Approximation Algorithm for Random MAX-kSAT

Yannet Interian

Center for Applied Mathematics.
Cornell University.
Ithaca, NY 14853, USA
interian@cam.cornell.edu

Abstract. An approximation algorithm for random k-SAT formulas (MAX-R-kSAT) is herein discussed. The proposed algorithm is similar to the unit clause with majority rule algorithm studied in [5] for the random 3-SAT problem. The results obtained improve the 9/8-approximation given in [6] for MAX-R-3SAT to a 10/9.5-approximation. The new approximation ratio is achieved by using a better algorithm than the one proposed in [6], along with a new upper bound on the maximum number of clauses that can be satisfied in a random k-SAT formula [1].

1 Introduction

We propose the analysis of a very simple greedy algorithm for approximating the Maximum random k-SAT problem (MAX-R-kSAT). Our work was inspired by the open question given by Fernandez de la Vega *et al* in [6] for the approximation of MAX-R-3SAT. Our algorithm performs better than the one proposed in [6] for approximating MAX-R-3SAT. The results are taken much further using Achlioptas *et al* [1] recent upper bounds on the maximum number of clauses that can be satisfied on a random k-SAT formula.

Our analysis relies on the method of differential equations studied by Wormald in [9]. This method have been used extensively before in the approximation (lower bounds) of the satisfiability threshold (see [2], [7] and [3]). In order to use this method for MAX-kSAT we have to be able to compute not the probability of finding an assignment, as done in [2], [7] and [3], but the expected number of clauses that are going to be falsified by the assignment. This is the main difference between our analysis and the ones done before for random k-SAT.

2 Outline of the results

Given n the number of variables and $H = \{C_0, C_1, C_2, \ldots, C_k\}$ a $k+1$-tuple of non negative integers. Denote Φ_H as the space of all CNF formulas with C_i i-clauses. For instance, the space of k-SAT formulas with m clauses is Φ_H with $H = \{0, 0, \ldots, m\}$. We consider the uniform distribution on the space Φ_H. For a k-CNF formula, m denotes the number of clauses.

For a k-CNF formula F, let $m(F)$ be the maximum number of clauses that can be satisfied in F, and $m_A(F)$ be the number of clauses satisfied by the assignment A. Let $r = m/n$ be the ratio between clauses and variables. Let $r(F)$ be the ratio clause/variables for the k-CNF formula F.

We say that a sequence of random $\mathcal{E}_n$ events occurs *with high probability* (w.h.p) if $\mathbf{Pr}(\mathcal{E}_n)$ goes to 1 as n goes to infinity.

We propose a randomized algorithm which, given a k-CNF formula F, outputs an assignment A, leading to $m_A(F)$ as our approximation to $m(F)$. Thus, we prove that for a fixed k, there exists a function $g(r)$ such that $m_A(F) = g(r)m + o(m)$ w.h.p, i.e

$$\mathbf{Pr}\{m_A(F) = g(r)m + o(m)\}$$

goes to 1 as m goes to infinity.

Then we prove that

$$\mathbf{Pr}\{m(F) \geq \alpha(g(r)m + o(m))\}$$

goes to zero as m goes to infinity. Where α is the approximation constant.

In part of the analysis we use an upper bound for the value of $m(F)$, as given in the results of Achlioptas *et al.* [1]. We prove the following result for random MAX-3SAT.

Theorem 1. *There is a polynomial time algorithm for approximating MAX-R3SAT within the ratio* $\alpha = 10/9.1$.

Remark: That result is easily improved to obtain a 10/9.5 approximation ratio.

3 The Algorithm

The algorithm is divided in two parts, as follows:

Algorithm
begin
if $r(F) > r_k$
    output a random assignment A
Otherwise
    output the assignment A given by
    the **Majority algorithm**
end

That is, if $r(F) > r_k$ for some constant value r_k, the output is a random assignment. If $r(F) \leq r_k$ the algorithm proceeds as follows: While there is an unassigned variable, it selects a random unassigned variable x. If x appears positive in at least half of the clauses, x is assigned to true. Otherwise, it is assigned to false. The formula is simplified after each assignment.

Majority algorithm
begin
While unset variables exist do
    Pick an unset variable x at random
    If x appears positively in at least half of the remaining
    clauses (in which x appears)
        Set x =TRUE
    Otherwise
        Set x =FALSE
    Del&Shrink
end do
output the current assignment
end

Chao and Franco proposed in [5] a unit clause with majority rule algorithm for the study of the satisfiability threshold for random 3-SAT formulas. The majority rule used in [5] attempts to minimize the number of 3-clauses that become 2-clauses, and the unit clause rule attempts to satisfy every unit clause that is produced while running the algorithm. Such strategy is aimed to find satisfying assignments. Since we consider the analysis of the algorithm mostly in the unsat phase, the unit clause rule does not help much. Many unit clauses and empty clauses will appear while running the algorithm. Our aim is to minimize the number of empty clauses at the end of the algorithm. The algorithm in [7] for 3-SAT assigns literals by their degree (number of clauses with that literal) and gives also the same weight to literals appearing in 2 and 3 clauses.

The technique used to analyze this algorithm is similar to the one used in [2] for the analysis of the unit clause with majority rule algorithm. Not using the unit clause rule enabled us to compute

the number of empty (not satisfiable) clauses produced by the final assignment.

The algorithm is also similar to the one proposed in [6]. The algorithm in [6] assigns statically every variable to its best value, i.e x is assigned to true if x appears positive in at least half of the clauses otherwise is assigned to false. Our algorithm does almost the same but dynamically. It assigns one variable at a time and shrinks the formula before considering a new variable to be assigned. Therefore, clauses are not taken into consideration by the algorithm if they are already satisfied.

4 Analysis

4.1 For $r > r_k$

This part of the analysis deals with the random assignment. Note that a random assignment A will satisfy $(1 - \frac{1}{2^k})m + o(m)$ clauses w.h.p.

Lemma 1. *[6] For every ϵ there exits $r_{\epsilon,k}$ such that for $r \leq r_{\epsilon,k}$ and F a random k-SAT formula*

$$\mathbf{Pr}\big\{m(F) \geq (1 - \frac{1}{2^k})m(1+\epsilon)\big\}$$

goes to zero as n goes to infinity.

Proof. Let $q = 1 - \frac{1}{2^k}$

$$\begin{aligned}
\mathbf{Pr}\{m(F) \geq qm(1+\epsilon)\} = &\ \mathbf{Pr}\{|A : m_A(F) \geq qm(1+\epsilon)| > 0\} \\
\leq &\ \mathbf{E}\{|A : m_A(F) \geq qm(1+\epsilon)|\} \\
= &\ 2^n\ \mathbf{Pr}\{\ \mathbf{Bin}(m,q) \geq qm(1+\epsilon)\} \\
\leq &\ 2^n \exp(-\frac{qm\epsilon^2}{2})
\end{aligned}$$

The last inequality follows by Chernoff bound, and goes to zero for $r \geq r_{\epsilon,k} = \frac{2^{k+1}\log 2}{(2^k-1)\epsilon^2}$. □

For instance, in order to obtain a 10/9.1-approximation for MAX-R3SAT, we take $\epsilon = 1/0.91 - 1$ and we obtain that a random assignment is enough for $r \geq 183$. For a 10/9.5-approximation we need $r_k \geq 643.5$.

4.2 For $r < r_k$

In this part of the analysis we are going to use Wormalds's theorem [9]. Wormalds's theorem provides a method for analyzing parameters of a random process using differential equations.

The analysis of how to use Wormalds theorem in this case is very similar to the one for the Unit Clause with Majority (UCWM) algorithm in [2]. The main difference is that we can write equations for the number of unit clauses and the number of empty clauses in the formula.

Using Wormalds's theorem we can keep track of $C_i(t)$ -the number of clauses with i literals in the formula-, and $C_0(t)$ -the number of empty clauses at time t-. At each time, the algorithm assigns a value to a variable, therefore $0 \leq t \leq n$. We are interested in the number of empty clauses at time $t = n$, that is $C_0(n)$. $C_0(n)$ is equal to the number of clauses not satisfied by the assignment found by the algorithm.

Let $H(t) = \{C_0(t), C_1(t), \ldots, C_k(t)\}$. It is not difficult to prove (see [2] and [8]) that at any step t the formula obtained by assigning the first t selected variables is a random formula (uniform in

the space Φ_H) given the values of the parameters $C_i(t)$. This fact enables to compute the expected value of $C_i(t+1) - C_i(t)$ given the values $C_i(t)$, i.e

$$\mathbf{E}[C_i(t+1) - C_i(t)| \quad H(t)] = -\frac{iC_i}{n-t}\delta_{i\neq 0} + \mu_\lambda \frac{(i+1)C_{i+1}}{\rho}\delta_{i\neq k}$$

where $i = 0, 1 \ldots k$ and $\delta_{i\neq j}$ is 0 if $i = j$ and 1 otherwise. Let's give the definition of ρ and μ_λ.

Let $F \in \Phi_{H(t)}$ and X be the random variable defined as number of clauses in F where the random literal l appears. The distribution of X can be approximated by a Poisson with parameter $\lambda = \frac{\rho}{2(n-t)}$, where $\rho = C_1(t) + 2C_2(t) + \cdots + kC_k(t)$. In our algorithm, we take a random variable x and satisfies the literal that appears the most among $\{x, \bar{x}\}$. Denote Z the number of clauses in which the falsified literal appears. Z has the same distribution as $\mathbf{min}(X, X')$, where X' is independent to X and has the same distribution. Finally denote $\mu_\lambda = \mathbf{E}(Z)$.

Wormalds' theorem says that we can approximate the values of $C_i(t)$ by the solutions $c_i(x)$ of the following system of differential equations.

$$\frac{dc_i}{dx} = -\frac{ic_2}{1-x}\delta_{i\neq 0} + \mu_\lambda \frac{(i+1)c_{i+1}}{\rho}\delta_{i\neq k}$$

For $i = 0, 1, \ldots, k$ and with initial conditions $c_i(0) = \frac{C_i(0)}{n}$. Here $\rho = c_1 + 2c_2 + \cdots + kc_k$ the scaled number of literals in the formula and $\lambda = \frac{\rho}{2(1-x)}$. μ_λ has the same definition as before.

At any time $t < n$, $c_i(t/n)$ give a good approximation of the scaled values of $C_i(t)$. More precisely,

$$C_i(nx) = c_i(x)n + o(n)$$

with high probability when n goes to infinity.

Wormalds' theorem holds for $0 \leq x \leq 1 - \epsilon$. We use $\epsilon = 10^{-5}$. To go around the problem we can analyzed the algorithm for $0 \leq t \leq n(1-\epsilon)$ and then counting all the remaining clauses plus the empty clauses as not satisfied by the assignment. Let $t_\epsilon = n(1-\epsilon)$, we use the following bound $C_0(n) \leq C_0(t_\epsilon) + C_1(t_\epsilon) + \cdots + C_k(t_\epsilon)$.

5 Proof of the theorem. Results for MAX-R-3SAT

We solve the differential equations numerically using the *ode45* function of matlab. The values of μ_λ are approximated numerically. The results are in agreement with simulations of the algorithm on randomly generated 3-SAT formulas.

In figure 1 we give the results of approximating $1 - \frac{C_0(n)}{m}$, the fraction the clauses that are satisfied by the algorithm, with a lower bound $1 - \frac{c_0(x_\epsilon)+c_1(x_\epsilon)+c_2(x_\epsilon)+c_3(x_\epsilon)}{r}$, where $x_\epsilon = 1 - \epsilon = t_\epsilon/n$ and $c_i(x)$ is the solution of the differential equations.

We will use the following result in the proof of theorem 1.

Theorem 2. *[1] Let F be a k-CNF random formula if*

$$r(F) > \tau(p) = 2^k log\ 2/(p + (1-p)log(1-p))$$

the probability that $m(F) > (1 - 2^k(1-p))m$ goes to zero as n goes to infinity.

The result in theorem 2 provides with an upper bound in the maximum number of clauses that can be satisfied in a typical random k-CNF formula.

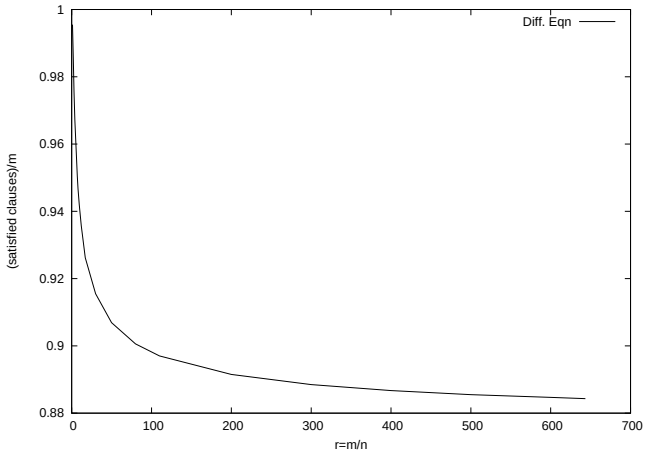

Fig. 1. Results of the differential equation approach. Plot of the function $g(r)$ (fraction clauses satisfied by the algorithm) versus the clause density r.

Proof. of Theorem 1
To prove the 10/9.1-approximation for MAX-R-3SAT we choose $r_3 = 183$ as the parameter for the algorithm. The result for $r = m/n > r_3$ holds just by the lemma in the subsection 4.1.

For $r = m/n \leq r_3$ we divide the arguments. First notice that the function $g(r)$ is a decreasing function of r. For $r \leq 12$, the function $g(r) \geq 0.9357$ so $\frac{m(F)}{m_A(F)} \leq \frac{m}{g(r)m} < \frac{10}{9.1}$ w.h.p.

For $12 < r \leq 183$ using theorem 2 for $p = 0.8$, $k = 3$ we get that for $r > 11.6$ the probability that $0.975m$ clauses can be satisfied goes to zero as n goes to infinity. Therefore, we can use that $m(F) \leq 0.975m$ w.h.p and the fact that for $r \leq 183$ $g(r) \geq 0.8922$ to obtain $\frac{m(F)}{m_A(F)} \leq \frac{0.975m}{g(r)m} < \frac{10}{9.1}$ w.h.p.

□

As we just did in the proof of theorem 1 and using more carefully theorem 2 together with the results of the analysis in subsection 4.2 we can get a 10/9.5-approximation result (we need to divided the arguments in many small intervals of the parameter r and use the same argument we gave above).

References

1. Dimitris Achlioptas, Assaf Naor, Yuval Peres.On the Fraction of Satisfiable Clauses in Typical Formulas. Extended Abstract in FOCS'03, pp. 362-370.
2. Dimitris Achlioptas, Lower Bounds for Random 3-SAT via Differential Equations, Theoretical Computer Science, 265 (1-2), (2001), p.159-185.
3. Dimitris Achlioptas, Gregory B. Sorkin, Optimal Myopic Algorithms for Random 3-SAT, in Proceedings of FOCS 00, p.590-600.
4. A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 322–330, 1993.
5. M-T. Chao and J. Franco. Probability analysis of two heuristics for the 3-satisfiability problem. SIAM J. Comput., 15(4):1106-1118, 1986.
6. Wenceslas Fernandez de la Vega, Marek Karpinski. 9/8-Approximation Algorithm for Random MAX-3SAT. Electronic Colloquium on Computational Complexity (ECCC)(070): (2002)
7. Alexis C. Kaporis, Lefteris M. Kirousis, Efthimios G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. In 10th Annual European Symposium on Algorithms (Rome, Italy, 2002).
8. A.C. Kaporis, L.M. Kirousis, and Y.C. Stamatiou, "How to prove conditional randomness using the Principle of Deferred Decisions," Technical Report, Computer Technology Institute, Greece, 2002. Available at: www.ceid.upatras.gr/faculty/kirousis/kks-pdd02.ps
9. N. C. Wormald, Differential equations for random processes and random graphs, Ann. Appl. Probab. 5 (4) (1995) 1217–1235. 36