CS761 (Randomized Algorithms) Final Project
Satisfiability and the `Walksat` Algorithm

Aditya Jayaprakash
ajayapra@edu.uwaterloo.ca

1 Introduction

The k-CNF problem is to find a satisfying assignment to a CNF formula where each clause has k variables. The most famous example of it is 3-SAT, where each clause has exactly three variables. 3-SAT was one of the first problems known to be NP-COMPLETE, proved the Cook-Levin theorem. Although there isn't a proof yet to whether or not $P \neq NP$, conventional wisdom suggests that the problem of finding a solution is much harder than verifying if it is correct.

Since satisfiability is NP-COMPLETE, one would imagine that is extremely difficult or largely infeasible to obtain solutions for a CNF formula for large n, since the current known algorithms have worst case runtime of $c^{f(n)}$ for some constant $c > 1$ for the problem of satisfiability. Yet, in practice, things seem very different. Today, there are SAT-solvers that find a solution to formulas with millions of variables and millions of clauses. Lots of industrial problems related to circuit design, logistics, etc. are often reduced to satisfiability. A big question is about how SAT instances with millions of variables are solved despite SAT being NP-COMPLETE. We will try to address this issue in this report.

We will start by making a few observations about k-CNF. Suppose we have a k-CNF formula F with n variables and m clauses. Fix a clause C. There are 2^k assignment to this clause C, but only one assignment for which C is not satisfiable i.e., there are $2^k - 1$ satisfying assignments. Suppose we give a random assignment to variables in C, it doesn't satisfy C with probability 2^{-k}. By the union bound, the probability that a random assignment satisfies F is at least $1 - m2^{-k}$. So when $\frac{m}{2^k}$ is very small, it is highly likely that a random assignment would satisfy F. Likewise, when m is close to 2^k, it becomes extremely unlikely that a random assignment satisfies F.

Using the Lovaszs Local Lemma [3], we can prove that k-CNF formula has a satisfying assignment if no variable appears in more than $\frac{2^k}{4k}$ clauses. Moser [6] proposed a fast algorithm to find such a satisfying assignment.

Another interesting aspect of k-CNF is it exhibits the sharp threshold phenomena based on the density $d = \frac{m}{n}$. In [1], it was shown that when $d << 2^k \log 2 - O(k)$, we're almost guaranteed that such a formula will have a satisfying assignment, but above the threshold, it sharply drops and becomes almost unlikely that it will have a satisfying assignment. From the ideas mentioned above, it is clear than not all instances of SAT are hard. A big open question is what makes SAT hard. We will try to answer some of those questions. There are numerous SAT algorithms, the most successful and well known are,

1. `Walksat` [8]

2. DLL-type algorithms
3. Unit Clause Propogation method

All the above algorithm perform very well in practice in finding a satisfying assingment for a k-CNF formula. The remaining focus of this report will be to analyze [5] and explain how it bounds the running time of the algorithm based on the density upper bounded by a function of k using the smoothed analysis model.

We will now talk about the smoothed analysis model of k-CNF. Smoothed analysis was introduced by Spielman and Teng [9] in order to explain why the simplex algorithm is efficient in practice, despite being exponential in the worst case. It is a more realistic measure of the performance of the algorithm by perturbing the worst-case inputs. One can observe that if the smoothed complexity is low, then it is very unlikely that an algorithm will take exponentially long is practical instances. We will now summarize the results of [5], which will be the central focus on this report.

2 Smoothed Analysis of k-CNF

We will first describe a very natural distribution for analyzing k-CNF. We will fix a $c, n > 0$ and choose $m = cn$ clauses uniformly at random. We will denote this distribution by $\mathcal{F}_{n,m,k}$. Given a formula F, we will now describe a model for obtaining a perturbed formula. Given an F, we obtain an ϵ-perturbation, called F^*, as follows, *flip the polarity of every literal in every clause, independently of the others, with probability* ϵ. We will denote $d_{\epsilon,k}$ to be the smallest integer such that for every k-CNF F with at least $d_{\epsilon,k}$ clauses, F^* is not satisfiable with high probability.

One of the goals of this paper is to give (up to logarithmic order) the dependency of $d_{\epsilon,k}$ on ϵ and k. In order to motivate the rest of the work, let us start by proving a simple proposition.

Proposition 0.1 *Let F be a k-CNF instance with m clauses over n variables, and let $d = \frac{m}{n}$. Let ϵ be the perturbation parameter, and let F^* be the perturbed instance. If $d > \epsilon^{-k} \ln 2$ then with high probability F^* is not satisfiable.*

Proof Sketch: Let us fix an assignment ψ. Pick any clause C in F. If C contains exactly i literals that are true under ψ, then C will continue to be satisfied in F^* with probability $1 - (1-\epsilon)^{k-i}\epsilon^i \leq 1 - \epsilon^k$. This is assuming $\epsilon \leq \frac{1}{2}$. There are m clauses that all are perturbed independently, hence the probability that ψ satisfies F^* is at most $(1-\epsilon^k)^m \leq e^{-m\epsilon^k} < 2^{-n}$. Since there are 2^n possible values for ψ, by the union bound, the probability is less than 1, proving our proposition. ■

We will now give provable upper and lower bounds for the value of d so that F^* is satisfiable with high probability when it is below d and isn't satisfiable with high probability when it is bigger than d. k-CNF exhibits the phase transition property where d is the threshold. The following theorem proves an upper bound

Theorem 0.2 *(upper bound) Let F be a k-CNF instance with m instances over n variables, and let $d = \frac{m}{n}$. Let ϵ be the perturbation parameter, and let F^* be the perturbed instance. There exists an absolute constant $c_1 > 0$ such that if*

$$d \geq \frac{c_1}{\epsilon^{k-1}} \cdot \log \frac{1}{\epsilon}$$

then with high probability F^* is not satisfiable.

We define a formula F to be w-expanding if for every $1 \leq t \leq n$, every subset of t variables participates in at least tw clauses. A key step in the proof is to consider a $\frac{d}{2}$ expanding subformula F', and prove that $F^{'*}$, F's ϵ-smoothed version is unlikely to be satisfiable. An implication is that since $F^{'*}$ is a subformula of F', F^* is unlikely to be satisfiable as well. We will also improve upon the first proposition that we proved.

Given a k-CNF formula F and some assignment ψ, let $\psi_i(F)$ be the number of clauses that have exactly i literals that are true under ψ. The probability that the assignment ψ survives an ϵ-perturbation is, $\prod_{i=0}^{k}\left(1-\epsilon^i(1-\epsilon)^{k-i}\right)^{\psi_i(F)}$. We must note that this is much more accurate bound of the probability that ψ survives compared to the bound we have from the proposition we proved earlier. The paper also proves a lemma that a formula F has a subformula F' that is $\frac{d}{2}$-expanding. We will use the lemma, without proving it.

Lemma 0.3 *Let F be a k-CNF instance with m clauses over n variables with density d (where $d = \frac{m}{n}$). F has a $\frac{d}{2}$-expanding subformula F'. Furthermore, F' contains at least $\frac{m}{2}$ clauses.*

The next step is to show that there aren't too many assignments for which $\psi_k(F)$ is too large. This is shown using the lemma

Lemma 0.4 *Let F be a d-expanding k-CNF formula on n variables and m clauses. There are at most $n\binom{n}{g/d}$ assignments satisfying $\psi_k(F) = m - g$.*

To finally prove the theorem, we have to union bound the on the probability of F^* being satisfiable, parameterized by the number of clauses not satisfied k times. For a fixed value of $\psi_k(F)$, the probability that ψ survives the ϵ perturbation where $\psi_k(F) = m - g$ is

$$n\binom{n}{g/d}\prod_{i=0}^{k}\left(1-\epsilon^i(1-\epsilon)^{k-i}\right)^{\psi_i(F)} \leq n\binom{n}{g/d}\left(1-\epsilon^k)^{\psi_k(F)}\right)\cdot\left(1-(1-\epsilon)\epsilon^{k-1}\right)^{|F|-\psi_k(F)} \leq n\binom{n}{g/d}\left(1-\epsilon^k\right)^{m-g}\left(1-\frac{\epsilon^{k-1}}{2}\right)^g$$

We know the range of $\psi_k(F)$ is between 0 and m, so we can enumerate over all possible values of $\psi_k(F)$ to arrive at

$$n\sum_{g=0}^{m} n\binom{n}{g/d}\left(1-\epsilon^k\right)^{m-g}\left(1-\frac{\epsilon^{k-1}}{2}\right)^g$$

The rest of the proof is simplifying the above equation and showing that for our value of $d > \frac{5}{\epsilon^{k-1}}\ln\left(\frac{1}{\epsilon}\right)$, the equation holds. Hence, we've shown how to prove this theorem. Our next goal is to prove the lowerbound.

Theorem 0.5 *(lower bound) There exists a constant c_2 and a k-CNF F with m clauses over n variables, and $d = \frac{m}{n}$ satisfying*

$$d \leq \frac{c_2}{\epsilon^{k-1}} \cdot \frac{1}{k^2},$$

so that its ϵ-perturbation, F^* is whp satisfiable. Furthermore $c_2 \geq \frac{1}{42}$.

We will first start by describing the a procedure which we will use throughout this theorem to generate F.

Procedure 0.6 *Go over all $\binom{n}{k}$ clauses that contain only positive literals; include each such clause with probability $p = \frac{dn}{\binom{n}{k}}$*

By using the expectation and Chernoff bounds, we can show that it will have dn clauses with high probability. Our goal is to show that F^* is also satisfiable with high probability. In fact, the analysis will show an even stronger argument i.e., there exists many formulas of size dn such that the corresponding F^* is satisifiable with high probability.

To start, we will divide our variables into two sets, one where all the variables stay TRUE in the satisfying assignment of F^*. We will call the other set a feeble set, A and we define it recursively as follows,

1. $A_0 = \{x | \exists (\overline{x} \vee \overline{y_1} \vee \ldots \vee \overline{y}_{k-1} \text{ in } F^*\}$.
2. $A_i = \{x | \exists \text{ clause containing } \overline{x} \text{ in which all non-negated variables belong to } \cup_{j \leq i-1} A_j\}$

We can write the feeble set $A = \cup_{i \geq 0} A_i$ to be the set of feeble variables. Let L be a set of variables and $F[L]$ denotes subformula of F where in each clause, all k literals are in L. One can observe that if F is generated via the procedure defined above, and F^* is its ϵ-perturbation. Every assignment that sets all variables in $V \setminus A$ to TRUE satisfies clauses in $F^* \setminus F^*[A]$. We can write $F^* = F^*[A] \cup F^* \setminus F^*[A]$. Now that we have shown how to satisfy $F^* \setminus F^*[A]$, we will shown that $F^*[A]$ also has a satisfying assignment with high probability.

Definition 0.7 *A variable x is called pure in F if it appears only in one polarity. A formula F is called complicated if all its variables are not pure. A formula is called degenerate if none of its subformulas is complicated.*

We will now introduce the *pure literal rule*: Repeat while possible: pick a variable that appears only in one polarity, set it in a satisfying manner, and remove all clauses in which it appears.

We can observe that if F is degenerate, then every subformula, there is some variable that is pure. Hence, we can use the pure literal rule to solve it.

Since we want to show that the pure literal rule can satisfy some subformulas, we have the following proposition which we will not prove.

Proposition 0.8 *Let F be generated via the random process defined in the procedure above, and let F^* be its ϵ-perturbation. If $d \leq \frac{\epsilon^{1-k}}{42k^2}$ then with high probability (over the choice of F and F^*), every subformula F' of F^* induced by at most $\frac{\epsilon n}{k}$ variables is degenerate.*

Returning to the original problem, if we want to show that the pure literal rule solves $F^*[A]$, we would have to show $|A| \leq \frac{\epsilon n}{k}$. We use the proposition to show it, but we will not prove it.

Proposition 0.9 *Let F be generated via the random process described above, and let F^* be its ϵ-perturbation. If $d = \frac{c\epsilon^{1-k}}{3k^2}$ where $c \leq 1$ then with high probability (over the choice of F and F^*), $|A| \leq \frac{c\epsilon n}{k}$.*

Using the last two lemmas, we can show that $F^*[A]$ is degenerate since $|A| \leq \frac{\epsilon n}{k}$ with high probability. Using the other proposition, we know that if $d \leq \frac{\epsilon^{1-k}}{42k^2}$, then every subformula F' of F^* induced by at most $\frac{\epsilon n}{k}$ variables is degenerate. We wrote earlier that if a formula is degenerate, then we can use the pure literal method to solve it. We will indeed use the pure literal method to satisfy $F^*[A]$. We will write this as a proposition.

Proposition 0.10 *Let F be generated via the random process defined, and let F^* be its ϵ-perturbation. If $d \leq \frac{\epsilon^{1-k}}{42k^2}$, then with high probability, $F^*[A]$ is satisfiable.*

We have shown that $F^*[A]$ is satisfiable with high probability. We earlier showed that $F^* \setminus F^*[A]$ is satisfiable. Since $F^* = F^*[A] \cup F^* \setminus F^*[A]$, F^* is satisfiable with high probability as well.

2.1 Connection to `Walksat`

We will now describe the startling link between smoothed analysis and the `Walksat` algorithm. In this section, we assume $k \geq 5$. If we set $\epsilon = \frac{1}{2}$ adn let A' be the set of variables defined as follows: $x \in A'$ if and only if there exists an execution in `Walksat`, starting from all all-TRUE assignment, such that x is flipped by `Walksat`. We know that $A' \subseteq A$ where A is the set of feeble variables defined earlier.

Since $A' \subseteq A$, the insight is that it is enough to analyze the performance of Walksat on $F[A]$. Since we know A is small with high probability from the previous proposition, it is much easier to analyze. We want to prove the following theorem that bounds the running time of `Walksat` with respect to the density.

Theorem 0.11 *Let F be a random instance of $\mathcal{F}_{n,m,k}$. If $k \geq 5$ and $\frac{m}{n} \leq \frac{2^k}{30k^2}$, then on input F `Walksat` finds a satisfying assignment after flipping only $O\left(\frac{n}{k}\right)$ variables whp. Furthermore, if $k \geq 30$, then the same is true for $\frac{m}{n} \leq \frac{2^k}{6k^2}$.*

We will reduce this problem to the problem of finding a matching. In order to bound the degree of a vertex, we will need to use the following lemma.

Lemma 0.12 *Let F be a random formula generated by the procedure in the last section. Let $d = \frac{c2^k}{k^2}$ for some suitably chosen constant $c = c(k)$ (to be determined in the proof). Let A be the set of feeble variables. Then F enjoys the following property with probability $1 - o(n^{-1})$: for any $S \subset V$ of size $|S| \leq |A|$ the formula $F[S]$ contains at most $\frac{3|S|}{2k}$ clauses.*

Proof Sketch: We prove it using the first moment. Give a set S of size $s = \beta n$, there are $\binom{n}{s}$ ways to choose set S. Let $p = \frac{3}{2k}$. The probability that any given S contains $t = ps$ clauses is $\binom{\binom{s}{k}}{t} p^t$. By the first moment, it is at most $\binom{n}{s}\binom{\binom{s}{k}}{t} p^t$. We get our desired result using the first moment. ∎

Consider a bipartite graph where one set is the set of variables A and the other are clauses in $F^*[A]$. By the lemma we proved above, this graph contains a $\lfloor \frac{2k}{3} \rfloor$-fold matching M from the set of clauses to the set A. M connects each clause in $F{*}[A]$ to exactly $\lfloor \frac{2k}{3} \rfloor$ variables, and each variable occurs in at most one edge of M.

We will now show a way to create an assignment σ. For all variables $v \in A$ that are incident with an edge $\{C, v\}$ of M,

$$\sigma(v) = \begin{cases} \text{TRUE}, & \text{if } v \text{ occurs positively in } C \\ \text{FALSE}, & \text{otherwise} \end{cases}$$

An implication from the above assignment scheme is that since each clause connects with at least $\lfloor \frac{2k}{3} \rfloor > \frac{k}{2}$ variables, σ satisfies more than $\frac{k}{2}$ literals in each clause of $F * [A]$. This also implies that during each execution of `Walksat`, the probability that the hamming distance between `Walksat`'s current assignment and σ (a satisfying assignment) decreases is strictly more than half. Since we view this as a markov chain, where each we can state i where i is the hamming distance between σ and our current assignment. We move from i to $i+1$ with probability probability $> \frac{1}{2}$ while the probability of staying in state i or going to $i-1$ is less than a half. Using the theory of random walks, we know that the time to hit state 0 i.e., reach σ, a satisfying assignment is at most $O(|A|) = O\left(\frac{n}{k}\right)$ with high probability.

Hence, this shows that we can reach a satisfying assignment after performing at most $O\big(\frac{n}{k}\big)$ flips.

3 Remarks

The paper uses a very clever idea to view perturbations as what is done during `Walksat`. This novel idea leads to a much simpler analysis of the `Walksat` algorithm for upto what threshold it can find a threshold in polynomial time. This was also the first paper to analyze the density as a function of k. In this paper, the lower bound up to which it runs in polynomial time is $\frac{2^k}{30k^2}$. Since then, the lower bound has been improved to $\Omega\big(\frac{2k}{k}\big)$ in [4].

Open Problem 0.13 *Is the lower bound $\Omega\big(\frac{2^k}{k}\big)$ tight?*

We must also note that the bounds don't hold when $k = 3$. According to [7], empirical results on random 3CNF formulas indicate that `Walksat` terminates successfully in linear time upto clause density 2.65. The current best rigorous analysis and provable linear time guarantee for 3-CNF is when the clause density is below 1.65 [2].

Open Problem 0.14 *Is there a better upper bound on the density up to which 3-CNF terminates in linear time?*

Suppose we relax the need for linear time, but are okay with polynomial time, we get the following open problem.

Open Problem 0.15 *Given a random 3-CNF formula, can we give provable guarantees that we will find an assignment in $O(n^{f(m,n)})$ where f is a function that relates to the density in some manner.*

We previously defined the *pure literal rule*: Repeat while possible: pick a variable that appears only in one polarity, set it in a satisfying manner, and remove all clauses in which it appears. We can view it as a preprocessing step/optimization before `Walksat` begins, eliminating unnecessary clauses. The following are vaguely defined, more so along the lines of wishful thinking.

Open Problem 0.16 *Are there other preprocessing techniques that can be used to eliminate many clauses from the formula, thereby decreasing the density and leading to finding a satisfying assignment faster using `Walksat`?*

We ask another problem related to the `Walksat` algorithm itself.

Open Problem 0.17 *In `Walksat` algorithm proposed in [8],*

1. *The clauses are picked arbitrarily. Would a deeper analysis possibly indicate that they should be picked according to some priority?*
2. *When a clause is chosen, the variable that is flipped if picked uniformly at random. Should this be the case? Should probability distribution be uniform or should it be non-uniform, dependent to the structure of the formula?*

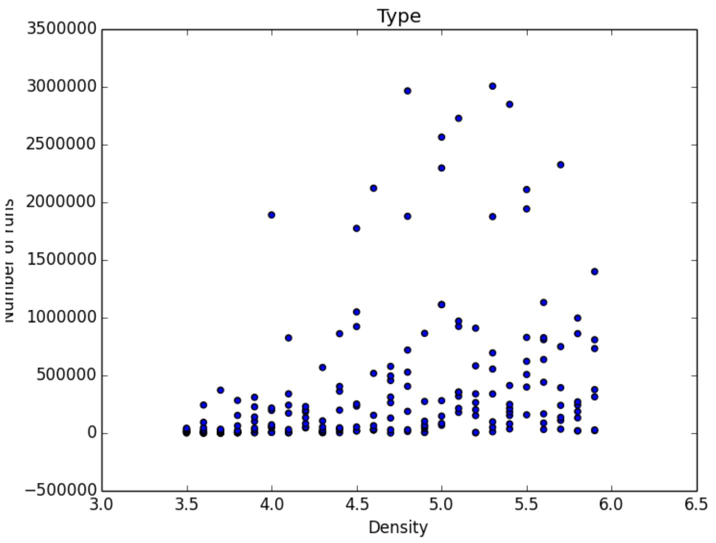
With respect to 3-SAT, empirical results and observation about it will be presented in the next section, mainly trying to answer a question related to whether `Walksat` can terminate in reasonable time and a relation to clause density.

4 Empirical results

All data and code related to this experiment can be found in `https://github.com/adijp/CS761-CNF`. We will start by describing the setup of our experiment. We fix $n = 20$ and $k = 3$. We picked $n = 20$ since it isn't large enough to make it computational infeasible given limited computational resources and not so small that we cannot make claims about our findings. Our goal is to analyze the performance of `Walksat` on different 3-SAT formulas of varying density. Fixing some value of d between 3 and 6, we picked a random 3-SAT formula with that $m = dn$. We used an external SAT solver to verify it has a satisfying assignment. If it does have a satisfying assignment, we count the runtime of `Walksat` until it finds a satisfying assignment.

For our experiment, we used a total of 208 formulas. In our first run, we will plot the amount of time it took with respect to its density, for all 208 formulas. It has been conjectured that the density up to which

Figure 1: All 208 formulas

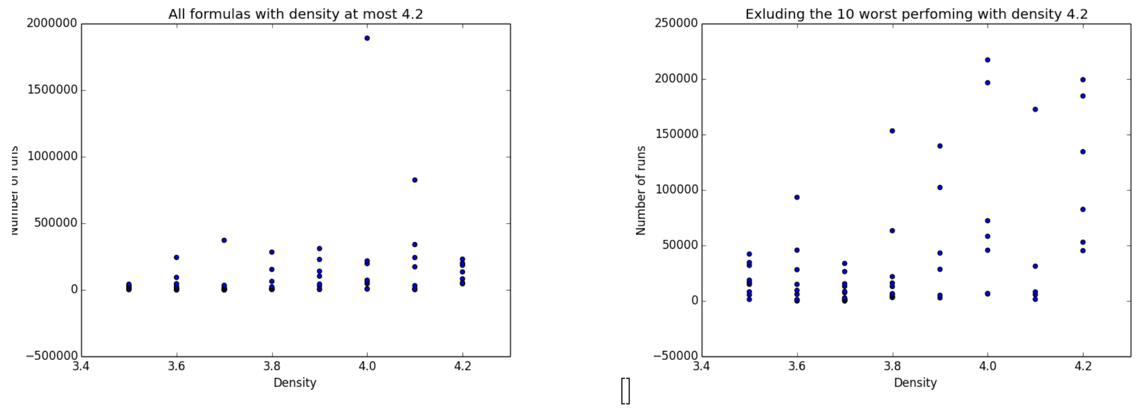


we can get a satisfying assignment is 4.2. We will now plot the instances for which the formulas have density $d \leq 4.2$. There are a total of 75 formulas with clause density at most 4.2. We have plotted this in figure 2. So far, we've observed that most of the worst performing runs are those with higher density exceeding 4.2.

For the rest of our experiment, we have turned our focus exclusively to formulas with density less than 5. Instead of running our formulas multiple times till they find an assignment, we time them out when they exceed a certain number of runs. We have created the timeout as follows,

$$\text{permitted runs} = \begin{cases} 10000, & \text{if } 3 \leq d \leq 3.5 \\ 25000, & \text{if } 3.5 < d \leq 4 \\ 35000, & \text{if } 4 < d \leq 4.5 \\ 65000, & \text{if } 4.5 < d \leq 5 \end{cases}$$

For all formulas that have density less than 5, we have repeated the experiment about 20-30 times. Each formula has a unique formula id from 1 to 208. Although we don't have any knowledge of the formulas individually so far, the plots clearly show that the time out could work, since the worst case instances are

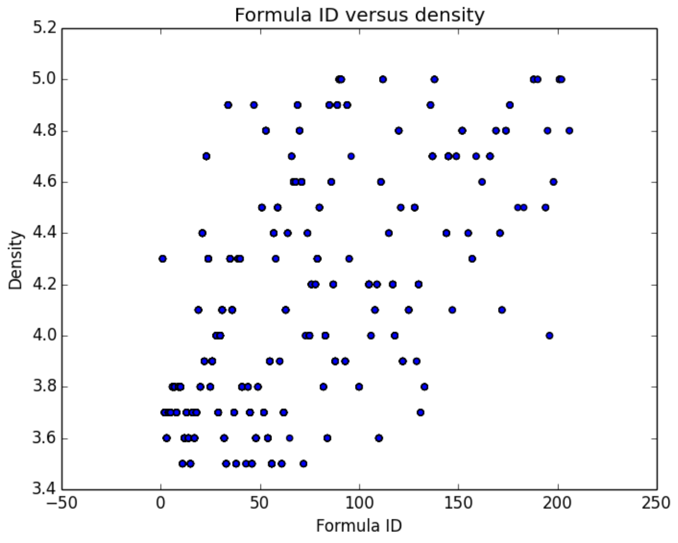


[]

Figure 2: (a) All formulas with density at most 4.2, (b) Density at most 4.2 excluding 25 worst

rare. We will first plot a graph of the formula id and density to give the reader more info about our formulas. The blanks spaces for some formula IDs are when their density is more than 5. They are more prominent

Figure 3: Formula ID vs Density



in plots (c) and (d) in figure 4. Next, we will plot a graph with density vs runs.

Since our primary focus is to look at formulas with density less than 4.2, we will plot it below, but we shall exclude the worst 25 runs out of a total of approximately 800. We wish to exclude the outliers to get an average case analysis of runs instead of just looking at worst case.

There are numerous instances where the density is at most 4.2 where the formulas terminate in reasonable time very often. Perhaps this suggests that although the threshold for linear time as we mentioned before is 1.65, for 3-SAT, we might be able to find a satisfying assignment is we terminate and restart for all formulas

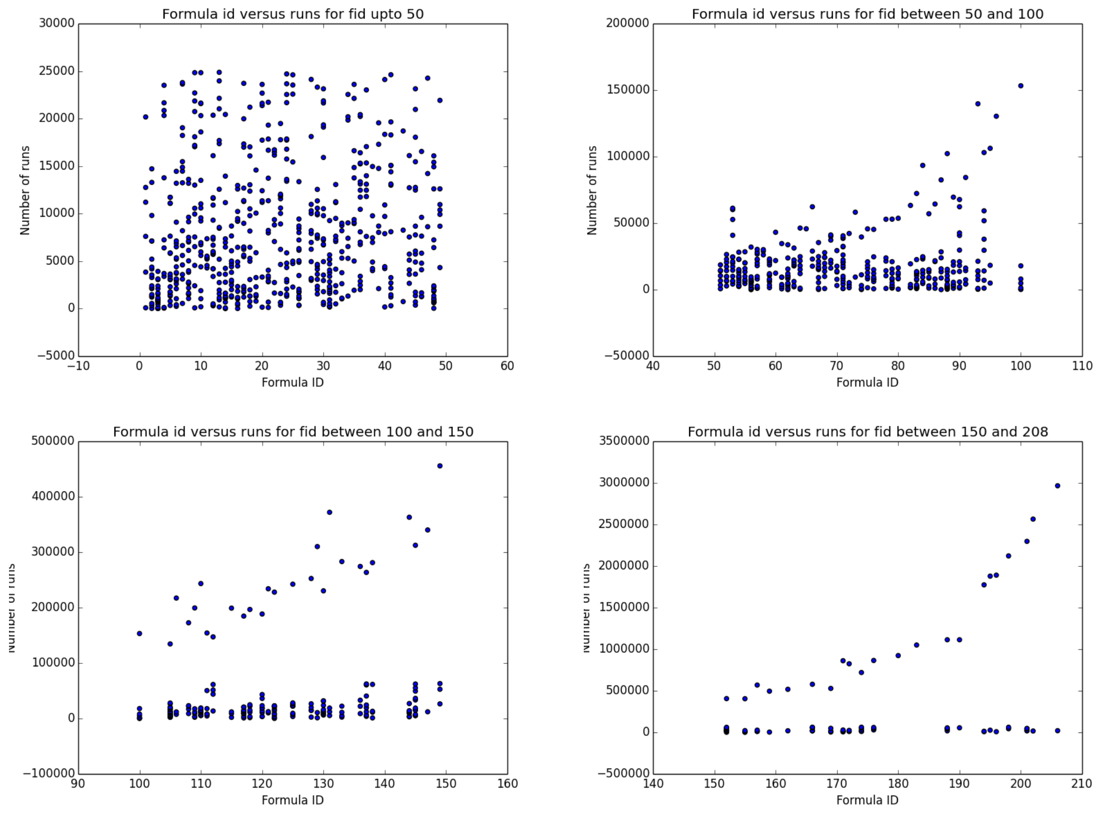


Figure 4: A set of four subfigures: fig:ex3-a Formula ID 0-50; fig:ex3-b Formula ID 50-100; fig:ex3-c Formula ID 100-150; and, fig:ex3-d Formula ID 150-208.

with density at most 4.2 or below the sharp threshold in general. Our approach doesn't always work. In our analysis, we also have 3 formulas (FID: 73,147,196) that have returned a satisfying assignment using the termination rule only once. Formula 65 hasn't returned a satisfying assignment using the termination rule even once, despite running it 30 times. Another property which we haven't included in our analysis, but could yield insight is to consider the number of times a variable appears in a clause.

Open Problem 0.18 *Given a 3-SAT formula F with density $d \leq 4.2$, is it possible to find a satisfying assignment as follows where $t \cdot f(m, n)$ is still poly(n)?*

- *Run `Walksat` and terminate if the number of runs is more than some $f(m, n)$ or a satisfying assignment is found.*
- *Repeat t times or until a satisfying assignment is found.*
- *If no satisfying assignment is found, return "No Assignment".*

Figure 5: Density vs runs

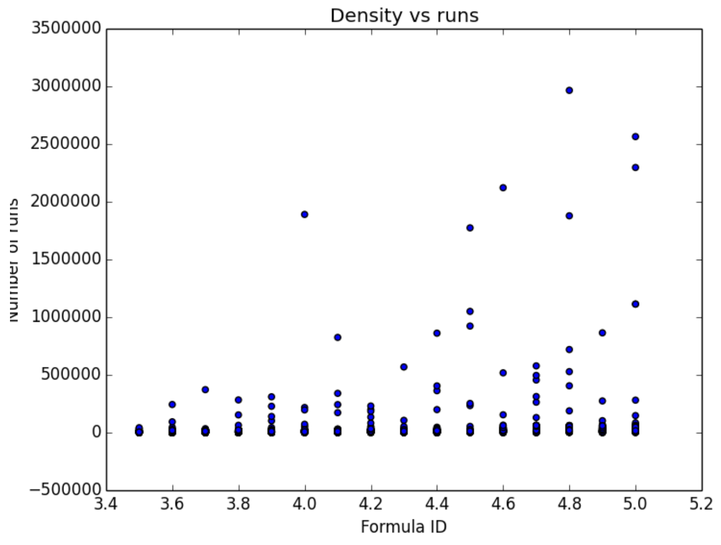
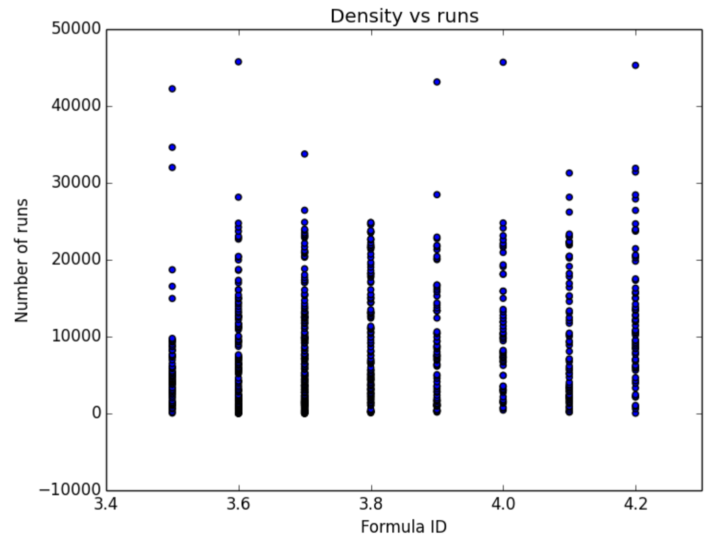


Figure 6: Density ≤ 4.2 vs runs



5 Conclusion

We have introduced and studied numerous properties of k-CNF and looked at models such as smoothed analysis to analyze the performance of `Walksat`. Along with our empirical findings that suggest a modified `Walksat` that could run in $O(poly(n))$, we have also suggested several open problems related to the problem of finding a satisfying assignment for a CNF formula.

References

[1] Dimitris Achlioptas and Yuval Peres. The threshold for random k-sat is 2k (ln 2 - o(k)). In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 223–231, New York, NY, USA, 2003. ACM.

[2] M. Alekhnovich and E. BenSasson. Linear upper bounds for random walk on small density random 3cnfs. *SIAM Journal on Computing*, 36(5):1248–1263, 2007.

[3] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley Publishing, 4th edition, 2016.

[4] A. Coja-Oghlan and A. Frieze. Analyzing walksat on random formulas. *SIAM Journal on Computing*, 43(4):1456–1485, 2014.

[5] Amin Coja-Oghlan, Uriel Feige, Alan Frieze, Michael Krivelevich, and Dan Vilenchik. On smoothed k-cnf formulas and the walksat algorithm. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 451–460, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[6] Robin A. Moser and Gábor Tardos. A constructive proof of the general lovÁsz local lemma. *J. ACM*, 57(2):11:1–11:15, February 2010.

[7] Andrew J. Parkes. Scaling properties of pure random walk on random 3-sat. In *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, pages 708–713, 2002.

[8] T. Schoning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 410–414, Oct 1999.

[9] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, May 2004.