some clause in S. If $|S| \geq t$, then we return t of the clauses in S. Otherwise, we return the set of $\leq k \cdot |S| \leq k(t-1)$ variables that occur in the clauses of S. Such a set of variables clearly "hits" all the clauses of φ. □

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 let t be the smallest integer such that $(1-1/2^k)^{t+1} < \varepsilon$, so that $t \leq 2^k(\ln 1/\varepsilon)$, and we run the algorithm of Lemma 1 on φ with parameter $t+1$.
  - If the algorithm of Lemma 1 finds $t+1$ clauses $C_1, \ldots, C_{t+1}$ over disjoint sets of variables, then it is clear that the probability that φ is satisfied by random assignment is at most ε, and we return the value 0 as our approximation.
  - If the algorithm of Lemma 1 finds a set V of at most $tk \leq k2^k \ln(1/\varepsilon)$ variables that hit all the clauses, then, for every assignment a to the variables V, define φ_a to be the formula obtained from φ by substituting the assigment into the variables. Note that φ_a is a $(k-1)$-CNF formula. We recurse 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 similar to 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 different and somewhat more difficult.

3 Pseudorandomness Against k-CNF Formulae

3.1 Some Technical Preliminaries

We begin this section with a few technical definitions.

We denote by U_n the uniform distribution over $\{0,1\}^n$. If $f : \{0,1\}^n \to \{0,1\}$ is a function and X is a distribution over $\{0,1\}^n$, then we say that X *ε-fools* f if

$$|\,\mathbf{Pr}[f(U_n)=1] - \mathbf{Pr}[f(X)=1]| \le \varepsilon$$

If $\mathcal{F}$ is a collection of functions, then we say that a distribution X ε-fools $\mathcal{F}$ if X ε-fools every function $f \in \mathcal{F}$.

Our goal will be to find a distribution X that ε-fools the class of k-CNF formulae over n variables, and that is uniform over an efficiently constructable support of polynomial size. Then, given a k-CNF formula f, we can approximate $\mathbf{Pr}[f(U_n)=1]$ by computing the close value $\mathbf{Pr}[f(X)=1]$, and we compute the latter by enumerating all the polynomially many elements in the support of X, and applying $f()$ to each of them. We will show that ε-biased distributions, defined below, can be used towards such goal.

We say that a distribution X over $\{0,1\}^n$ is ε-biased [NN93] if for every subset $S \subseteq \{1,\ldots,n\}$ we have

$$\frac{1}{2} - \varepsilon \le \mathbf{Pr}\left[\bigoplus_{i\in S} x_i = 1\right] \le \frac{1}{2} + \varepsilon$$

Equivalently, we can say that a distribution is ε-biased if it ε-fools every linear function. (Where, of course, we mean *linear* over the field $GF(2)$.)

Theorem 2 ([NN93, AGHP92]) *For every ε, and n, there is an ε-biased distribution over $\{0,1\}^n$ that is uniform over a support of size polynomial in n and $1/\varepsilon$. Furthermore, the support can be constructed in time polynomial in n and $1/\varepsilon$.*

We say that a distribution $X = (X_1 \cdots X_n)$ over $\{0,1\}^n$, where each X_i is unbiased, is *k-wise independent* if every k of the random variables $X_1,\ldots,X_n$ are mutually independent. Equivalently, a distribution is k-wise independent if it 0-fools the class of functions that depend only on k or fewer input variables.

We say that X is *ε-close to k-wise independent* if for every function $g : \{0,1\}^n \to \{0,1\}$ that depends on k or fewer inputs we have

$$|\,\mathbf{Pr}[g(U_n)=1] - \mathbf{Pr}[g(X)=1]| \le \varepsilon$$

that is, if X ε-fools the class of functions that depend on at most k inputs.

We say that X is *ε-close to k-wise independent in ℓ_∞ norm* if for every $t \leq k$, for every t indices $i_1, \ldots, i_t$ in $\{1, \ldots, n\}$ and for every t values $a_1, \ldots, a_t \in \{0,1\}$ we have

$$\frac{1}{2^t} - \varepsilon \leq \mathbf{Pr}[X_{i_1} = a_1 \wedge \cdots \wedge X_{i_t} = a_t] \leq \frac{1}{2^t} + \varepsilon$$

that is, if X ε-fools the class of functions that can be expressed as checking that a subset of at most k bits of the input equals a particular sequence of values.

The following connection between the notions that we have described is well known.

Lemma 3 *Let X be an ε-biased distribution over $\{0,1\}^n$. Then, for every k, X is $\varepsilon \cdot 2^{k/2}$-close to k-wise independent, and also 2ε-close to k-wise independent in ℓ_∞ norm.*

3.2 ε-Biased Distribution and k-CNF Fomulae

The main result of this section is the following theorem.

Theorem 4 *There are functions $t(k,\varepsilon) = O(k \cdot 2^k \cdot \log(1/\varepsilon))$ and $\delta(k,\varepsilon) = 1/2^{O(k \cdot 2^k \cdot \log(1/\varepsilon))}$ such that the following happens.*

Let $f : \{0,1\}^n \to \{0,1\}$ be a function defined by a k-CNF formula and let X be a distribution over $\{0,1\}^n$ that is $\delta(k,\varepsilon)$-close to $t(k,\varepsilon)$-wise independent in ℓ_∞ norm. Then X ε-fools f.

The application of Theorem 4 to ε-biased distributions is immediate.

Corollary 5 *There is a function $\delta'(k,\varepsilon) = 1/2^{O(k \cdot 2^k \cdot \log(1/\varepsilon))}$ such that if X is a $\delta'(k,\varepsilon)$-biased distributions, then X ε-fools every function computed by a k-CNF formula.*

The rest of this section is devoted to the proof of Theorem 4.

Let $f : \{0,1\}^n \to \{0,1\}$ be a function defined by a k-CNF formula φ over variables $x_1, \ldots, x_n$, and consider a *decision tree* over the variables $x_1, \ldots, x_n$. Every leaf of the decision tree (indeed, every node of the decision tree) defines a *restriction*, that is, an assignment to a subset of the variables $x_1, \ldots, x_n$. If a leaf v is at distance t from the root, then it defines an assignment to t variables; if we pick a random assignment and then apply the decision tree to it, there is a probability $1/2^t$ that we reach the leaf v. In general, for a vertex v at distance t from the root we define the *probability* of v to be $1/2^t$, and for a set of vertices such that none of them is an ancestor of any other we define the probability of the set as the sum of the probabilities of the individual vertices.

Lemma 6 *Let $f : \{0,1\}^n \to \{0,1\}$ be the function defined by a k-CNF formula φ and $\varepsilon > 0$. Let t be an integer such that $(1 - 1/2^k)^t \leq \varepsilon$. Then there is a decision tree of depth at most tk such that: either (i) all the leaves define restrictions relative to which f is a constant, or (ii) all the leaves, except possibly a set of probability at most ε, define restrictions relative to which φ becomes a $(k-1)$-CNF.*

PROOF: We apply Lemma 1 to φ with parameter t. Then we either find t clauses over disjoint variables or $k(t-1)$ variables that hit all the clauses.

In the former case, consider the decision tree that reads all the $\leq kt$ variables that occur in the t clauses. All but an ε fraction of the leafs of the decision tree correspond to restrictions relative to which φ is zero, and, in particular, is constant.

In the latter case, consider the decision tree that reads all the $\leq kt$ variables returned by the algorithm. All the leafes of the decision tree correspond to restrictions relative to which φ is a $(k-1)$-CNF. □

We now compose the construction.

Lemma 7 *There is a function $t(k,\varepsilon) = O(k2^k \ln(1/\varepsilon))$ such that for every $f : \{0,1\}^n \to \{0,1\}$ defined by a k-CNF formula and for every $\varepsilon > 0$ there is a decision tree of depth at most $t(k,\varepsilon)$ such that all the leaves, except possibly a subset of probability ε, correspond to a restriction relative to which f is a constant.*

PROOF: We prove the theorem by induction. Lemma 6 proves the theorem for $k = 1$ and $t(1,\varepsilon) = \log_2 1/\varepsilon$. For general k, start by constructing the decision tree for f as in Lemma 6. If all but an ε fraction of the leaves of the tree make f become a constant, then we are done. Otherwise, every leaf of the tree defines a restriction relative to which f is a $(k-1)$-CNF, and we can apply the induction hypothesis to get decision trees for each of these $(k-1)$-CNF.

This argument proves the theorem for every function $t()$ that satisfies $t(1,\varepsilon) = \log_2 1/\varepsilon$ and $t(k,\varepsilon) \geq k2^k \ln(1/\varepsilon) + t(k-1,\varepsilon)$. In particular, the theorem is true for $t(k,\varepsilon) = 2k2^k \ln(1/\varepsilon)$. □

To prove Theorem 4 we now only need the following simple last step, which is well known.

Lemma 8 *Let $f : \{0,1\}^n \to \{0,1\}$ be a function and T be a decision tree of depth t such that all but an ε fraction of the leaves of T define a restriction relative to which f is a constant. Let X be a distribution that is δ-close to t-wise independent in ℓ_∞ norm. Then*

$$|\,\mathbf{Pr}[f(U_n) = 1] - \mathbf{Pr}[f(X) = 1]| \leq \varepsilon + \delta \cdot 2^t$$

PROOF: We may assume withouth loss of generality that T has 2^t leaves, all at distance t from the root. (Otherwise, from leaves that are closer to the root, we read additional variables until we reach distance t. This does not change the properties of T assumed in the Lemma.) Let S be the set of leaves of T that define a restriction relative to which f is the constant 1. Then we have

$$\mathbf{Pr}[f(U_n) = 1] \leq \frac{|S|}{2^t} + \varepsilon$$

If we sample an assignment according to X, we see that for each leaf of T there is a probability at least $1/2^t - \delta$ that the assignment is consistent with the leaf. In each of these event, f evaluates to one and, moreover, all these events are disjoint. We deduce

$$\mathbf{Pr}[f(X) = 1] \geq \frac{|Z|}{2^t} - \delta \cdot |Z|$$

and

$$\mathbf{Pr}[f(U_n) = 1] - \mathbf{Pr}[f(X) = 1] \leq \varepsilon + \delta \cdot 2^t$$

Similarly, we can prove

$$\mathbf{Pr}[f(X) = 1] - \mathbf{Pr}[f(U_n) = 1] \leq \varepsilon + \delta \cdot 2^t$$

□

Ajtai and Wigderson [AW89] prove a result that is similar to Lemma 7 but that has a weaker application to ε-biased distributions.

Theorem 9 ([AW89]) *There is a constant $\varepsilon > 0$ and a function $t(k) = k^{O(k^2)})$ such that the following is true. For every k-CNF function f there is a subset V of $t(k)$ variables of f such that, if we pick at random a restriction to the variables in V, there is a probability at least $1 - \varepsilon$ that f is a constant relative to the restriction.*

Our result, like the result of Ajtai and Wigderson, defines a distribution over restrictions such that with high probability the resulting restriction makes f constant. In our proof, the restriction assigns values to $O(k2^k)$ variables, in the result of Ajtai and Wigderson the restriction assigns values to $k^{O(k^2)}$ variables.

4 Perspective

The current body of work on derandomization (see [Kab02] for a survey) strongly suggests that every problem (including search problems and promise problems) that is solvable probabilistically in polynomial time can also be solved deterministically in polynomial time. It is then a natural research program to look for deterministic polynomial time algorithms for all the interesting problems for which only probabilistic polynomial time algorithms are known.

After the discovery of a deterministic polynomial time algorithm for testing primality [AKS02], the most interesting algorithms to derandomize are now the identity test for low-degree polynomials and the approximate counting algorithms based on the Markov-Chain Monte-Carlo approach. Kabanets and Impagliazzo [KI03] show that derandomizing the polynomial identity test algorithm for general arithmetic circuits implies the proof of circuit lower bounds that may be beyond our current proof techniques. It is not clear whether there are similar inherent difficulties in derandomizing approximate counting algorithms such as, say, the Permanent approximation algorithm of [JSV01].

The problem of approximately counting the number of satisfying assignments for a given circuit up to a small *additive* error is clearly precisely the same problem as derandomizing every promise-BPP problem. (In particular, such an approximate counting algorithm would imply that $NEXP \not\subseteq P/poly$.) It seems possible, however, to derandomize at least bounded-depth circuits, and, at the very least, depth-two circuits in polynomial time using current techniques. In this paper we note that a special case of this problem can be solved by a simple divide-and-conquer algorithm, without using derandomization techniques. We also presented some improvement to the application of derandomization techniques to the problem.

References

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Manuscript, 2002.

[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.

[JSV01] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial time approximation algorithm for the permanent of a matrix with non-negative entries. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 712–721, 2001.

[Kab02] Valentine Kabanets. Derandomization: A brief overview. *Bulletin of the European Association for Theoretical Computer Science*, 76:88–103, 2002.

[KI03] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 355–364, 2003.

[LV96] Michael Luby and Boban Velickovic. On deterministic approximation of DNF. *Algorithmica*, 16(4/5):415– 433, 1996.

[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] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 12(4):63–70, 1991.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994. Preliminary version in *Proc. of FOCS'88*.