Analysis of the Random Walk Algorithm on Random 3-CNFs

Mikhail Alekhnovich
Laboratory for Computer Science
MIT
Cambridge, MA
mishka@theory.lcs.mit.edu

Eli Ben-Sasson
DEAS
Harvard University
Cambridge, MA
eli@eecs.harvard.edu

April 25, 2002

Abstract

We analyze the efficiency of the random walk algorithm on random 3-CNF instances, and prove the first polynomial time upper bound for small clause density, less than 1.63. We complement this by proving exponential lower bounds for the running time of this algorithm on the planted-SAT distribution with large constant clause density. This is the first polynomial upper bound on the running time of a *local improvement* algorithm on random instances, and conforms with the empirically observed efficiency of these algorithms on random CNFs.

1 Introduction

Understanding the complexity of SAT-solving algorithms on various natural instances is a fundamental problem of computer science. Our current research analyzes the running time of one of the simplest SAT solving algorithms on one of the simplest instance distributions. Our motivation is to understand an interesting observed phenomena, which is the efficient performance of the *random walk* algorithm on random 3-CNFs, for small clause density[1], well below the satisfiability threshold. Random 3-CNFs are extremely interesting and have been the focus of much recent research. The algorithm we investigate is one of the simplest SAT solving heuristics, and is used heavily in practice [29]. From a theoretical perspective, several lower and upper bounds are known for this algorithm, all of them exponential (see e.g. [24] problem 11.5.6 for lower bounds, and [33] for upper bounds). None of these explains why `RWalkSAT` performs well on random instances. Our paper tries to offer such an explanation.

We then proceed to check whether the efficiency of the algorithm holds also for large clause densities, well over the satisfiability threshold. In this case, we examine the random planted SAT distribution, that assures each random formula to be satisfiable. For this case, we show `RWalkSAT` is inefficient, and typically requires exponential time to find a satisfying assignment.

1.1 3-SAT and Random Walk

3-SAT is one of the most famous NP-complete problems [11]. An input to this problem is a 3-CNF, and we are asked to decide whether there exists an assignment to the variables, that satisfies all clauses, i.e. sets it to 1. Assuming $P \neq NP$, no efficient algorithm exits for this problem. Still, over the years, many different solutions have been proposed and investigated, both theoretically and empirically (see [12] for a survey of related results). Most algorithms are *local*, concentrating only on a small part of the formula, usually a constant number of clauses or variables, and making a *local assignment* to a few variables, based on this information, and hoping that all local decisions will eventually produce a satisfying assignment.

One particularly simple and elegant local method, starts with an assignment to all variables, and then, as long as the assignment is not good, an unsatisfied clause C is selected, and the assignment to one of its literals is changed. This creates a new assignment, that indeed satisfies C, but may "ruin" the situation in other clauses. We repeat this procedure, until hopefully all clauses are satisfied. Since the CNF may be unsatisfiable, we usually put a time limit on the number of iterations, and once the limit has been reached, we give up.

The simplest variant of this local approach, originally introduced by [23], is at the center of our current investigation. This algorithm, denoted `RWalkSAT`, relies on randomness in all its decisions. Thus, the initial assignment is picked at *random*, at each step a *random* unsatisfied clause is picked, and this clause is corrected by flipping a *random* literal in the clause.

Surprisingly, `RWalkSAT` behaves much better than one might expect. [23] proved that it succeeds in finding a satisfying assignment in expected quadratic time on *any* satisfiable 2-CNF. Although 2-SAT is solvable in polynomial time, it is not obvious that such a simple local heuristic as `RWalkSAT` is good enough to solve it. Another interesting result is that on any satisfiable 3-CNF, the expected running time of `RWalkSAT` is substantially less than 2^n (where n is the number of variables) [33]. We point out that similar to the case of other 3-SAT algorithms, it is not hard to find specific instances on which `RWalkSAT` requires exponential running time (see e.g. [24], problem 11.5.6).

[1] The clause density is the ratio between the number of clauses and number of variables.

1.2 Random CNFs

A random 3-CNF with clause density Δ and n variables, is a 3-CNF formula obtained by selecting Δn random clauses, each of size 3, from the set of all $2^3 \cdot \binom{n}{3}$ possible clauses. This distribution has received much interest in recent years, being a natural distribution on NP-complete instances that seems (empirically as well as theoretically) computationally hard for a wide range of parameters. This distribution is investigated in such diverse fields as physics [19], combinatorics [16], proof complexity [10], algorithm analysis [3] and hardness of approximation [14], to mention just a few.

One of the basic properties of random CNFs, is that for small density ($\Delta < 3.22...$ see [3]) almost all formulas are satisfiable, whereas for large density ($\Delta > 4.5...$ see [17]) they are almost all unsatisfiable. Another interesting property is that the threshold between satisfiability and unsatisfiability is sharp [16]. It is conjectured that a *threshold constant* exists, and empirical experiments estimate it to be ≈ 4.26 [13]. The settling of the satisfiability threshold conjecture, and determination of the satisfiability threshold constant (if it exists) are important open problems in combinatorics (see [3] for more information).

An interesting algorithmic question, is the performance of SAT solving methods on random CNFs. This question has been extensively researched empirically, and random CNFs are commonly used as test cases for analysis and comparison of SAT solvers. From a theoretical point of view, several lower and upper bounds were given for DLL type algorithms under this distribution [3, 8, 21]. Upper bounds for algorithms imply lower bounds on the satisfiability threshold, and in fact, all lower bounds on the threshold so far, have come from analyzing specific SAT solving algorithms. Most of the algorithms for which average case analysis has been applied so far are classified as *myopic* algorithms [3], and much less is known about non-myopic algorithms. We hope our paper will increase the understanding of some of these non-myopic algorithms, at least on random instances.

We point out that the lack of analysis for `RWalkSAT` is not for lack of interest, but rather because of the inherent difficulty in approaching this problem. Myopic algorithms owe their name to the fact that they view only a small portion of the formula before assigning a variable, and more importantly, never backtrack. This allows one to analyze these algorithms assuming the unobserved formula is still random. `RWalkSAT` is very far from being myopic. Even for 2-CNFs, the expected running time is quadratic, and thus many variables will be reassigned different values before the satisfying assignment is found. Thus, when analyzing this algorithm, new techniques are required.

1.3 The Observed Efficiency of `RWalkSAT`

For a long period of time, the SAT-solving "market" was dominated by DLL-type algorithms. These algorithms, that are still extremely popular, operate by fixing a value to one variable at a time, and recursively solving the SAT problem on the restricted formula, using backtracking when a contradiction is reached. In the 1990's, new players entered the scene, in the form of *local improvement* algorithms, of which `RWalkSAT` is the simplest variant [31]. These algorithms (the most famous of which are WalkSAT and GSAT) were found to compete successfully, and many times outperform the best DLL-type algorithms. Today, the two families of algorithms are recognized to have their advantages and limitations, and a final verdict on the "best" SAT-solver has not been reached (in fact, there are annual SAT-solving contests).

One particularly striking example on which `RWalkSAT` and other local improvement perform extremely well, is that of random CNF [28]. For instance, based on experiments using random CNFs with up to $\approx 100,000$ variables, `RWalkSAT` appears to require linear time on random CNFs with clause density ≤ 2.6 [26]. More advanced algorithms such as WalkSAT and GSAT (that use `RWalkSAT` as a subroutine), appear empirically to solve random instances with clause density ≤ 4, in quadratic time, and there is data indicating polynomial running time up to density ≤ 4.2 (recall that the empirical threshold is ≈ 4.26) [27]. This performance is well above the largest theoretical lower bound on the satisfiability threshold (which is

around 3.26 [3]), and even above the best conjectured density for which DLL algorithms are efficient [2]. The observed efficiency of these algorithms suggests that their analysis may push the satisfiability threshold constant above what is currently known, and our results can be viewed as an initial step in this direction.

1.4 Results and Proof Techniques

Our main results are a polynomial upper bound on the running time of `RWalkSAT` for small clause density, and an exponential lower bound for large clause density.

1.4.1 Upper Bound

We show that for any clause density below 1.63, `RWalkSAT` succeeds with high probability in finding a satisfying assignment for a random CNF with n variables and Δn clauses, in $n^{O(1)}$ time (theorem 3.6).

Notice that there are two different notions of randomness in our statement. The first is the randomness in picking $\mathcal{C}$. Once $\mathcal{C}$ is fixed, we have the randomness in the choices of `RWalkSAT`[2].

Since all our claims will deal with this dual randomness, let us clarify our meaning. Let us say that $\mathcal{C}$ is *easy* for `RWalkSAT` if it succeeds with high probability (whp) in finding a satisfying assignment for $\mathcal{C}$ in time $n^{O(1)}$. We claim that for $\Delta \leq 1.63$ all but a negligible fraction of CNFs with Δn clauses are easy for `RWalkSAT`. This theoretical claim coincides with the observed efficiency of `RWalkSAT`. We point out that there is still a large gap between the observed and explained. Both the observed density for which `RWalkSAT` is efficient, and the observed running time, are much better that what we prove. Still, even a thousand mile walk starts with a single step [1].

A word about our proof techniques. We heavily rely on the work of [8], and show that for such small clause density, a random CNF has an "onion-like" structure. It can be partitioned into $O(\log n)$ *layers*, such that each layer, itself a CNF, can be easily satisfied by `RWalkSAT`, assuming all outer layers are already satisfied. The running time of `RWalkSAT` for such an onion-like formula is exponential in the number of layers. This immediately leads to an upper bound of $n^{O(\log n)}$. We then look more closely at the innermost layers of the onion, and show that they can be combined into one layer, on which `RWalkSAT` succeeds quickly. This reduces the number of layers from $O(\log n)$ to $O(1)$, giving us the polynomial upper bound.

We end our discussion by showing the optimality of our technique, and presenting onion-like formulas on which the running time of `RWalkSAT` is exponential in the number of layers.

1.4.2 Lower Bound

`RWalkSAT` has a one-sided error, and on any unsatisfiable formula always gives the right answer. Hence it is only interesting to investigate the behavior of `RWalkSAT` on satisfiable formulas. For clause density greater than $4.5...$ almost all formulas are unsatisfiable. One would like to analyze the behavior of `RWalkSAT` on the satisfiable portion of these random formulas, alas we have no good characterization of them. Thus, we investigate the closely related *planted-SAT* distribution. In this distribution, one starts with a random assignment, and makes sure that the formula obtained satisfies at least this assignment. This distribution has been studied empirically [9] and theoretically [6], and is closely related to other *planted-NP* instances such as *planted-clique* and *planted-bisection*, that have received much attention in recent years [4, 15, 18].

For this distribution we have negative results. We show that for large enough constant density, almost all formulas are hard for `RWalkSAT`, and require exponential time until a satisfying assignment is found (theorem 4.2).

[2] This dual role played by randomness is well studied in statistical mechanics models, e.g. in the study of the *spin glass* model and its variants

In order to prove this, we first investigate the *full CNF*, comprising of all clauses that are satisfied by the planted assignment. We show that RWalkSAT performs poorly on this CNF. We then use standard concentration theorems to show that the behavior on the random CNF is similar to the behavior on the full CNF.

2 Preliminaries

Some general definitions are deferred to appendix A. Our upper bound is for the following distribution.

Definition 2.1 *Let $\mathbb{F}^n_\Delta$ be the probability distribution obtained by selecting Δn clauses uniformly at random from the set of all $8 \cdot \binom{n}{3}$ clauses of size 3 over n variables. $\mathcal{C} \sim \mathbb{F}^n_\Delta$ means that $\mathcal{C}$ is selected at random from this distribution. We call such a $\mathcal{C}$ a* random 3-CNF

RWalkSAT operates as follows. Given as input a CNF over n variables, and a time bound t, it starts by selecting a random initial assignment $\alpha_0 \in \{0,1\}^n$. Then, at each time step t it checks whether $\mathcal{C}$ is satisfied by α_t. If α_i satisfies $\mathcal{C}$, the algorithm *succeeds.* If this is not the case, a random clause C is selected from the set of all clauses *not satisfied* by α_t. α_t is then locally "corrected" as to satisfy C. This is done by selecting at random a literal appearing in C, and flipping the assignment of α_t on this literal. The new assignment is denoted α_{t+1} and the process is repeated, T times. If after T times the algorithm has not found a satisfying assignment, it gives up and declares its *failure*. For the following algorithm, we use the notation $UNSAT(\mathcal{C}, \alpha)$ for the set of clauses of $\mathcal{C}$ that are unsatisfied by α. Pseudo-code of this algorithm is given in appendix B.

Throughout the paper, we use the notation α_t to denote the assignment of RWalkSAT at time t. α_0 is called the *initial assignment.* We also denote by RWalkSAT$(\mathcal{C}, T)$ a single execution of RWalkSAT on the CNF $\mathcal{C}$, given time bound T on the number of execution steps.

3 Upper Bounds for Small Density

In this section we prove that with high probability, RWalkSAT finds in polynomial time a satisfying assignment for a random 3-CNF with low enough clause density, $\Delta < 1.63$ (theorem 3.6). Our proof shows that at low clause density, a random CNF can be partitioned into a *constant* number of layers, such that each layer is easily satisfied by RWalkSAT, if the higher layers have already been satisfied. The expected running time is exponential in the number of layers, and so we get our polynomial upper bound.

At such low clause density, even the simple pure literal heuristic finds with high probability a satisfying assignment [8, 21]. Indeed, our proof uses heavily elements of the original proof for the pure literal heuristic, given by [8], and all the "layers" but the last are sets of clauses removed by a single application of the pure literal rule.

3.1 The Pure Literal Rule

A literal ℓ in $\mathcal{C}$ is called *pure* if it appears only as a positive literal, or only as a negative literal, in $\mathcal{C}$. A clause in $\mathcal{C}$ is said to be *pure* if it contains a pure literal. When seeking a satisfying assignment, a natural strategy is to start by assigning all pure literals their satisfying assignment, and thus remove all pure clauses. The removal of pure clauses may result in the emergence on new pure literals in the restricted CNF, and the process may be repeated. The *pure literal heuristic* is the heuristic that applies this removal process until no pure clauses remain. If the remaining CNF is empty, the pure literal heuristic has found a satisfying assignment, and otherwise it failed.

Let us introduce some notation. For $\mathcal{C}$ a CNF, define $\mathcal{C}_0 = \mathcal{C}$, L_0 to be the set of pure literals in $\mathcal{C}$, and P_0 to be the set of pure clauses in $\mathcal{C}$. Recursively define $\mathcal{C}_{i+1}$ to be $\mathcal{C}_i \setminus P_i$, and let L_{i+1}, P_{i+1} be respectively the set of pure literals, and pure clauses, in $\mathcal{C}_{i+1}$. Finally, let r be the minimal i such that $L_i = \emptyset$. Notice that the pure literal succeeds on $\mathcal{C}$ iff $\mathcal{C}_r = \emptyset$. If $\mathcal{C}_r = \emptyset$ we say $\mathcal{C}$ is *r-pure*.

[8] showed that with high probability the pure literal heuristic finds a satisfying assignment for clause density < 1.63. In particular, they implicitly proved the following theorem.

Theorem 3.1 *[8] For every $\Delta < 1.63$ there exists an integer d such that with high probability for $\mathcal{C} \sim \mathbb{F}^n_\Delta$, $|\mathcal{C}_d| \leq \frac{n}{600\Delta^2}$.*

3.2 Efficiency of `RWalkSAT` on Pure CNFs

We claim that the running time of `RWalkSAT` on an r-pure formula with m clauses, is with high probability $O(m^r)$. This will immediately give quasi-polynomial upper bounds on the running time for random CNFs. For brevity, the proof of the following theorem is deferred to appendix C.

Theorem 3.2 *For all $r \geq 1$, if $\mathcal{C}$ is an r-pure 3-CNF formula over m clauses, then for any initial assignment α,*

$$\mathbf{P}[\texttt{RWalkSAT}(\mathcal{C}, (6 \cdot m)^r) \textit{ fails} \mid \alpha_0 = \alpha] < (7m)^r \cdot e^{-m/4}$$

3.3 Quasi-Polynomial Upper Bounds

Theorem 3.2 immediately implies quasi-polynomial upper bounds for random CNFs, with low clause density (less than 1.63). This fact follows from the next theorem.

Theorem 3.3 *For every $\Delta < 1.63$ there exists a constant c such that with high probability $\mathcal{C} \sim \mathbb{F}^n_\Delta$ is $(c \cdot \log n)$-pure.*

Proof: We need a standard definition and claim from the context of random CNFs (see e.g. [10]).

Definition 3.4 *For $\mathcal{C}$ a CNF, we say $\mathcal{C}$ is an* (r, c)-expander *if for all $\mathcal{C}' \subset \mathcal{C}$ $|\mathcal{C}'| \leq r$, $|Vars(\mathcal{C}')| \geq c \cdot |\mathcal{C}'|$.*

For $\mathcal{C}' \subseteq \mathcal{C}$, the boundary *of $\mathcal{C}'$, denoted $\partial(\mathcal{C}')$, is the set of variables that appear in exactly one clause of $\mathcal{C}'$. $\mathcal{C}$ is called an* (r, c)-boundary expander *if for all $\mathcal{C}' \subset \mathcal{C}$ $|\mathcal{C}'| \leq r$, $|\partial(\mathcal{C}')| \geq c \cdot |\mathcal{C}'|$.*

We use the fact that a random CNF is a decent boundary expander, as given by the following lemma, the proof of which is deferred to appendix E.

Lemma 3.5 *Let $\Delta_0 = 1.63$. For any constant $\Delta \leq \Delta_0$ whp $\mathcal{C} \sim \mathbb{F}^n_\Delta$ is a $(\frac{n}{600\Delta_0^2}, 10^{-3})$-boundary expander.*

We now complete the proof. By theorem 3.1, there exists a constant d such that whp $\mathcal{C}_d$ is a 3-CNF with $\leq n/(600\Delta^2)$ clauses. By lemma 3.5 we also know that whp $\mathcal{C}_d$ is a $(|\mathcal{C}_d|, 10^{-3})$-boundary expander. Notice that the boundary of $\mathcal{C}_d$ is a set of pure literals, since each variable in the boundary appears in a single clause of $\mathcal{C}_d$. There are linearly many pure literals, and each clause can contain at most 3 pure literals. Thus, there exists a linear fraction of clauses in $\mathcal{C}_d$ that are pure. Removing these clauses, we apply lemma 3.5 again to $\mathcal{C}_{d+1}$ and remove another linear fraction of clauses. After $O(\log n)$ repetitions we have removed all clauses from $\mathcal{C}$, and hence $\mathcal{C}$ is an $O(\log n)$-pure CNF. $\square$

3.4 Polynomial Upper Bounds

Theorem 3.6 (Main Upper Bound) *For any $\Delta < 1.63$ there exists a constant d such that whp for $\mathcal{C} \sim \mathbb{F}^n_\Delta$,*

$$\mathbf{P}[\texttt{RWalkSAT}(\mathcal{C}, m^d) \textit{ fails}] = \exp(-\Omega(n))$$

Proof: First the proof intuition. Theorem 3.3 shows that $\mathcal{C}$ is $O(\log n)$-pure. We take the innermost layers of $\mathcal{C}$, and view them as a single layer. Although this layer is not a 1-pure CNF, we use it's large expansion to claim that RWalkSAT terminates quickly on it, given any initial assignment.

Formally, we use the following lemma and theorem. The proof of the lemma is deferred to appendix E whereas the proof of the theorem will promptly follow.

Lemma 3.7 *For any $\Delta < 1.63$ and any constant $c < 2$ there exists an integer d such that whp for $\mathcal{C} \sim \mathbb{F}^n_\Delta$, $\mathcal{C}_d$ is a $(|\mathcal{C}_d|, c)$-expander.*

Theorem 3.8 *If $\mathcal{C}$ is an $(|\mathcal{C}|, \frac{7}{4})$-expander, then for any initial assignment α,*

$$\mathbf{P}[\texttt{RWalkSAT}\ (\mathcal{C}, m^3) \textit{fails} \mid \alpha_0 = \alpha] \leq e^{-\Omega(m)}.$$

For $c = 7/4$, let d be the constant promised by lemma 3.7. $\mathcal{C}_d$ is whp a $(|\mathcal{C}_d|, 7/4)$-expanding CNF. We now proceed as in the proof of theorem 3.2.

Let $\mathcal{C}' \stackrel{\text{def}}{=} \mathcal{C} \setminus \mathcal{C}_d$. Notice that $\mathcal{C}'$ is by definition a d-pure formula. Split the execution of $\texttt{RWalkSAT}(\mathcal{C})$ into *epochs*, where each epoch starts when a clause from $\mathcal{C}'$ is considered. By theorem 3.2 and its proof, the number of epochs is whp $(6m)^d$. By theorem 3.8, $\mathcal{C}_d$ is whp an $(|\mathcal{C}_d|, 7/4)$-expander, and so by theorem 3.8 each epoch terminates in cubic time. The theorem follows by a union bound as in (11)-(13). □

The rest of this subsection is devoted to the proof of theorem 3.8. First a few words of intuition. For a highly expanding CNF, there is an easy way to find a satisfying assignment, namely by finding a matching from clauses to variables and using a unique variable per clause for the satisfying assignment. This still does not mean that RWalkSAT will find such an assignment in short time. Notice that if a CNF is 2-expanding, then there exists an assignment α' that satisfies 2 literals per clause, and thus RWalkSAT will terminate in polynomial time (because for each unsatisfied clause there is a probability of at least $2/3$ of heading in the direction of α').

Unfortunately, lemma 3.7 only gives us $(2 - \epsilon)$ expansion, and thus we need a new technique to show that RWalkSAT succeeds on such an instance. We show that for highly expanding CNFs, there exists a *restriction* ρ that satisfies the CNF, and at each step there is a good probability of heading toward ρ.

Proof [Theorem 3.8]: For $\mathcal{C}$ a CNF, a *restriction* ρ to $\mathcal{C}$ is an assignment to $V \subseteq Vars(\mathcal{C})$, i.e. $\rho \in \{0,1\}^V$, and V is called the *support* of ρ. We say ρ satisfies $\mathcal{C}$, if for every clause $C \in \mathcal{C}$ there is some literal $\ell \in C$ that is satisfied by ρ. For C a clause, the *disagreement* of ρ and C is the number of literals in C that are fixed by ρ (i.e. belong to its support) and do not satisfy C. Notice that if ρ is a satisfying restriction, then the disagreement of C is at most 2. Let the *maximal disagreement* of ρ be the maximum of the disagreement of a clause in $\mathcal{C}$ with ρ.

Lemma 3.9 *If $\mathcal{C}$ is an $(m, \frac{7}{4})$- expander over m clauses, then there exists a satisfying restriction ρ with maximal disagreement* 1.

Lemma 3.10 *If $\mathcal{C}$ is a 3-CNF over n variables, having a satisfying restriction with maximal disagreement* 1*, then*

$$\mathbf{P}[\texttt{RWalkSAT}(\mathcal{C}, n^3) \textit{ fails}] \leq \exp(-\Omega(n))$$

Theorem 3.8 follows immediately from the previous lemmas. □

Proof [Lemma 3.9]: Form the following bipartite graph G. On the left hand, put one vertex for each clause in $\mathcal{C}$. On the right hand side, put 4 distinct vertices for each variable appearing in $\mathcal{C}$. Connect the vertex labeled by the clause C to all 12 vertices labeled by variables appearing in C. We do not care if the appearance is as positive or negative literals.

Since $\mathcal{C}$ is an $(m, \frac{7}{4})$-expander, G has expansion factor 7, i.e. for all subsets S on the left hand side, $|N(S)| \geq 7 \cdot |S|$, where $N(S)$ is the set of neighbors of S. By Hall's matching theorem we conclude there is an 7-matching from the left hand side to the right, i.e. each node C on the left hand can be associated with a set of 7 of its neighbors on the right hand side (denoted $N'(C)$), such that for all clauses $C \neq D$, $N'(C) \cap N'(D) = \emptyset$. We now use N' to define ρ. For any variable x, if there exists a clause C such that $N'(C)$ has at least 3 members labeled by x, then set $\rho(x)$ to the value that satisfies C, and otherwise, leave $\rho(x)$ unassigned. Only 4 vertices are labeled by x, and $N'(C) \cap N'(D) = \emptyset$ for all clause $C \neq D$, so ρ is well defined. The lemma follows from the following pair of claims.

Claim 3.11 *For any clause C, ρ satisfies C.*

Proof: C has only three variables, whereas $|N'(C)| = 7$. By the pigeonhole principle some variable x appearing in C must have three representatives in $N'(C)$, and by the definition of ρ, C is satisfied by ρ's assignment x. □

Claim 3.12 *For any clause C, ρ can disagree with C on at most one literal.*

Proof: Assume for the sake of contradiction that ρ disagrees with C on two variables. From the definition of ρ, these two variables have at least three of their representatives in some $N'(D), N'(E)$ for some clauses $D, E \neq C$. Since $N'(C)$ has no intersection with $N'(D), N'(E)$, we conclude $|N'(C)| \leq 12 - 6 = 6$. But this contradicts the fact that $|N'(C)| = 7$. □

Lemma 3.9 follows. □

Proof [Lemma 3.10]: For $\alpha \in \{0,1\}^n$ and ρ a restriction, let $\Delta_H(\alpha, \rho)$ be the Hamming distance between ρ and α, measured only on the support of ρ (the support is the set of variables fixed by ρ). For $t \leq m^2$, let $d_t = \Delta_H(\alpha_t, \rho)$. Let $X_t = d_{t+1} - d_t$ be the random variable indicating whether the t'th step of the execution was good ($X_t = 1$, and α_{t+1} is closer to ρ), bad (X_t=(-1)), or neutral ($X_t = 0$). Notice that since ρ has maximal disagreement 1, we get that for all t, the probability of a neutral step is at most $2/3$, and conditioned on a step being non-neutral, the probability of the step being good, is at least $1/2$.

Standard arguments from the theory of random walks show that the probability of failure in time $O(m^3)$ is exponentially small (see e.g. [24], chapter 11, pp. 245-247, for similar analysis). The lemma is proved. □

3.5 Tightness of Bounds

Deferred to appendix D

4 Lower Bounds for Large Density

In this section, we show that `RWalkSAT` is not a good algorithm for random CNFs with large clause density. By definition, `RWalkSAT` gives the correct answer on any unsatisfiable formula. For large enough clause

density ($\Delta > 4.6$), almost all formulas in $\mathbb{F}_\Delta^n$ are unsatisfiable [17]. Thus, one may argue that RWalkSAT operates very well for these densities. On second thought, on this distribution, even the constant time algorithm that fails on every input, without reading it, operates well. Thus, it makes sense to discuss the performance of RWalkSAT only on the uniform distribution over *satisfiable* formulas with Δn clauses (denoted $\mathbf{SAT}_n^\Delta$). The problem is that for small densities, $\mathbf{SAT}_n^\Delta$ is not well characterized, we don't know how to analyze it. We remark that for $\Delta = \Omega(\log n)$, $\mathbf{SAT}_n^\Delta$ can be characterized (see [6] for more details).

Thus, following [6], we propose to look at the following pair of *planted SAT* distributions over satisfiable 3-CNFs. This distribution is highly interesting in its own right. It has been studied empirically in [9], and is the analog of the *planted clique* and *planted bisection* distributions, studied e.g. in [4, 15, 18].

Definition 4.1 (Planted SAT) *Let $\mathbb{S}_\Delta^n$ be the distribution obtained by selecting at random $\beta \in \{0,1\}^n$, and selecting at random Δn clauses out of all clauses of size 3 that are satisfied by β. Denote a random formula from this distribution by $\mathcal{C} \sim \mathbb{S}_\Delta^n$.*

Let $\mathbb{P}_\Delta^n$ be the distribution obtained by selecting at random $\beta \in \{0,1\}^n$, and for each clause C satisfied by β, select C to be in $\mathcal{C}$ with independent probability $p_n^\Delta = \frac{6\Delta}{7(n-1)(n-2)}$. Denote a random formula from this distribution by $\mathcal{C} \sim \mathbb{P}_\Delta^n$.

Theorem 4.2 (Main Lower Bound) *There exists a constant $\Delta_0 > 0$, such that for all $\Delta \geq \Delta_0$, (Δ may be a function of n), whp for $\mathcal{C} \sim \mathbb{P}_\Delta^n$, or $\mathcal{C} \sim \mathbb{S}_\Delta^n$*

$$\mathbf{P}[\texttt{RWalkSAT}(\mathcal{C}, 2^{\epsilon n}) \textit{ succeeds}] \leq 2^{-\epsilon n}$$

where $\epsilon > 0$ is some a constant, depending on Δ.

For simplicity, our proof will be only for the distribution $\mathbb{P}_\Delta^n$, and we point out that the same analysis can be carried over to the distribution $\mathbb{S}_\Delta^n$.

The rest of this section is devoted to the proof of theorem 4.2. We warm up by discussing the case of $\mathcal{C}$ being the maximal size CNF satisfying β, and then apply our insights to the case of a random CNF. For the rest of this section we assume without loss of generality that β, the random planted assignment, is the all zero vector, denoted $\vec{0}$.

The full CNF of size n has all clauses of size 3 that are satisfied by $\vec{0}$. In the next subsection we analyze the behavior of RWalkSAT on this CNF, and show its inefficiency. We then generalize our analysis to $\mathcal{C} \sim \mathbb{P}_\Delta^n$, claiming that $\mathcal{C}$ is a "random piece" of the full CNF, and hence the behavior of RWalkSAT on $\mathcal{C}$ will be close to its behavior on this simple formula.

4.1 Analysis of RWalkSAT on the Full CNF

Definition 4.3 (Full CNF) *$\mathcal{F}_n$ is the 3-CNF containing all clauses of size exactly 3 (without repetition of literals) satisfying $\vec{0}$.*

Theorem 4.4 $\mathbf{P}[\texttt{RWalkSAT}(\mathcal{F}_n, 2^{n/100}) \textit{ succeeds}] \leq 2^{-n/100}$

For the proof of theorem, we need the following lemma, the proof of which is deferred to appendix E. The *line* of length n is the graph $G =< V, E >$ where $V = \{0, 1, \ldots, n\}$ and $E = \{(i, i+1) : 0 \leq i < n\}$.

Lemma 4.5 *Let R be a random walk on the line of length n, starting at a position $b' \geq b$. Assume there is an interval $0 \leq a < b$ such that for all vertices $i \in \{a, a+1, \ldots, b\}$ the probability of making a move in the direction of 0 when standing on vertex i, is at most $1/2 - \gamma$, for some constant $\gamma > 0$. Then*

$$\mathbf{P}[R \textit{ reaches } 0 \textit{ in less than } e^{\gamma(b-a)} \textit{ steps }] \leq e^{-\gamma(b-a)}$$

Proof [Theorem 4.4]: For $i = 1 \dots 3$, let $\mathcal{F}_n^i$ be the set of clauses having i literals satisfied under the unique satisfying assignment $\vec{0}$. For $\alpha \in \{0,1\}^n$, $|\alpha| = \delta n$, let $m_{\delta,i}$ be the number of clauses in $\mathcal{F}_n^i$ that are not satisfied by α. $m_{\delta,i}$ is well defined, because it depends only on δ, and not on the actual assignment α. $m_{\delta,1}$ is the set of all clauses having exactly one negative literal, such that this literal is assigned 1 by α, and the remaining positive literals are assigned 0 by α. Since $|\alpha| = \delta n$, there are $\delta n \cdot \binom{(1-\delta)n}{2}$ such clauses. analogous calculations performed for $m_{\delta,2}, m_{\delta,3}$ yield:

$$m_{\delta,1} = \delta n \cdot \binom{(1-\delta)n}{2}; \qquad m_{\delta,2} = \binom{\delta n}{2} \cdot (1-\delta)n; \qquad m_{\delta,3} = \binom{\delta n}{3}; \tag{1}$$

Since $\mathcal{F}_n$ has only one satisfying assignment, one can characterize the moves of $\texttt{RWalkSAT}(\mathcal{F}_n)$ as either *good*, or *bad*. A *good move* is one that reduces the weight of α_t, bringing us closer to the satisfying assignment, whereas a *bad move* increases this weight. We now calculate the probability of making a good move on $\mathcal{F}_n$, assuming the current assignment has weight δn. Let us denote this probability by P_δ, and let $m_\delta = m_{\delta,1} + m_{\delta,2} + m_{\delta,3}$.

$$P_\delta = \frac{m_{\delta,3}}{m_\delta} + \frac{m_{\delta,2}}{m_\delta} \cdot \frac{2}{3} + \frac{m_{\delta,1}}{m_\delta} \cdot \frac{1}{3} \approx \frac{3\frac{\delta^3}{6} + 2\frac{\delta^2}{2}(1-\delta) + \delta\frac{(1-\delta)^2}{2}}{3\left(\frac{\delta^3}{6} + \frac{\delta^2(1-\delta)}{2} + \frac{\delta(1-\delta)^2}{2}\right)} \tag{2}$$

$$= \frac{\delta^2 + 2\delta(1-\delta) + (1-\delta)^2}{\delta^2 + 3\delta(1-\delta) + 3(1-\delta)^2} = \frac{1}{\delta^2 - 3\delta + 3} \tag{3}$$

Elementary analysis shows that P_δ is an increasing function of δ in the range $(0,1)$, with $\lim_{\delta\to 0} P_\delta = 1/3$, $\lim_{\delta\to 1} P_\delta = 1$, and $P_\delta = 1/2$ when $(\delta-1)^2 = \delta$, which happens for $\delta^\star = \frac{3-\sqrt{5}}{2} \approx 0.382$.

From here we are almost done. Whp $|\alpha_0| \geq (1/2-\epsilon)n$, and for any assignment α_t, the probability of a good move depends only on $|\alpha_t|$. Thus the probability $\texttt{RWalkSAT}$ succeeds in T steps is at most the probability of a random walk on the line, starting at distance $> (1/2-\epsilon)n$, reaches 0 in T steps, where the probability of moving in the direction of 0 when standing at position i is $P_{i/n}$. Using (3), for $i \leq n/10$ we get $P_{i/n} < 1/2 - 1/10$. Lemma 4.5 completes our proof. □

4.2 Generalization to $\mathbb{P}_\Delta^n$

In this section we complete the proof of theorem 4.2. As previously mentioned, the lower bound follows from the observation that $\mathcal{C}$ is obtained by taking a "small piece" of $\mathcal{F}_n$. Thus, we expect the behavior of $\texttt{RWalkSAT}$ on $\mathcal{C}$ to be similar to its behavior on $\mathcal{F}_n$. There are two points we need to be careful about. The first is that $\mathcal{C}$ may have other satisfying assignments. We argue in lemma 4.6 that although this is true, for large Δ these assignments are all found within small Hamming distance of $\vec{0}$. The second problem is that there might be small weight assignments for which the probability of making a good step is large. We show in lemma 4.7 that for large Δ, whp this does not happen. Thus $\texttt{RWalkSAT}$ operates on $\mathbb{P}_\Delta^n$ almost as poorly as it operates on $\mathcal{F}_n$. The proofs of lemmas 4.6,4.7 are deferred to appendix E.

Proof [Theorem 4.2]: Let A_δ be the event that $\mathcal{C}$ has a satisfying assignment of Hamming distance greater than δn from β (where β is the planted assignment).

Lemma 4.6 *For any $\delta > 0$ there exists a constant Δ', such that for all $\Delta \geq \Delta'$, the probability that $\mathcal{C} \sim \mathbb{P}_\Delta^n$ has property A_δ is exponentially small.*

Define a *good move* of RWalkSAT as one that decreases the weight of the current assignment. Call a clause *good* iff it has at least two negative literals. A good clause, if selected, will make a good move with probability at least $2/3$, whereas a *bad clause*, having only one negative literal, will make a good move only with probability $1/3$. We claim that the numbers of bad and good clauses are tightly concentrated around their expected values, and thus the probability of a good move in $\mathcal{C}$ is roughly that of a good move on $\mathcal{F}_n$.

Fix $\alpha \in \{0,1\}^n$, $|\alpha| = \delta n$, and recall the definition of $m_{\delta,i}$ from equations (1). For $i = 1 \ldots 3$ let $X_{\delta,i}$ be the number of clauses in $\mathcal{C}$ that have i negative literals, and are not satisfied by α. Since α is fixed, whereas $\mathcal{C}$ is random, $X_{\delta,i}$ is a random variable depending only on δ and independent of the particular assignment α. $X_{\delta,i}$ is a sum of $m_{\delta,i}$ independent coin tosses, each with probability p_n^{Δ}. Moreover, all $X_{\delta,i}$'s are completely independent of each other, because the $\mathcal{F}_n^i$'s are disjoint. Let $B[n,p]$ denote the binomial distribution over n coin tosses, each with success probability p. According to our definitions, $X_{\delta,i} \sim B[m_{\delta,i}, p_n^{\Delta}]$, and $\mu_{\delta,i} \stackrel{\text{def}}{=} E[X_{\delta,i}] = p_n^{\Delta} \cdot m_{\delta,i}$. Plugging in the values of $m_{\delta,i}$ we get

$$\mu_{\delta,1} = \frac{3\delta(1-\delta)^2 \Delta n}{7} \tag{4}$$

$$\mu_{\delta,2} = \frac{3\delta^2(1-\delta)\Delta n}{7} \tag{5}$$

$$\mu_{\delta,3} = \frac{\delta^3 \Delta n}{7} \tag{6}$$

Let $B_\delta^1(\epsilon)$ be the event that the number of bad clauses not satisfied by α, $X_{\delta,1}$, is smaller than $(1-\epsilon)\mu_{\delta,1}$. Similarly, let $B_\delta^2(c)$ be the probability that the number of good clauses, $X_{\delta,2} + X_{\delta,3}$ is greater than $(1+c) \cdot (\mu_{\delta,2} + \mu_{\delta,3})$. Notice that by definition, $\epsilon \leq 1$, whereas c can be much larger. Let $Bad(\delta, c, \epsilon)$ be the event that there exists an assignment α, $|\alpha| = \delta' n$, $\delta/2 \leq \delta' \leq \delta$, such that $B_{\delta'}^1(\epsilon)$ or $B_{\delta'}^2(c)$ occurs, and hence the numbers of good or bad clauses are far from the expected. The next lemma shows that for large enough Δ, this is not likely to happen.

Lemma 4.7 *For any $\epsilon > 0, c > 2e^2$ and $0 < \delta \leq 1/2$ there exists a constant $\Delta'' > 0$ such that for all $\Delta \geq \Delta''$, the probability that $\mathcal{C} \sim \mathbb{P}_\Delta^n$ has property $Bad(\delta, c, \epsilon)$ is exponentially small.*

We do not attempt to optimize constants. Set $\epsilon = 1/2, c = 20$, and $\delta = c^{-3}$. By lemmas 4.6 and 4.7, there exists some Δ for which both $\mathbf{P}[A_{\delta/2}]$ and $\mathbf{P}[Bad(\delta, c, \epsilon)]$ are exponentially small. Fix this Δ. Whp for $\mathcal{C} \sim \mathbb{P}_\Delta^n$, all satisfying assignments have weight $< \delta n/2$. Notice that for $\delta < 1/2$, $\mu_{\delta,i}$ increases with δ. For any α with weight $\delta n/2 \leq |\alpha| \leq \delta n$, by lemma 4.7 the number of bad clauses unsatisfied by α is at least

$$1/2 \cdot \mu_{\delta/2,1} > \frac{3\delta\Delta n}{28} > \frac{c^{-3}\Delta n}{10}$$

Similarly, assuming $\mu_{\delta,2} > \mu_{\delta,3}$, which is true for $\delta < 1/2$, the number of good clauses unsatisfied by α is at most

$$21(\mu_{\delta,2} + \mu_{\delta,3}) \leq 42 \cdot \frac{3}{7}\delta^2(1-\delta)\Delta n < 20\delta^2 \Delta n = c^{-5}\Delta n$$

Thus,

$$\mathbf{P}[\text{Good move}] \leq \frac{\left(\frac{1}{3} \cdot \frac{c^{-3}}{10} + c^{-5}\right)\Delta n}{\frac{c^{-3}}{10}\Delta n} = \frac{1}{3} + \frac{1}{40} << 1/2$$

Whp the initial assignment of RWalkSAT will have weight greater than δn. Any satisfying assignment must have weight less than $\delta n/2$. For any assignment α having weight $\delta n/2 \leq |\alpha| \leq \delta n$, the probability of RWalkSAT making a good move is very close to $1/3$. Lemma 4.5 completes the proof of the theorem. $\square$

5 Open Problems

Deferred to appendix F

6 Acknowledgments

We thank Madhu Sudan for many useful discussions. The second author thanks Rocco Servedio and Salil Vadhan for many helpful discussions. We thank Bart Selman and Andrew Parkes for valuable information on the empirical results regarding `RWalkSAT`.

References

[1] O. Chinese Proverb. In *Fortune Cookie*, Chang Sho Restaurant, Cambridge, MA.

[2] D. Achlioptas, P. Beame, M. Molloy. A Sharp Threshold in Proof Complexity. In *Proc. STOC 2001*, pp 337-346.

[3] D. Achlioptas and G. B. Sorkin. Optimal myopic algorithms for random 3-SAT In *IEEE Symposium on Foundations of Computer Science* pp 590-600 (2000).

[4] N. Alon, M. Krivelevich, B. Sudakov. Finding a large hidden clique in a random graph, In *Random Structures and Algorithms* 13 (1998), 457-466.

[5] E. Ben-Sasson. Size Space Tradeoffs for Resolution. In *STOC 2002*.

[6] E. Ben-Sasson, Y. Bilu, D. Gutfreund. Finding a Randomly Planted Assignment in a Random 3-CNF. In preparation (2002).

[7] E. Ben-Sasson, A. Wigderson. Short proofs are narrow - resolution made simple. In *Proceedings of the* 31*st ACM STOC*, pages 517–526, 1999.

[8] A. Broder, A. Frieze, E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 322-330.

[9] B. Cha, K. Iwama. Performance Test of Local Search Algorithms Using New Types of Random CNF Formulas In *Proc. IJCAI-95 14th International Joint Conference on Artificial Intelligence*,Vol. 1, pp 304-310, Montreal, August, 1995.

[10] V. Chvátal, E. Szemerédi. Many Hard Examples for Resolution. *In Journal of the Association for Computing Machinery*. vol. 35 (1988), pp. 759-768.

[11] S.A. Cook. The Complexity of Theorem-Proving Procedures. In *Proc.* 3*rd IEEE FOCS* pp. 151-158, 1971.

[12] S.A. Cook, D. Mitchell. Finding Hard Instances of the Satisfiability Problem: A Survey. In *DIMACS Series in Discrete Math. and Theoretical Computer Science*, vol. 35, 1997, pp 1-17.

[13] Crawford, J. M., and Auton, L. D. Experimental Results on the Crossover Point in Random 3-SAT. In *Artificial Intelligence*, 81:31-57.

[14] U. Feige. Relations Between Average Case Complexity and Approximation Complexity. In *Proc. of STOC 2002*, Montreal.

[15] U. Feige, R. Krauthgamer Finding and certifying a large hidden clique in a semi-random graph In *Random Structures and Algorithms* 16(2): 195-208, 2000.

[16] E. Friedgut. Sharp Thresholds of Graph Properties, and the k-sat Problem. In *J. Amer. Math. Soc. 12 (1999)*, no. 4, 1017–1054.

[17] S. Janson, Y.C. Stamatiou, M. Vamvakari. Bounding the unsatisfiability threshold for 3-SAT. *Random Structure and Algorithms* (17) 2 pp.118-116 (2000).

[18] M. Jerrum, G. B. Sorkin. Simulated annealing for graph bisection. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, 94–103, November 1993.

[19] Kirpatrick S., C.D. Gelatt, M.P. Vecchi Optimization by simulated annealing. In *Science* 220, 671-680 (1983).

[20] Koutsoupias E. and C. H. Papadimitriou. On the greedy algorithm for satisfiability. In *Information Processing Letters*, 43(1):53–55, August 1992.

[21] Luby M. , M. Mitzenmacher, and A. Shokrollahi. Analysis of Random Processes via And-Or Tree Evaluation. In *Proceeding of ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[22] Motwani R., P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[23] Papadimitriou, C. H. On selecting a satisfying truth assignment. In *Proceedings of the 32nd Annual IEEE FOCS'91*, pages 163-169,

[24] Papadimitriou, C. H. Computational Complexity. *Addison Wesley*, 1994.

[25] Parkes, A. J. Easy Predictions for the Easy-Hard-Easy Transition. In *Proc. AAAI02*, 2002.

[26] Parkes, A. J. Personal Communication.

[27] Paturi R., Pavel Pudlk, Michael E. Saks, Francis Zane An Improved Exponential-time Algorithm for k-SAT In *IEEE Symposium on Foundations of Computer Science* 1998.

[28] Schoning, U. A probabilistic algorithm for k-SAT and constraint satisfaction problems, In *Proc. FOCS'99*, pp.410–414, 1999.

[29] Selman, B. Personal Communication.

[30] Selman, B., H. Kautz Local Search Strategies for Satisfiability Testing In *DIMACS Series in Discrete Mathematics.* (to appear)

[31] B. Selman, H. Kautz An Empirical Study of Greedy Local Search for Satisfiability Testing. In *Proc. AAAI-93.*

[32] Selman B., Kautz H., Cohen B. Local Search Strategies for Satisfiability Testing. In *Dimacs Series in Discr. Math. and Theoretical Computer Science*, Vol. 26, 1996, 521–532.

[33] Selman B., Levesque H., Mitchell D. A New Method For Solving Hard Satisfiability Problems. In *Proc. of the Tenth Natl. Conference on Artificial Intelligence (AAAI-92)*, San Jose, CA, 1992, 440-446.

A Definitions

For x a Boolean variable, a *literal* over x is either x or $\bar{x}$ (the negation of x), where x is called a *positive literal* and $\bar{x}$ is a *negative* one. A *clause* is a disjunction of literals, and a CNF formula is a set of clauses. Throughout this paper we reserve calligraphic notation for CNF formulas. For $\mathcal{C}$ a CNF, let $Vars(\mathcal{C})$ denote the set of variable appearing in $\mathcal{C}$. An *assignment* to $\mathcal{C}$ is some Boolean vector $\alpha \in \{0,1\}^{Vars(\mathcal{C})}$. The *weight* of α, denoted $|\alpha|$, is the number of ones in it.

A *coin toss* X is an independent $\{0,1\}$-valued random variable, and its *success probability* is $\mathbf{P}[X=1]$. For P a distribution over some universe Ω, we use the notation $X \sim P$ to denote that X is a random element from Ω, picked according to the distribution P. We use the following Chernoff bounds (see e.g. [22], Theorems 4.1, 4.2).

Theorem A.1 *For $X \sim B[n,p]$, with $0 < p < 1$, $\mu = E[X] = np$, and for all $c > 0$:*

$$\mathbf{P}[X > (1+c)\mu] < \left(\frac{e^c}{(1+c)^{1+c}}\right)^{\mu} \tag{7}$$

For $0 < \epsilon \leq 1$:

$$\mathbf{P}[X < (1-\epsilon)\mu] < e^{-\epsilon^2\mu/2} \tag{8}$$

Additionally, we make often use of the standard bound

$$\binom{n}{\lambda n} \leq e^{H_e(\lambda)\cdot n} \tag{9}$$

where $H_e(q) = q \ln \frac{1}{q} + (1-q) \ln \frac{1}{1-q}$ is the entropy measure.

B RWalkSAT - Pseudocode

```
RWalkSAT(𝒞,T)
Select α ∈ {0,1}^n at random
Initialize i = 0
While t < T do {
      If 𝒞(α) = 1 return (''Input is satisfied by '' α)
      Else { Select C ∈ UNSAT(𝒞,α) at random
             Select literal ℓ ∈ C at random
             Flip assignment of α at ℓ.
             t++ } }
Return ''Failed to find satisfying assignment in T steps.''
```

C Proof of Theorem 3.2

By induction on r. For $r = 1$, notice that all clauses in $\mathcal{C}$ are pure, and thus, if $C \in \mathcal{C}$ is the clause unsatisfied by α_t, and selected by `RWalkSAT` at time t to be satisfied, with probability $\geq 1/3$ we will fix a pure literal to its correct value, and thus the clause C will never be unsatisfied again during the execution of the algorithm.

For $t \leq 6m$, let X_t be the indicator random variable that is 1 if the clause considered at time t was satisfied by a pure literal, and 0 if this is not the case. If `RWalkSAT` terminates at time $T < 6m$, let $X_t,\ \ t > T$ be an independent coin toss with success probability $1/3$. Let $X = \sum_{t=1}^{6m} X_t$. Notice that by

definition of the algorithm, and the fact that all clauses are pure, each X_t is an independent coin toss with success probability $\geq 1/3$, so $\mu = \mathbf{E}(X) \geq 2m$. By the Chernoff bound (7) we get:

$$\Pr[\texttt{RWalkSAT}(\mathcal{C}, 6m) \text{ fails}] \leq \Pr[X < 1/2\mu] < e^{-\mu/8} = e^{-m/4} \tag{10}$$

For the induction step, notice that the CNF P_1 is 1-pure, and $\mathcal{C}_1$ is an $(r-1)$-pure CNF over the variable set $Vars(\mathcal{C}) \setminus Vars(L_1)$. We can view RWalkSAT as operating on $\mathcal{C}_1$ with occasional "noise". Each time a clause $C \in P_1$ is selected one of two things may happen. Either C is satisfied by fixing some pure literal to its satisfying assignment. In this case C is permanently satisfied and removed from $\mathcal{C}$ and this removal has no effect on $\mathcal{C}_1$, since $Vars(\mathcal{C}_1) \cap Vars(L_1) = \emptyset$. Otherwise, C is satisfied by some literal that is not in L_1, and we view this as a perturbation to the current assignment α_t. The main point is that with high probability, the number of such perturbations in $O(m)$. Now we apply induction to claim that after each such perturbation, RWalkSAT will take at most $(6m)^{r-1}$ time to find a satisfying assignment to $\mathcal{C}_1$.

Formally, divide the execution of $\texttt{RWalkSAT}(\mathcal{C}, (6m)^r)$ into *epochs*. An epoch starts after a clause $C \in \mathcal{C}_1$ is picked and satisfied by a literal *not* in L_1. Let s be the number of epochs, and p be the maximal time of an epoch.

Claim C.1 $\Pr[s > 6m] \leq e^{-m/4}$.

Proof: Similar to the case of $r = 1$. Let T be the number of clauses from P_1 considered during the execution of RWalkSAT, and let $C_1, \ldots, C_T$ be the set of these clauses, ordered in the order which they were picked by the algorithm, with repetitions. For $t \leq 6m$, let X_t be the indicator random variable that is 1 if the C_t was satisfied by a pure literal, and 0 otherwise. If $T < 6m$, let $X_t, \ \ t > T$ be an independent coin toss with success probability $1/3$. Let $X = \sum_{t=1}^{6m} X_t$. Apply the Chernoff bound as in (10). □

$\mathcal{C}_1$ is $r-1$ pure, so by induction we know that for any assignment α

$$\mathbf{P}[\texttt{RWalkSAT}(\mathcal{C}_1, (6 \cdot m)^{r-1}) \text{ fails} \mid \alpha_0 = \alpha] < (7m)^{r-1} \cdot e^{-m/4}$$

Using claim C.1 we get that for any α,

$$\begin{aligned} \mathbf{P}[\texttt{RWalkSAT}(\mathcal{C}, (6 \cdot m)^r) \text{ fails} \mid \alpha_0 = \alpha] &\leq \mathbf{P}[s > 6m] + \mathbf{P}[p > (6m)^{r-1} \mid s \leq 6m] && (11) \\ &< e^{-m/4} + 6m \cdot (7m)^{r-1} \cdot e^{-m/4} && (12) \\ &\leq (7m)^r \cdot e^{-m/4} && (13) \end{aligned}$$

Theorem 3.2 follows. □

D Tightness of Bounds

One may hope that in the case of a formula that is satisfiable by the simple pure literal heuristic, there might be a better upper bound for RWalkSAT. In this subsection we show that our method encapsulated in theorem 3.2 is essentially tight. The following theorem also shows a separation RWalkSAT from the pure literal heuristic.

Theorem D.1 *For arbitrarily large n, there exist formulas of size n, that are satisfiable (in polynomial time) by the pure literal heuristic, whereas* $\mathbf{P}[\texttt{RWalkSAT}\ (\mathcal{C}_n, 2^{\epsilon n}) \text{ succeeds}] = o(1)$.

Proof: Use the following formula, which is a slight variation of the X-DAG contradiction used in [5].

Definition D.2 *Let $\mathcal{G}_n$ be the following CNF over variables $x_1, \ldots, x_n, y_1, \ldots, y_n, z$:*

$$\{\bar{x}_1\} \wedge \{\bar{y}_1\} \wedge \bigwedge_{i=1}^{n-1} \{x_i \vee y_i \vee \bar{x}_{i+1}\} \wedge \bigwedge_{i=1}^{n-1} \{x_i \vee y_i \vee \bar{y}_{i+1}\} \wedge \{x_n \vee y_n \vee \bar{z}\}$$

$\mathcal{G}_n$ has a unique satisfying assignment, $\vec{0}$. Moreover, $\mathcal{G}_n$ is n-pure, because $\bar{z}$ appears only in one clause, and once z is satisfied and removed, $\bar{y}_n, \bar{x}_n$ each appear in one clause in the remaining formula. Thus, one can repeatedly remove x_{i-1}, y_{i-1} until all the formula is satisfied. This immediately implies that the pure literal heuristic succeeds on $\mathcal{G}_n$ (in polynomial time). We claim that `RWalkSAT` requires exponential time to succeed on $\mathcal{G}_n$.

Let X_t be the number of ones assigned by α_t to the variables $x_2, \ldots x_n, y_2, \ldots y_n$. Whp $X_0 > (1-\epsilon)n$, and if `RWalkSAT`$(\mathcal{G}_n, T)$ succeeds, we know $X_T = 0$. But for every step t, the probability of X_t decreasing is at most $1/3$. The theorem follows from standard arguments from the theory of random walks. □

E Proofs

Proof [Lemma 3.5]: First we note that a very good expander is also a decent boundary expander:

Lemma E.1 *If $\mathcal{C}$ is 3-CNF (r, c)-expander, then it is an $(r, 2 \cdot c - 3)$-boundary expander.*

Our proof follows immediately from lemma E.1 and the following slightly stronger version[3] of lemma 4.4 of [8].

Lemma E.2 *[8] Let $\Delta_0 = 1.63$. For any constant $\Delta \leq \Delta_0$ whp $\mathcal{C} \sim \mathbb{F}_\Delta^n$ is a $(\frac{n}{600\Delta_0^2}, 3/2 + 10^{-3})$-expander.*

Proof: Set $\epsilon = 10^{-3}$. Let A_k be the event that there exists a set of k clauses having less than $3/2 + \epsilon$ variables. Let us bound the probability of these bad events, using a union bound. Let $r = \frac{n}{600\Delta_0^2}$, and $c = 3/2 + \epsilon$. We make use of the following well-known inequality $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$.

$$\begin{aligned}
\mathbf{P}[Bad] &\leq \sum_{k=1}^{r} \mathbf{P}[A_k] \leq \sum_{k=1}^{r} \binom{\Delta n}{k} \cdot \binom{n}{ck} \cdot \left(\frac{ck}{n}\right)^{3k} && (14)\\
&\leq \sum_{k=1}^{r} \left(\frac{e\Delta_0 n}{k}\right)^k \cdot \left(\frac{en}{ck}\right)^{ck} \cdot \left(\frac{ck}{n}\right)^{3k} && (15)\\
&\leq \sum_{k=1}^{r} \left[(e^{1+c} c^{3-c} \Delta_0) \cdot \left(\frac{k}{n}\right)^{2-c}\right]^k && (16)\\
&\leq \sum_{k=1}^{r} \left[37 \cdot \left(\frac{k}{n}\right)^{\frac{1}{2}-\epsilon}\right]^k = o(1) && (17)
\end{aligned}$$

Where the last inequality holds for $r \leq \frac{n}{600\Delta_0^2}$. □

Proof [Lemma 3.7]: We need the following lemma of [10].

[3] The original lemma 4.4 of [8] only needed expansion factor of $3/2$, whereas we need a constant fraction more than $3/2$. The proof is essentially the same.

Lemma E.3 *[10] For all constants $\Delta > 0, c < 2$ there exists some constant $\delta > 0$ such that whp $\mathcal{C} \sim \mathbb{F}_\Delta^n$ is a $(\delta\Delta n, c)$-expander.*

Let δ be the constant promised by lemma E.3, for $\Delta = 1.63$ and $c = 7/4$. By theorem 3.1, $|\mathcal{C}_{d_1}| \leq n/(600\Delta_0^2)$, for some constant d_1. By lemma E.2, $\mathcal{C}_{d_1}$ is a $(|\mathcal{C}_{d_1}|, \epsilon)$-boundary expander, for some $\epsilon > 0$. As observed in the proof of theorem 3.3, $|P_i| \geq \epsilon' n$ for all $i \geq d_1$, and hence $|\mathcal{C}_{d_2}| \leq \delta n$. The lemma is proved. □

Proof [Lemma 4.5]: Let $T = 2^{\epsilon n}$. for $t \leq T$, let r_t be the position of R at time t, and let X_t be the random variable that is 1 if R does a good move (towards 0) at time t, and -1 otherwise, Formally, $X_t = r_t - r_{t+1}$. Assume R reaches 0 at time t_2. Since $r_1 \geq b$, there must be a time interval $[t_1, t_2]$ in which $0 \leq r_t \leq b \;\; t \in [t_1, t_2]$. This means $\sum_{t=t_1}^{t_2} X_t = c$, where $c = b - a$. Let $t_2 - t_1 = 2s + c$. Recalling that for all $t \in [t_1, t_2]$, $\mathbf{P}[X_t = -1] \leq 1/2 - \gamma$, we get

$$\begin{aligned}
\mathbf{P}[\sum_{t=t_1}^{t_2} X_t = c] &\leq \binom{2s+c}{s} \cdot (1/2 - e)^{s+c} \cdot (1/2 + \gamma)^s && (18)\\
&\leq \binom{2s+c}{s+c/2} \cdot (1/4 - \gamma^2)^s \cdot (1/2 - \gamma)^c && (19)\\
&\leq \frac{2^{2s+c}}{\sqrt{2s+c}} \cdot (1/4 - \gamma^2)^s \cdot (1/2 - \gamma)^c && (20)\\
&\leq \frac{1}{\sqrt{2s+c}} \cdot (1 - 4\gamma^2)^s \cdot (1 - 2\gamma)^c && (21)\\
&\leq \frac{1}{\sqrt{2s+c}} \cdot \exp(-(4\gamma^2 s + 2\gamma c)) && (22)\\
&\leq \frac{\exp(-(2\gamma c))}{\sqrt{c}} && (23)
\end{aligned}$$

Since there are at most T^2 possible selections of t_1, t_2, setting $T = e^{\gamma c/2}$, and taking a union bound, completes the proof. □

Proof [Lemma 4.6]: Assuming wlog $\beta = \vec{0}$, let us bound the probability of the event A_δ. For each fixed $\alpha \in \{0,1\}^n$ having weight δn, there are $\binom{n}{3} - \binom{(1-\delta)n}{3} \approx (1 - (1-\delta)^3)n^3/6$ clauses that are satisfied by $\vec{0}$ and not by α.

$$\begin{aligned}
\mathbf{P}[\mathcal{C}(\alpha) = 1] &\approx (1 - p_n^\Delta)^{(1-(1-\delta)^3)n^3/6} && (24)\\
&\leq \exp(-(1 - (1-\delta)^3) \cdot \Delta/7) && (25)
\end{aligned}$$

Using a union bound and the standard inequality (9), we get

$$\begin{aligned}
\mathbf{P}[A_\delta] &\leq n \cdot \binom{n}{\delta n} \cdot \exp(-(1 - (1-\delta)^3) \cdot \Delta/7) && (26)\\
&\leq \exp((H_e(\delta) - \frac{\Delta(1 - (1-\delta)^3)}{7}) \cdot n + \ln n) && (27)
\end{aligned}$$

Thus, when

$$\Delta > \frac{7H_e(\delta)}{1 - (1-\delta)^3} \qquad (28)$$

we get that $\mathbf{P}[A_\delta] \leq \exp(-\epsilon n)$, for some constant $\epsilon > 0$. $\square$

Proof [Lemma 4.7]: We bound the probabilities of $B^1_{\delta'}(\epsilon)$ and $B^2_{\delta'}(c)$ individually, for any $\delta/2 \leq \delta' \leq \delta$. We start with $B^1_{\delta'}(\epsilon)$. Plugging the value of $\mu_{\delta,1}$, given in (4), into the Chernoff bound (8) we get

$$\mathbf{P}[B^1_{\delta'}(\epsilon)] < \exp(-\epsilon^2 \mu_{\delta',1}/2) < \exp(\frac{-3\epsilon^2 \delta'(1-\delta')^2 \Delta n}{14}) \tag{29}$$

Now we deal with $B^2_{\delta'}(c)$. First notice that for any $\delta < 1/2$, $m_{\delta,3} < m_{\delta,2}$, so

$$\mathbf{P}[B^2_{\delta'}(c)] = \mathbf{P}[X_{\delta',2} + X_{\delta',3} > (1+c)\cdot(\mu_{\delta',2} + \mu_{\delta',3})] \leq 2\mathbf{P}[X_{\delta',2} > (1+c)\mu_{\delta',2}]$$

Assuming $c > 2e^2$, and plugging the value of $\mu_{\delta,2}$ given in (5) into the Chernoff bound (7) we get

$$\begin{aligned}
\mathbf{P}[B^2_{\delta'}(c)] &\leq 2\mathbf{P}[X_{\delta',2} > (1+c)\mu_{\delta',2}] && (30)\\
&< 2(\frac{e^c}{(1+c)^{1+c}})^{\mu_{\delta',2}} < (\frac{e}{c})^{c\mu_{\delta',2}} < \exp(-c\mu_{\delta',2}) && (31)\\
&= \exp(\frac{-3c\delta'^2(1-\delta')\Delta n}{7}) && (32)
\end{aligned}$$

Notice that for $\delta < 1/2$, the value of (29) and (32) is maximal when δ' is minimal (i.e $\delta' = \delta/2$). Using a union bound over all assignments of weight between $\delta n/2$ and δn, we get

$$\begin{aligned}
\mathbf{P}[Bad(\delta,c,\epsilon)] &\leq n \cdot \binom{n}{\delta n} \cdot \Big(\mathbf{P}[B^1_{\delta/2}(\epsilon)] + \mathbf{P}[B^2_{\delta/2}(c)]\Big)\\
&\leq 2n \cdot \exp\left(n \cdot \left(H_e(\delta) - \Delta \cdot \min\{\frac{-3\epsilon^2(\delta/2)(1-\delta/2)^2}{14}, \frac{-3c(\delta/2)^2(1-\delta/2)}{7}\}\right)\right)
\end{aligned}$$

Notice that once δ, ϵ, c are fixed, we can select Δ such that

$$H_e(\delta) - \Delta \cdot \min\{\frac{-3\epsilon^2(\delta/2)(1-\delta/2)^2}{14}, \frac{-3c(\delta/2)^2(1-\delta/2)}{7}\} < 0$$

Lemma 4.7 follows. $\square$

F Open Problems

1. What is the largest Δ for which one can prove `RWalkSAT` to have polynomial running time on $\mathcal{C} \sim \mathbb{F}^n_\Delta$? As a first step, can one prove polynomial upper bounds for $\Delta \leq 3.22$, for which similar bounds are known for myopic algorithms [3] ?

2. What is the largest Δ for which one can prove `RWalkSAT` to have polynomial running time on $\mathcal{C} \sim \mathbb{P}^n_\Delta$?

3. Is there an efficient algorithm that finds whp a satisfying assignment for $\mathcal{C} \sim \mathbb{P}^n_\Delta$, and large constant Δ ?

4. What is the connection of $\mathbb{P}^n_\Delta$ to $\mathbf{SAT}^\Delta_n$ for constant Δ ? Do bounds in one distribution imply bounds in the other ?

5. Can one further characterize instances and distributions on which `RWalkSAT` performs well ? How about instances that correspond to CNFs encountered in practice ?