

# Approximation algorithm for Random MAX- $k$ SAT

Yannet Interian

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

**Abstract.** We provide a rigorous analysis of a greedy approximation algorithm for the maximum random  $k$ -SAT (MAX-R- $k$ SAT) problem. The algorithm assigns variables one at a time in a predefined order. A variable is assigned TRUE if it occurs more often positively than negatively; otherwise, it is assigned FALSE. After each variable assignment, problem instance is simplified and a new variable is selected. We show that this algorithm gives a  $10/9.5$ -approximation, improving over the  $9/8$ -approximation given by de la Vega and Karpinski [7]. The new approximation ratio is achieved by using a different algorithm than the one proposed in [7], along with a new upper bound on the maximum number of clauses that can be satisfied in a random  $k$ -SAT formula [2].

## 1 Introduction

In the MAX  $k$ -SAT problem we are given a Boolean formula in conjunctive normal form, with  $k$  literals in each clause, and we ask for a truth assignment that maximizes the number of satisfied clauses. The random version of this optimization problem considers inputs drawn from a predefined probability distribution. An  $\alpha$ -approximation algorithm for the Maximum random  $k$ -SAT problem (MAX-R- $k$ SAT) finds with high probability (w.h.p.)<sup>1</sup> an assignment satisfying at least  $\alpha$  times the maximum number of possible satisfiable clauses.

The most popular model for generating random SAT problems is the uniform  $k$ -SAT model, formed by uniformly and independently selecting  $m$  clauses from the set of all  $2^k \binom{n}{k}$   $k$ -clauses on a given set of  $n$  variables. Interesting problems for this model arise when the ratio  $\alpha$  of clauses to variables remains constant as the number of variables increases. The most famous conjecture is that such formulas exhibit a “phase transition” as a function of  $\alpha$  [11]. There exists a constant  $c_k$  such that, uniform  $k$ -SAT problem instances with values of  $\alpha$  below the threshold  $\alpha_k$ , typically have one or more satisfying assignment, whereas problems with  $\alpha$  larger than  $\alpha_k$  have too many constraints and become unsatisfiable.

We propose the analysis of a simple greedy algorithm for approximating MAX-R- $k$ SAT. Previous work on this problem was done by de la Vega and Karpinski [7], where a  $9/8$ -approximation algorithm for MAX-R-3SAT is analyzed. We improve upon this ratio by analyzing a different algorithm, and using recent results Achlioptas *et al* [2] giving 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 [13]. This method has been used extensively in the approximation (lower bounds) of the satisfiability threshold (see [1, 8, 3]). To use this method for MAX-R- $k$ SAT, we have to be able to compute not the probability of finding an assignment using a certain algorithm, as done for the random 3-SAT problem [1, 8, 3], but the expected number of clauses that such an assignment satisfies.

## 2 Outline of the results

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

---

<sup>1</sup> The events  $\mathcal{E}_n$  hold with high probability (w.h.p.) if  $\Pr(\mathcal{E}_n) \rightarrow 1$  when  $n \rightarrow \infty$

We propose an algorithm which, given a  $k$ -CNF formula  $F$ , outputs an assignment  $A$ , leading to  $m_A(F)$  as our approximation to  $\max(F)$ . We prove that

$$\lim_{n \rightarrow \infty} \Pr \left\{ \frac{\max(F)}{m_A(F)} \leq \alpha \right\} = 1 \quad (1)$$

To obtain (1) 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.*,  $\Pr \{m_A(F) = g(r)m + o(m)\}$  goes to 1 as  $m$  goes to infinity. Then we show that  $\Pr \{\max(F) \geq \alpha(g(r)m + o(m))\}$  goes to zero as  $m$  goes to infinity, where  $\alpha$  is the approximation constant.

In part of the analysis we use an upper bound for the value of  $\max(F)$ , as given in the results of Achlioptas *et al.* [2]. We prove the following result for random MAX-R-3SAT.

**Theorem 1** *There is a polynomial time algorithm for approximating MAX-R-3SAT with approximation ratio  $\alpha = 10/9.1$ .*

**Remarks:** In section 5 we discuss how to extend the proof of theorem 1 to obtain a 10/9.5 ratio. The proof can be further generalized to obtain results for any fixed  $k$ .

### 3 The Algorithm

The main loop of our greedy algorithm is 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

```

where  $r_k$ , is a constant that depends on  $k$ . For  $k = 3$ , we use  $r_3 = 183$ , as will be explained in the the proof of theorem 1 in section 5.

If  $r(F) > r_k$ , the output is a random assignment. We can change this part of the algorithm to output any fixed assignment, turning the algorithm into a deterministic algorithm. We will show below that any fixed assignment satisfies  $\frac{7}{8}m + o(m)$  clauses w.h.p., and moreover we prove that  $\frac{7}{8}m + o(m)$  is a good approximation of  $\max(F)$  when  $r$  sufficiently large.

If  $r(F) \leq r_k$ , the algorithm proceeds as follows: while there are unassigned variables, select an unassigned variable  $x$ . If  $x$  appears positive (*i.e.*, appears as  $x$  as opposed to  $\bar{x}$ ) in at least half of the clauses that contain  $x$ ,  $x$  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  $x$ 
    If  $x$  appears positively in at least half of the remaining
    clauses (in which  $x$  appears)
      Set  $x = \text{TRUE}$ 
    Otherwise
      Set  $x = \text{FALSE}$ 
    Del&Shrink
  end do
  output the current assignment
end

```

Chao and Franco [5] proposed a unit clause with the majority rule algorithm for the study of the satisfiability threshold for random 3-SAT formulas. The majority rule used by Chao and Franco [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 a strategy aims at finding satisfying assignments.

It's an interesting question for future research whether adding unit-clause propagation (*i.e.*, selecting variables occurring in unit clauses first) is helpful in the MAX-SAT problem when the problem instances are over-constrained. For finding a satisfying assignment (assuming such an assignment exists), unit propagation, and its generalization “the shortest clause first” heuristic, have been shown to be very effective both empirically and in formal analysis. However, in standard satisfiability testing one has in some sense “no choice” — once a unit clause is obtained, the variable in the clause has to be set such that the unit clause is satisfied. In over-constrained MAX-SAT instances, the situation is quite different. Once a unit clause is obtained, the question is whether one should proceed to satisfy that clause or instead work on satisfying other clauses and leave the unit clause unsatisfied. In order to satisfy the maximum number of clauses, it may be beneficial to not treat unit clauses any different from other clauses. Our intuition is that after a certain number of variable settings, a sufficient number of unit clauses appears, such that from then on, a procedure with unit propagation will only set variables in unit clauses (note that setting a few unit clauses will generally produce new unit clauses). It would be interesting to analyze the production of unit clauses as a branching process and check whether at some point the expected number of offspring (new unit clauses) is greater than one.

Our algorithm is also similar to the one proposed in de la Vega and Karpinski [7]. Their assignment strategy is static. The algorithm assigns every variable to its majority value, *i.e.*,  $x$  is assigned to TRUE if  $x$  appears positively more often than negatively in the original formula; otherwise, it is assigned to FALSE. Our algorithm is similar but proceeds dynamically. It assigns one variable at a time and simplifies 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

In this section we prove that if  $A$  is the assignment given by the algorithm, then we know with an  $o(m)$  error, the number of clauses that  $A$  satisfies. More precisely,  $m_A(F) = g(r)m + o(m)$  for some function  $g(r)$  that depends only on the parameter  $r$ .

### 4.1 For $r > r_k$

This part of the analysis shows that for large values of the parameter  $r$ , a random assignment, or any fixed assignment, will yield a good approximation.

Any fixed assignment  $A$  will satisfy  $(1 - \frac{1}{2^k})m + o(m)$  clauses w.h.p., which can be seen from the following argument. Let  $A$  be a fixed assignment and  $F$  a random  $k$ -SAT formula. The number of clauses satisfied by  $A$  is the sum of  $m$   $\{0, 1\}$ -independent random variables. That is, if  $X_i = 1$  if the  $i$ th clause is satisfied by  $A$ , and 0 otherwise for  $1 \leq i \leq m$ , then  $m_A(F) = \sum_{i=1}^m X_i$  where  $X_i$  are independent identically distributed, and  $\Pr(X_i = 1) = 1 - \frac{1}{2^k}$ . So  $\mathbf{E}(m_A) = (1 - \frac{1}{2^k})m$ . Moreover,  $\mathbf{Var}(m_A) = m\mathbf{Var}(X_1) = m\frac{1}{2^k}(1 - \frac{1}{2^k})$ , using Chebyshev's inequality we obtain

$$\Pr\{|m_A - \mathbf{E}(m_A)| \geq m^{2/3}\} \rightarrow 0 \text{ as } m \rightarrow \infty$$

therefore,  $m_A = (1 - \frac{1}{2^k})m + o(m)$  w.h.p.

The next result says that  $\max(F)$  is very close to  $(1 - \frac{1}{2^k})m$  for large values of the parameter  $r$  and so very close to  $m_A(F)$ , for any fixed  $A$ .

**Lemma 1** [7] *For every  $\epsilon$  there exists  $r_{\epsilon,k}$  such that for  $r \geq r_{\epsilon,k}$  and  $F$  a random  $k$ -SAT formula*

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

*goes to zero as  $n$  goes to infinity.*

*Proof.* Let  $q = 1 - \frac{1}{2^k}$ . Note that, the random variable  $\max(F) = \max_{A \in \{0,1\}^n} m_A(F)$  and that for any fixed  $A$ ,  $m_A(F)$  has distribution binomial with parameter  $m$  and  $q$  ( $\mathbf{Bin}(m, q)$ ).

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

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-R-3SAT, we take  $\epsilon = 1/0.91 - 1$  and obtain that a random assignment (or any fixed assignment  $A$ ) gives the desired approximation ratio for  $r_3 \geq 183$ . To achieve the 10/9.5-approximation, we set  $r_3 \geq 643.5$ .

## 4.2 For $r < r_k$

We now analyze the majority rule algorithm. With this analysis we aim to compute how many clauses are satisfied by the assignment chosen by the algorithm, or equivalently, how many empty clauses are generated during the assignment process. To do that, we trace certain parameters during the execution of the algorithm. One of those parameters is the number of empty clauses generated up to time  $t$ . (An empty clause is generated when all literals in the clause are assigned FALSE, so the clause is not satisfied by the current assignment.) An important aspect of the algorithm is that the order in which the variables are assigned has to be chosen in advance, *i.e.*, without looking at the particular formula. Each variable is assigned using the majority rule, *i.e.*, if the variable occurs more positively than negatively in the remaining clauses, set the variable to TRUE, else to FALSE. Using this approach, we are assured that the remaining formula is still a uniformly random formula, in the following sense.

Our algorithm sets variables one at a time. If we start for example with a 3-SAT formula, after assigning a variable we end up with a mix of 2-clauses and 3-clauses. After the next assignment, we may have some unit clauses, and even some empty clauses. Starting with a  $k$ -SAT formula, at any time  $t$ , the remaining formula has  $n - t$  variables and a mix of  $\{1, \dots, k\}$ -clauses and empty clauses. We define a random model that includes clauses of different length, and that includes random  $k$ -SAT as the special case in which all clauses have length  $k$ . The model is very simple. If  $n$  is the number of variables, and there are  $C_i$ ,  $i$ -clauses for  $1 \leq i \leq k$ , we generate for each  $i$ , a  $i$ -SAT random formula with  $n$  variables and  $C_i$  clauses, and we consider the formula  $F$  to be the conjunction of all the clauses. Denote  $\Phi_C$  a random formula generated in this way with  $C = (C_1, \dots, C_k)$ .

Fix  $k$ , denote by  $C_i(t)$  the number of clauses with  $i$  literals remaining at time  $t$ ,  $1 \leq i \leq k$ , and  $C_0(t)$  the number of empty clauses at time  $t$ . The next lemma establishes that at the end of each step  $t$  of the algorithm the remaining formula is random on the space of formulas  $\Phi_{C(t)}$  with  $C(t) = (C_1(t), \dots, C_k(t))$ .

**Lemma 2** [5, 6] *For every time  $1 \leq t \leq n$ , conditional on the values  $C_i(t)$ ,  $1 \leq i \leq k$ , the number of clauses of length  $i$ , the remaining formula is a random formula with parameters  $C_i(t)$ ,  $1 \leq i \leq k$  and  $n' = n - t$  variables.*

Lemma 2 can be used to compute parameters of the formula conditioned on the values of  $C_i(t)$   $0 \leq i \leq k$ . For example we can compute expected value of  $C_i(t+1) - C_i(t)$  given the values of  $C_i$  at time  $t$ .

The analysis we propose here relies on the method of differential equations described in [13]. The sketch of the analysis is as follows: suppose  $C(t) = (C_0(t), \dots, C_k(t))$  are stochastic parameters related to a formula, in our case are the number of  $i$ -clauses at time  $t$ . We want to estimate the trajectory of  $C(t)$  through the duration of our algorithm. In a restricted version, the theorem states that if

- (a)  $\Pr(|C_i(t+1) - C_i(t)| > n^{1/5}) = o(n^{-3})$   
(b)  $\mathbf{E}(C_i(t+1) - C_i(t) | C(t)) = f_i(t/n, C(t)/n) + o(1)$   
(c) the functions  $f_i$  are continuous and satisfies a Lipschitz condition on some set  $D$   
then

$$C_i(t) = nc_i(t/n) + o(n),$$

where  $c_i(x)$  is the solution of the system of differential equations

$$\frac{dc}{dt} = f(x, c) \quad c(0) = \left(\frac{C_0(0)}{n}, \dots, \frac{C_k(0)}{n}\right) = (0, \dots, \alpha),$$

and  $c = (c_0, c_1, \dots, c_k)$ .

In our case, the equation for the conditional expectation of  $C_i(t+1) - C_i(t)$  is as follows:

$$\mathbf{E}[C_i(t+1) - C_i(t) | C_0(t), \dots, C_k(t)] = -\frac{iC_i}{n-t}\delta_{i \neq 0} + \mu_\lambda \frac{(i+1)C_{i+1}}{\rho}\delta_{i \neq k} \quad (2)$$

where  $i = 0, 1 \dots k$  and  $\delta_{i \neq j}$  is 0 if  $i = j$  and 1 otherwise. Note we have an equation for each value of  $i$ .

To better understand these difference equations, let's consider a specific case. For example, with  $k = 3$  and  $i = 3$ , we obtain

$$\mathbf{E}[C_3(t+1) - C_3(t) | C_0(t), \dots, C_3(t)] = -\frac{3C_3}{n-t}$$

The terms on the right, measure the expected reduction of ternary clauses at time  $t$ . Note that, at time  $t$  there are  $n - t$  variables, so the probability that a 3-clause has a fixed variable  $x$  is  $\frac{\binom{n-1}{2}}{\binom{n}{3}}$ .

Therefore,  $\frac{3C_3}{n-t}$  is the expected number of 3-clauses with a variable  $x$ .

The definition of  $\rho$  and  $\mu_\lambda$  are as follows. Let  $C(t) = (C_1(t), \dots, C_k(t))$ ,  $F \in \Phi_{C(t)}$  and  $X$  be the random variable defined as the number of clauses in  $F$  where the random literal  $l$  appears. The distribution of  $X$  can be approximated by a Poisson random variable with parameter  $\lambda = \frac{\rho}{2(n-t)}$ , where  $\rho = C_1(t) + 2C_2(t) + \dots + kC_k(t)$ . The algorithm takes a variable  $x$  and satisfies the literal that appears the most among  $\{x, \bar{x}\}$ . Denote by  $Z$  the number of clauses in which the falsified literal appears (that is, the literal that appears the least number of times from among  $\{x, \bar{x}\}$ ).  $Z$  has the distribution of  $\min(X, X')$ , where  $X'$  is independent to  $X$  and have both distribution Poisson with parameter  $\lambda$ . Let  $\mu_\lambda = \mathbf{E}(Z)$ , be the expected value of  $Z$ .

Wormald's 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.

$$\frac{dc_i}{dx} = -\frac{ic_2}{1-x}\delta_{i \neq 0} + \mu_\lambda \frac{(i+1)c_{i+1}}{\rho}\delta_{i \neq k} \quad (3)$$

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

At any time  $t < (1 - \epsilon)n$ ,  $c_i(t/n)$  gives a good approximation of the scaled values of  $C_i(t)$ . More precisely,

$$C_i(nx) = c_i(x)n + o(n) \quad (4)$$

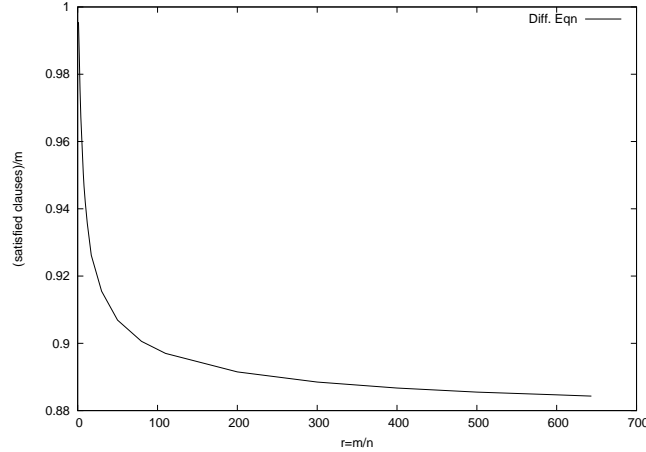
with high probability when  $n$  goes to infinity.

Wormald's theorem can be applied for  $0 \leq x \leq 1 - \epsilon$ , for any  $\epsilon > 0$ . That is because  $x = 1$  is a singularity point for our function  $f$ . In our analysis, we take  $\epsilon = 10^{-5}$ . To get around the problem of not having equation (4) for all the values of  $t$ , we analyze the algorithm for  $0 \leq t \leq n(1 - \epsilon)$  and then count 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  $C_0(n) \leq C_0(t_\epsilon) + C_1(t_\epsilon) + \dots + C_k(t_\epsilon)$  for the number of empty clauses generated by the algorithm. A precise statement of the theorem is given on the Appendix.

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

We solve the differential equations (3) numerically using the *ode45* function of matlab. The values of  $\mu_\lambda$  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 of 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  $c_i(x)$ ,  $1 \leq i \leq 3$  are the solutions of the differential equations.



**Fig. 1.** Fraction of satisfied clauses ( $g(r)$ ) as the function of  $r$ . Results from the solution of the differential equations.

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

**Theorem 2** [2] *Let  $F$  be a  $k$ -CNF random formula, if*

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

*then  $\Pr(\max(F) > (1 - 2^k(1-p))m)$  goes to zero as  $n$  goes to infinity.*

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

*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 split our proof into two parts, for  $r \leq 12$  and  $12 \leq r \leq 183$ . For  $r = 12$ , the function  $g(r) \geq 0.9357$ , as  $g(r)$  is a decreasing function of  $r$  then  $g(r) \geq 0.9357$  for  $r \leq 12$ . Then  $\frac{\max(F)}{m_A(F)} \leq \frac{m}{g(r)m} < \frac{10}{9.1}$  w.h.p. Here we take  $m$  as an approximation to the optimal value of  $\max(F)$ .

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  $\max(F) \leq 0.975m$  w.h.p. and the fact that for  $r \leq 183$   $g(r) \geq 0.8922$  to obtain  $\frac{\max(F)}{m_A(F)} \leq \frac{0.975m}{g(r)m} < \frac{10}{9.1}$  w.h.p.

The 10/9.5-approximation result can be obtained by carefully dividing the interval  $r \in (0, 643.5)$  in several pieces. For each piece, using theorem 2, we get an upper-bound for  $\max(F)$ , and our function  $g(r)$  for the approximation of  $m_A(F)$ . The analysis for  $r \geq 643.5$  comes from the results in subsection 4.2.

## References

1. D. Achlioptas. Lower Bounds for Random 3-SAT via Differential Equations. *Theoretical Computer Science*, 265 (1-2), p.159-185 (2001).
2. D. Achlioptas, A. Naor, and Y. Peres. On the Fraction of Satisfiable Clauses in Typical Formulas. Extended Abstract in FOCS'03, p. 362-370.
3. D. Achlioptas and G. 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*, p. 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) p.1106-1118, (1986).
6. M. T. Chao, and J. Franco. Probabilistic analysis of a generalization of the unit clause selection heuristic for the  $k$ -satisfiability problem. *Information Sciences* 51 p. 289-314, (1990).
7. W. Fernandez de la Vega, and M. Karpinski. 9/8-Approximation Algorithm for Random MAX-3SAT. *Electronic Colloquium on Computational Complexity (ECCC)(070)* (2002).
8. A. C. Kaporis, L. M. Kirousis, and E. G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. In *10th Annual European Symposium on Algorithms (Rome, Italy, 2002)*.
9. A. C. Kaporis, L. M. Kirousis, and E. Lalas. Selecting complementary pairs of literals. *Electronic Notes in Discrete Mathematics*, Vol. 16 (2003).
10. 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](http://www.ceid.upatras.gr/faculty/kirousis/kks-pdd02.ps).
11. D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of sat problems. In *Proc. 10-th National Conf. on Artificial Intelligence (AAAI-92)*, p. 459–465.
12. B. Selman, D. Mitchell, and H. Levesque. Generating Hard Satisfiability Problems. *Artificial Intelligence*, Vol. 81, p. 17–29, (1996).
13. N. C. Wormald. Differential equations for random processes and random graphs. *Ann. Appl. Probab.* 5 (4) p. 1217–1235. 36, (1995).

## A Differential Equations

We consider here a sequence of random process  $Y_t = Y_t(n), n = 1, 2, \dots$ . For simplicity the dependence on  $n$  is dropped from the notation. Let  $\mathcal{F}_t$  be the the  $\sigma$ -algebra generated by the process up to time  $t$ , i.e  $\mathcal{F}_t = \sigma(Y_0, Y_1, \dots, Y_t)$ . Our process  $Y_t = (Y_t^{(1)}, \dots, Y_t^{(j)})$  is a vector of dimension  $j$ . Let  $\|Y\| = \max(|Y^{(1)}|, \dots, |Y^{(j)}|)$ . Suppose that  $Y_0 = z_0 n$  the value of the process at time 0.

We say that  $X = o(f(n))$  *always* if  $\max\{x : \Pr(X = x) \neq 0\} = o(f(n))$ . The term *uniformly* means that the convergence implicit in the  $o()$  is uniform on  $t$ .

**Theorem 3** [13] *Let  $f : \mathbb{R}^{j+1} \rightarrow \mathbb{R}^j$ . Suppose there exists a constant  $C$  such that the process  $Y_t$  is bounded by  $Cn$ , i.e  $\|Y_t\| < Cn$ . Suppose also that for some function  $m = m(n)$ :*

(i) *uniformly over all  $t < m$*

$$\Pr(\|Y_{t+1} - Y_t\| > n^{1/5} | \mathcal{F}_t) = o(n^{-3})$$

*always;*

(ii) *for all  $l$  and uniformly over all  $t < m$ ,*

$$\mathbf{E}(Y_{t+1} - Y_t | \mathcal{F}_t) = f(t/n, Y_t/n) + o(1)$$

*always;*

(iii) *The function  $f$  is continuous and satisfies a Lipschitz condition on  $D$ , where  $D$  is some bounded open set containing  $(0, z_0^{(1)}, \dots, z_0^{(j)})$ .*

*then:*

(a) *The system of differential equations*

$$\frac{dz}{ds} = f(s, z)$$

has a unique solution in  $D$  for  $z : \mathbb{R} \rightarrow \mathbb{R}^j$  with initial conditions  $z(0) = z_0$  and which extends to points arbitrarily closed to the boundary of  $D$ .

(b)

$$Y_t = nz(t/n) + o(n) \text{ w.h.p}$$

uniformly for  $0 \leq t \leq \mathbf{min}\{\sigma n, m\}$ , where  $\sigma$  is the supremum of those  $s$  to which the solution can be extended.