Linear Upper Bounds for Random Walk on Small Density Random 3-CNFs

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

Eli Ben-Sasson†
Division of Engineering and Applied Sciences
Harvard University and
Laboratory for Computer Science
MIT
Cambridge, MA.
eli@eecs.harvard.edu.

Abstract

We analyze the efficiency of the random walk algorithm on random 3*-CNF instances, and prove* linear *upper bounds on the running time of this algorithm for small clause density, less than* 1.63*. Our upper bound matches the observed running time to within a multiplicative factor. This is the first sub-exponential upper bound on the running time of a* local improvement *algorithm on random instances.*

Our proof introduces a simple, yet powerful tool for analyzing such algorithms, which may be of further use. This object, called a terminator, *is a weighted satisfying assignment. We show that any CNF having a good (small weight) terminator, is assured to be solved quickly by the random walk algorithm. This raises the natural question of the* terminator threshold *which is the maximal clause density for which such assignments exist (with high probability).*

We use the analysis of the pure literal heuristic presented by Broder, Frieze and Upfal [12, 22] and show that for small clause densities good terminators exist. Thus we show that the Pure Literal threshold (≈ 1.63*) is a lower bound on the terminator threshold. (We conjecture the terminator threshold to be in fact higher).*

One nice property of terminators is that they can be found efficiently, via linear programming. This makes tractable the future investigation of the terminator threshold, and also provides an efficiently computable certificate for short running time of the simple random-walk heuristic

*Supported in part by NSF Awards CCR 0205390 and MIT NTT Award 2001-04.

†Supported by NSF grants CCR-0133096, CCR-9877049, CCR 0205390, and NTT Award MIT 2001-04.

1. Introduction

The phenomena we explain in this paper is best described by the following figure 1.

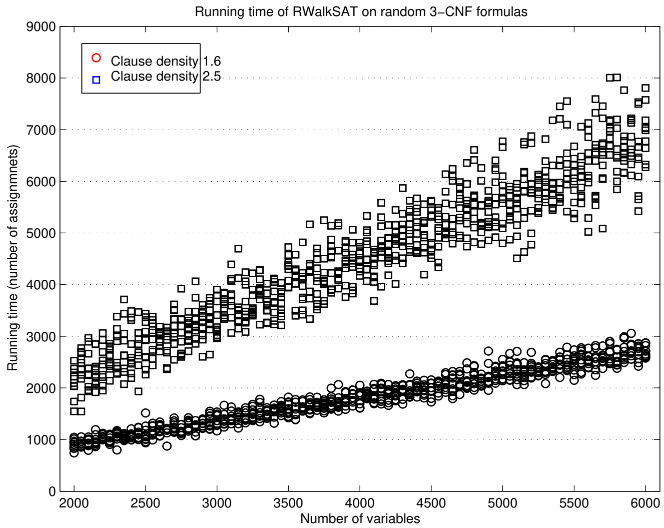

Figure 1. Running time of `RWalkSAT` **on random 3-CNF formulas. Circles represent instances with clause density 1.6 and squares represent clause density 2.5.**

`RWalkSAT` , originally introduced by Papadimitriou [24], tries to find a satisfying assignment for a CNF $\mathcal{C}$ by the following method. We start with a random assignment and as long as the assignment at hand does not satisfy the CNF, an unsatisfied clause $C \in \mathcal{C}$ is picked and the assignment to a random literal in this clause is flipped. The new assignment satisfies C, but may "ruin" the satisfiability of other clauses. We repeat this process (of flipping a bit in the current assignment according to some unsatisfied clause) until either a satisfying assignment is found (success), or we get tired and give up (failure).

In figure 1 we selected 1,600 random 3-CNF formulas and gave them as inputs to `RWalkSAT` . Half of the instances (circles) have clause density 1.6 (i.e. the clause/variable ratio is 1.6) and the other half have higher clause density 2.5. The number of variables ranges between $2,000$ and $6,000$. For each formula, we display the number of assignments checked until a satisfying assignment was found (the algorithm succeeded on all inputs).

Looking at this figure raises the conjecture that in both cases (clause densities) the running time is linear. Actually, the running time is even less than the number of variables (and clauses). It seems like the slope of the 1.6 density batch is $\approx 1/2$ whereas that of the 2.5 density is ≈ 1.

In this paper we offer an explanation of the lower batch (circles). We prove that random 3-CNFs with clause density less than 1.63 require (with high probability) a *linear* number of `RWalkSAT` steps. We leave the explanation of the upper cloud as an intriguing open problem, showing our results are merely the first step in understanding this phenomena.

1.1. Techniques - Terminators

Our technique can be viewed as a generalization of the analysis of `RWalkSAT` on satisfiable 2-CNF formulas [24], so we briefly review this result. Papadimitriou showed that the Hamming distance of the assignment at time t from some fixed satisfying assignment α, is a random variable that decreases at each step with probability at least $1/2$. Thus, in at most $O(n^2)$ this random variable will reach 0, implying we have found α. (Of-course, the algorithm may succeed even earlier, by finding some other satisfying assignment).

We look at *weighted satisfying assignments*, i.e. we give non-negative weights to the bits of α. Instead of Hamming distance, we measure the *weighted distance* between α and the current assignment α^t. We show that in some cases, one can find a satisfying assignment α and a set of weights w such that for *any* unsatisfied clause at time t, the expectation of the weighted distance (between α and α^t) decreases by at least 1 (!). Moreover, the maximal weight given to any variable is constant. In this case the running time of `RWalkSAT` will be linear with high probability (even better than the quadratic upper bound of [24] for 2-CNFs). We call such weighted assignments *terminators*, as their existence assures us that `RWalkSAT` will terminate successfully in linear time.

Two parameters of a terminator bound the running time of `RWalkSAT`. The *total weight* (sum of weights of all variables) bounds the distance needed to be traversed by the random walk, because the weighted distance of α^0 from α can be as large as this sum. The second parameter is the maximal weight of a variable, that bounds the variance of our random walk. Thus we define the termination weight of $\mathcal{C}$ (denoted $\mathrm{Term}(\mathcal{C})$) to be the minimal product of these two parameters, taken over all terminators for $\mathcal{C}$. As stated above, the running time of `RWalkSAT` is linear (at most) in the termination weight of $\mathcal{C}$. Not all satisfiable CNFs have these magical terminators, and if $\mathcal{C}$ has no terminator we define its termination weight to be ∞.

1.2. Results

With the terminator concept in hand, we examine the running time of `RWalkSAT` on random 3-CNF formulas. If $\mathcal{C}$ is a random 3-CNF then $\mathrm{Term}(\mathcal{C})$ is a random variable. Understanding this variable and bounding it from above bounds the running time of `RWalkSAT` . Our main result (Theorem 4.1) is that for clause density ≤ 1.63, a random 3-CNF has *linear* termination weight (hence `RWalkSAT` succeeds in linear time). This matches the behavior of the lower batch in figure 1 up to a multiplicative constant.

Our result relies on previous analysis done for bounding a different SAT heuristic, called the *pure literal heuristic* [12] (see also [22] for a different and shorter analysis). This heuristic is known to succeed up to a clause density threshold of 1.63 and fail above this density. We believe terminators should exist even beyond the pure literal threshold, as the experimental data seems to indicate (see Appendix B). However, at clause density ≥ 2 only a negligible fraction of random CNFs have terminators, meaning we need to develop new techniques for explaining the observed linear running time at (say) density 2.5 depicted in figure 1 (see Section 5).

One nice property of terminators is that they can be found efficiently. A terminator is a solution to a linear system of inequalities, and thus linear programming can be used to find it. Thus, the behavior of the random variable $\mathrm{Term}(\mathcal{C})$ for random $\mathcal{C}$ can be estimated and analyzed efficiently (see Appendix B). These tests will allow us to better understand the behavior of `RWalkSAT` and its connection to the termination weight parameter.

The success of the pure literal heuristic does not necessarily imply polynomial running time for `RWalkSAT` . Indeed, in section 6 we provide a counter example that requires exponential time from `RWalkSAT` , although a solution can be found using the pure literal heuristic in linear time.

1.3. History and Related Results

Local Improvement Algorithms: `RWalkSAT` was introduced by Papadimitriou, who showed it has quadratic running time on satisfiable 2-CNFs [24]. An elegant upper bound was given by Schoning, who showed that the expected running time of `RWalkSAT` on any k-CNF is at most $(1 + 1/k)^n$ (compared with the exhaustive search upper bound of 2^n) [27]. The (worst case) upper bound of [27] was improved in a sequence of results, [10, 14, 5, 15, 26], and the best upper bound for 3-SAT is $(1.324)^n$ given by the recent paper [15].

`RWalkSAT` is one of a broad family of *local improvement* algorithms, (re)introduced in the 1990's with the work of [30]. These algorithms (the most famous of which is WalkSAT) are close relatives of the Simulated Annealing method, and were found to compete with DLL-type algorithms (also known as Davis-Putnam algorithms). Empirical results on random CNFs with up to $100,000$ variables seem to indicate that `RWalkSAT` requires linear time up to clause density $\leq$ 2.6 [25, 29]. More advanced algorithms such as WalkSAT (a Metropolis algorithm that is related to `RWalkSAT`), 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 (the empirical SAT threshold is ≈ 4.26) [28].

Random 3-CNFs: Random CNFs have 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 [21], combinatorics [18], proof complexity [9], algorithm analysis [3] and hardness of approximation [17], to mention just a few. One of the basic properties of random CNFs, is that for small density ($\Delta <$ 3.52... see [20]) almost all formulas are satisfiable, whereas for large density ($\Delta >$ 4.506... see [11]) they are almost all unsatisfiable. Another interesting property is that the threshold between satisfiability and unsatisfiability is sharp [18]. It is conjectured that a *threshold constant* exists, and empirical experiments estimate it to be $\approx$ 4.26 [16]. The analysis of SAT solving algorithms on random CNFs 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 upper bounds were given on the running time of DPLL type algorithms of increasing sophistication [1, 2, 3, 12, 22, 7, 8, 13, 20]. The best upper bound for random 3-CNF is given by the recent [20]. An exponential lower bound on a wide class of DPLL algorithms for density $\approx$ 3.8 and above, was given by [3].

Upper bounds for algorithms imply lower bounds on the satisfiability threshold, and in fact, all lower bounds on the threshold (for $k = 3$) 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 DPLL algorithms (and typically, with the exception of the recent paper [20], when proving upper bounds on these algorithms, myopic[1] versions are considered). Much less is known about non-DPLL algorithms, and local improvement ones in particular. Our result is (to the best of our knowledge) the first rigorous theoretical analysis of a non-DPLL algorithm on random CNFs.

2. Preliminaries

Random 3-CNFs For x_i a Boolean variable, a *literal* ℓ_i over x_i is either x_i or $\bar{x}_i$ (the negation of x_i), where x_i is called a *positive literal* and $\bar{x}_i$ 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 $\mathcal{C}$ (we will always assume $Vars(\mathcal{C}) = \{x_1, \ldots, x_n\}$ for some n). An *assignment* to $\mathcal{C}$ is some Boolean vector $\alpha \in \{0,1\}^n$. A literal ℓ_i is satisfied by α iff $\ell_i(\alpha_i) = 1$. We study 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

The Algorithm `RWalkSAT` is described by the following pseudo-code. $\mathcal{C}$ is the input CNF and T is the time bound, i.e. if no satisfying assignment is found in T steps we give up. We use the notation $UNSAT(\mathcal{C}, \alpha)$ for the set of clauses of $\mathcal{C}$ that are unsatisfied by α.

```
RWalkSAT(C,T)
  Select α ∈ {0,1}^n (uniformly) at random
  Initialize t = 0
  While t < T {
    If C(α) = 1
      Return ("INPUT 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."
```

3. Terminators

In this section we develop the tools needed to bound the running time of `RWalkSAT` on various interesting in-

[1] See [3] for definition and tightest analysis of myopic algorithms.

stances.

Intuition Suppose a k-CNF $\mathcal{C}$ over n variables has a satisfying assignment α such that each clause of $\mathcal{C}$ is satisfied by at least $k/2$ literals under α. In this case `RWalkSAT` will terminate in quadratic time (with high probability). The reason is that if a clause C is unsatisfied at time t by α^t, then α^t must disagree with α on at least half of the literals in C. So with probability $\geq 1/2$ we decrease the Hamming distance between our current assignment and α. If we let sim^t be the similarity of α^t and α, i.e. the number of bits that are identical in both assignments (notice $0 \leq \text{sim}^t \leq n$), then sim^t is a sub-martingale, i.e. $E(\text{sim}^t|\text{sim}^1, \ldots, \text{sim}^{t-1}) \geq \text{sim}^{t-1}$. (See e.g. [23] for more information on martingales). Standard techniques from the theory of martingales show that sim reaches n (so α^t reaches α) within $O(n^2)$ steps. One elegant example of this situation is when $\mathcal{C}$ is a satisfiable 2-CNF. Papadimitriou [24] proved quadratic upper bounds on the running time of `RWalkSAT` in this case, using the proof method outlined above.

For a general 3-CNF we don't expect a satisfying assignment to have two satisfying literals per clause. Yet all we need in order to prove good running time, is to set up a measure of similarity between α^t and some fixed satisfying assignment α such that *(i)* if sim^t reaches its maximal possible value, then $\alpha^t = \alpha$; and *(ii)* the random variables $\text{sim}^1, \text{sim}^2, \ldots$ are a sub-martingale. We achieve both these properties by giving *non-negative weights* $w_1, \ldots, w_n$ to the variables $x_1, \ldots, x_n$. Instead of similarity, we measure the *weighted similarity* between α and α^t, defined by $\text{sim}_w(\alpha, \alpha^t) \stackrel{\text{def}}{=} \sum_{\alpha_i^t = \alpha_i} w_i$. Now, suppose there exists a satisfying assignment α such that for any clause C, the expected change in sim_w, conditional on C being unsatisfied, is non-negative. Suppose further more that all w_i are bounded by a constant. Then we may conclude as above that α^t will reach its maximal value $W = \sum_i w_i$ in time $O(W^2)$.

In some cases we can do even better. We set up a system of weights such that (for any clause C) the expected change in sim_w (conditional on C being unsatisfied) is *strictly positive*. In this case the running time is *linear* in $W = \sum w_i$ (instead of quadratic). As we shall later see, such a setting of weights is possible (with high probability) for random 3-CNFs. But first we formalize our intuition.

Notation In what follows Boolean variables range over $\{-1, 1\}$. A CNF $\mathcal{C}$ with n variables and m clauses is represented by an $m \times n$ matrix $A^{\mathcal{C}}$ with $\{-1, 0, 1\}$-entries. The ith clause is represented by $A_i^{\mathcal{C}}$ (the ith row of $A^{\mathcal{C}}$) and has a -1-entry in the jth position if $\bar{x_j}$ is a literal of the ith clause of $\mathcal{C}$, a 1-entry if x_j is a literal of C_i, and is zero otherwise. Thus, if $\mathcal{C}$ is a k-CNF then the support size of each row $A_i^{\mathcal{C}}$ is at most k. A Boolean assignment is $\alpha \in \{-1, 1\}^n$ and we say α satisfies $\mathcal{C}$ iff for all $i \in [m]$

$$\langle A_i^{\mathcal{C}}, \alpha \rangle > -\left|\left|A_i^{\mathcal{C}}\right|\right|_1 \tag{1}$$

where $\langle \alpha, \beta \rangle$ is the standard inner-product over $\mathbb{R}^n$ (defined by $\sum_{i=1}^n \alpha_i \cdot \beta_i$) and $||\cdot||_1$ is the ℓ_1 norm (defined by $||\beta||_1 = \sum_{i=1}^n |\beta_i|$). It is easy to see that this definition of satisfiability coincides with the standard one.

Terminator - definition A terminator is a generalization of a satisfying assignment. On the one hand, we allow α to be any vector in $\mathbb{R}^n$, but we require a stronger satisfying condition than (1).

Definition 3.1 (Terminators) *Let $\mathcal{C}$ be a k-CNF with n variables and m clauses represented by the matrix $A^{\mathcal{C}}$. $\alpha \in \mathbb{R}^n$ is a* terminating *satisfying assignment (or* terminator*) if for all $i \in [m]$:*

$$\langle A_i^{\mathcal{C}}, \alpha \rangle \geq 1 \tag{2}$$

The termination weight of $\mathcal{C}$ is

$$\text{Term}(\mathcal{C}) \stackrel{\text{def}}{=} \min\{||\alpha||_1 \cdot ||\alpha||_\infty : \alpha \textit{ terminator for } \mathcal{C}\}$$

In case $\mathcal{C}$ has no terminator we define $\text{Term}(\mathcal{C})$ *to be ∞.*

One may think of $\text{sign}(\alpha_i)$ as the Boolean assignment to variable x_i (where $\text{sign}(\alpha_i)$ is 1 if $\alpha_i \geq 0$ and is -1 otherwise) and $|\alpha_i|$ as the weight given to x_i. Notice that if α is a terminator then the $\{-1, 1\}$-vector $\text{sign}(\alpha)$ satisfies $\mathcal{C}$. This is because by property (2) in each clause there is at least one literal that agrees in sign with α.

The decisive name given in the previous definition is justified by the following claim, which is the main theorem of this section.

Theorem 3.2 (Terminator Theorem) *If a k-CNF $\mathcal{C}$ has a terminator α, then* `RWalkSAT` *succeeds on $\mathcal{C}$ in time $O(||\alpha||_1 \cdot ||\alpha||_\infty)$ with probability $\geq 1 - \exp\left(-\Omega\left(||\alpha||_1 / ||\alpha||_\infty\right)\right)$.*

Notice that we do not claim that when `RWalkSAT` terminates, it finds the assignment $\text{sign}(\alpha)$, but rather the existence of any terminator of small weight implies short running time. We can say that `RWalkSAT` is "drawn to" α, but only when using the weighted distance measure given by $|\alpha|$. If $|\alpha_i| = 1$, this means `RWalkSAT` indeed approaches α (as is the case when each clause is satisfied by two literals). But in general, being "close" according to the weighted measure $|\alpha|$ does not imply small Hamming distance.

Proof (of Theorem 3.2): Let $\mathcal{C}$ be a k-CNF such that $\text{Term}(\mathcal{C}) = W$, and let α be a terminator with $||\alpha||_1 \cdot$

$||\alpha||_\infty = W$. Let $\beta^t \in \{-1,1\}^n$ be the sequence of assignments traversed by RWalkSAT($\mathcal{C}$) starting from the random assignment β^0, where $t \leq T = c \cdot k \cdot W$ (c will be fixed later). Let Y^t be the random variable $\langle \beta^t, \alpha \rangle$. If RWalkSAT fails to find a satisfying assignment in T steps, then

$$Y^t < ||\alpha||_1 \quad \text{for all } t < T \tag{3}$$

This is because $\langle \beta^t, \alpha \rangle = ||\alpha||_1$ implies $\beta^t = \text{sign}(\alpha)$ and $\text{sign}(\alpha)$ satisfies $\mathcal{C}$. Thus we only need to bound the probability of (3). Suppose clause C_i is picked at time t (i.e. C_i is unsatisfied by β^{t-1}). We claim the expected change in Y^t (with respect to Y^{t-1}) is precisely

$$\frac{1}{k} \cdot \langle A_i^{\mathcal{C}}, \alpha \rangle \tag{4}$$

With probability $1/k$ we flip the assignment to each literal x_j of C_i, which amounts to multiplying β_j^{t-1} by -1. Thus the expected change in Y^t is $\frac{-1}{k} \cdot \langle \beta^{t-1}|_i, \alpha \rangle$, where $\beta^{t-1}|_i$ is the restriction of β^{t-1} to support of $A_i^{\mathcal{C}}$. But C_i being unsatisfied by β^{t-1} implies $\beta^{t-1}|_i = -A_i^{\mathcal{C}}$, so equation (4) is proved. Thus by property (2) in Definition 3.1

$$E[Y^t | Y^1, \ldots, Y^{t-1}] = Y^{t-1} + \frac{1}{k} \langle A_i^{\mathcal{C}}, \alpha \rangle \geq Y^{t-1} + 1/k$$

Let $\mu^t = E[Y^t]$, and notice that by linearity of expectation, $\mu^t \geq t/k - ||\alpha||_1$ (because $\mu^0 \geq -||\alpha||_1$). We conclude that the sequence of random variables $\{X_t \stackrel{\text{def}}{=} Y^t - \mu^t : t = 1, 2, \ldots\}$ is a *martingale* and by assumption $|X_t - X_{t-1}| \leq ||\alpha||_\infty$. In order to bound (3), it suffices to bound the probability of the event "$X_T < 2\,||\alpha||_1 - T/k$" (if this event does not occur then $Y^T \geq -||\alpha||_1 + X_T \geq ||\alpha||_1$). Recalling $T = c \cdot k \cdot ||\alpha||_1 \cdot ||\alpha||_\infty$ we will pick $c > \frac{4}{k}$ so that

$$2\,||\alpha||_1 - T/k = 2\,||\alpha||_1 - c \cdot ||\alpha||_1 \cdot ||\alpha||_\infty < -\frac{ck}{2}\,||\alpha||_1 \cdot ||\alpha||_\infty$$

We now apply Azuma's inequality and get

$$\begin{aligned}
(3) \quad &\leq \quad \Pr\left[X_T < -\frac{ck}{2}\,||\alpha||_1\,||\alpha||_\infty\right] \\
&\leq \quad \exp\left(-\frac{(\frac{ck}{2}\,||\alpha||_1\,||\alpha||_\infty)^2}{2T(||\alpha||_\infty)^2}\right) \\
&\leq \quad \exp\left(-\frac{c^2k^2(||\alpha||_1)^2(||\alpha||_\infty)^2}{8ck \cdot ||\alpha||_1\,(||\alpha||_\infty)^3}\right) \\
&\leq \quad \exp\left(-\frac{ck\,||\alpha||_1}{8\,||\alpha||_\infty}\right) = \exp\left(-\Omega\left(\frac{||\alpha||_1}{||\alpha||_\infty}\right)\right)
\end{aligned}$$

The theorem is proved. ▮

4. Linear Upper Bounds on Random CNFs

In this section we show that for clause densities for which the pure literal heuristic succeeds, there exist linear weight terminators. Our current analysis uses insights into the structure of such pure CNFs, but we see no reason to believe that the terminator threshold is linked to the pure literal threshold. The main theorem of this section is the following.

Theorem 4.1 *For any $\Delta < 1.63$, there exists a constant c such that with high probability $\mathcal{C} \sim \mathbb{F}_\Delta^n$ has a terminator $\alpha \in \mathbb{R}^n$ with $||\alpha||_\infty \leq c$, hence $||\alpha||_1 \leq c \cdot n$.*

Corollary 4.2 *For any $\Delta < 1.63, \epsilon > 0$, there exists a constant c such that with high probability for $\mathcal{C} \sim \mathbb{F}_\Delta^n$,* RWalkSAT *succeeds on $\mathcal{C}$ in time $c \cdot n$ with probability $\geq 1 - \epsilon$.*

To prove our main theorem, we construct small weight terminators for pure and expanding CNFs, and then merge the two into one small weight terminator.

4.1. Terminators for Pure CNFs

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*.

Theorem 4.3 *Every r-pure k-CNF over n variables has a terminator $\alpha \in \mathbb{R}^n$ with $||\alpha||_\infty \leq k^r$ and $||\alpha||_1 \leq n \cdot (k)^r$, so* $\text{Term}(\mathcal{C}) \leq n \cdot k^{2r}$*. Moreover α is supported only on $\cup_{i=0}^{r-1} L_i$.*

Notice that invoking Theorem 3.2 we bound the running time of RWalkSAT on an r-pure k-CNF by $n \cdot k^{2r}$ (with high probability).

Proof: Let $L_0, \ldots, L_{r-1}$ be the pure literals in $\mathcal{C}_0, \ldots, \mathcal{C}_{r-1}$. Notice that $\bigcup_{j=0}^{r-1} L_j$ does not necessarily cover all variables in $\mathcal{C}$, but assigning each pure literal to 1 (i.e. if ℓ_i is pure, then set sign(α_i) = sign(ℓ_i)) and assigning the other variables arbitrarily gives a satisfying assignment α. We now deal with the weights (absolute values) of α. Fix the weight of each variable in L_j to k^{r-j}. For any variable $x_i \notin \bigcup_{j=0}^{r-1} L_j$ fix its weight to 0.

To see α is a terminator (of weight nk^r), consider any clause $C_i \in P_j$. By definition of P_j there are no literals from $L_0, \ldots, L_{j-1}$ appearing in C. Thus all literals appearing in C have weight $\leq (k)^{r-j}$. There is at least one literal $\ell_s \in C$ that has weight k^{r-j} and agrees with α_s in sign, and any literal disagreeing with α must have weight $\leq (k)^{r-j-1}$. Hence

$$\langle A_i^{\mathcal{C}}, \alpha \rangle \geq k^{r-j} - (k-1) \cdot k^{r-j-1} \geq 1$$

□

Broder, Frieze and Upfal showed that with high probability the pure literal heuristic finds a satisfying assignment for a random 3-CNF with clause density < 1.63 [12] (for a simpler analysis of the same heuristic see [22]). In particular, the following theorem follows from the work of [12]. A proof of this theorem can be found in appendix A.

Theorem 4.4 *[12] For every $\Delta < 1.63$ there exists a constant c such that with high probability $\mathcal{C} \sim \mathbb{F}_{\Delta}^{n}$ is $c \log n$-pure.*

By applying Theorems 3.2,4.3 to Theorem 4.4 we conclude that the running time of `RWalkSAT` on a random instance (with small enough clause density) is at most polynomial.

4.2. Terminators for Expanding CNFs

Our next step in proving Theorem 4.1 starts with the following Theorem, which is a combination of a result of Broder, Frieze and Upfal [12] and (the now) standard analysis of random CNFs, originating in the work of Chvátal and Szemerédi [9]. Being standard and somewhat technical, we defer its proof to appendix A.

Definition 4.5 *For $\mathcal{C}$ a CNF, we say $\mathcal{C}$ is an* (r, c)-expander *if for all $\mathcal{C}' \subseteq \mathcal{C}$ $|\mathcal{C}'| \leq r$, $|Vars(\mathcal{C}')| \geq c \cdot |\mathcal{C}'|$.*

Theorem 4.6 *For every $\Delta < 1.63$ there exists an integer d such that for $\mathcal{C} \sim \mathbb{F}_{\Delta}^{n}$, with high probability $\mathcal{C}_d$ is a $(|\mathcal{C}_d|, 7/4)$-expander, where $\mathcal{C}_d$ is the CNF remaining of $\mathcal{C}$ after removing the d outermost pure layers.*

This Theorem assures us that after removing a *constant* number of the layers from a random $\mathcal{C}$ (with small clause density), we have in hand a residual CNF $\mathcal{C}_d$, such that *any* subset of it, including all of $\mathcal{C}_d$, has a very large set of neighbors. This in turn implies the existence of small weight terminators for $\mathcal{C}_d$.

Theorem 4.7 *If $\mathcal{C}$ is an $(|\mathcal{C}|, 7/4)$-expanding 3-CNF over n variables, then $\mathcal{C}$ has a terminator $\alpha \in \mathbb{R}^n$ with $||\alpha||_\infty \leq 4$ (hence $||\alpha||_1 \leq 4n$).*

Proof: 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 $(|\mathcal{C}|, \frac{7}{4})$-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 our terminator α. For any variable x, if there exists a clause C such that $N'(C)$ has at least 3 members labeled by x, then we say x is *associated* with C, and the weight of x is $|N'(C)|$ (notice this weight is either 3 or 4). For any variable x_i associated with a clause C, set sign(α_i) to the value that satisfies C and set $|\alpha_i|$ to the weight of x_i. Set all other variables to zero. α is well defined because a variable can be associated with at most one clause. We are left with verifying that it is a terminator. This follows by a case analysis, using the fact that each clause has a dozen neighbors, and seven of them are in $N'(C_i)$. There are three cases to consider.

C_i **has at least two associated variables:** In this case, sign(α) agrees with C on at least two variables, and each variable has weight at least 3. The remaining variable has weight at most 4, so $\langle A_i^{\mathcal{C}}, \alpha \rangle \geq 6 - 4 \geq 2$.

C_i **has one associated variable of weight three:** The remaining four neighbors of $N'(C_i)$ must be evenly split between the two remaining variables of C (otherwise C_i would have two associated variables). So the remaining pair of variables of C_i have weight zero. Hence $\langle A_i^{\mathcal{C}}, \alpha \rangle = 3$.

C_i **has one associated variable of weight four:** The remaining three neighbors of $N'(C_i)$ are split between the remaining two variables. One variable has two such neighbors (and hence zero weight) and the other has one neighbor, so the weight of this literal is at most 3. Thus, $\langle A_i^{\mathcal{C}}, \alpha \rangle \geq 4 - 3 = 1$.

Theorem 4.7 follows. □

4.3. Small Weight Terminators for Random CNFs

Proof (of Theorem 4.1): By Theorem 4.6, (with high probability) $\mathcal{C}$ can be partitioned into the d outermost pure layers $\mathcal{C}' \stackrel{\text{def}}{=} \cup_{i=0}^{d-1} P_i$ and the remaining residual inner core $\mathcal{C}'' = \mathcal{C}_d$. This inner core is a $(|\mathcal{C}''|, 7/4)$-expander. We know (by Theorems 4.3, 4.7 respectively) how to construct terminators for each of these formulas, so all we need to do is merge them into a single terminator for $\mathcal{C}$.

Let α', α'' be the respective terminators of $\mathcal{C}', \mathcal{C}''$. By Theorem 4.3 α' has all its support on pure literals, which do not appear in $\mathcal{C}''$. Thus the supports of α' and α'' are disjoint. We merge the two assignments by defining α as the assignment that agrees with $9 \cdot \alpha'$ on the support of α', and agrees with α'' otherwise (the reason for multiplying α' by the scalar 9 will soon become clear) . By our previous remark (that α' and α'' have disjoint supports) α is well defined and we now prove it is a terminator.

Consider a clause $C_i \in \mathcal{C}$. If $C_i \in \mathcal{C}''$ then $\langle A_i^{\mathcal{C}}, \alpha \rangle = \langle A_i^{\mathcal{C}}, \alpha'' \rangle \geq 1$, because all literals appearing in $\mathcal{C}''$ are given zero weight by α'. Otherwise, $C_i \in \mathcal{C}'$ might have some of its (non-pure) literals in $Vars(\mathcal{C}'')$, but recall that the maximal weight of α'' is 4, so in the worst case C_i has two literals with weight 4 coming from α''. Thus $\langle A_i^{\mathcal{C}}, \alpha \rangle \geq 9 - 2 \cdot 4 = 1$. We have shown the existence of a terminator of linear total weight, and the proof of Theorem 4.1 is complete. ∎

5. Investigating the Terminator Threshold

Acknowledgements The contents of this section are due to Jeong Han Kim [Personal Communication] and we are grateful to him for allowing us to include them in this paper.

When $\mathcal{C}$ is a random CNF, Term($\mathcal{C}$) is a random variable. Since Term($\mathcal{C}$) bounds the running time of `RWalkSAT` , investigating this random variable is an interesting question. The property of having a terminator α with $||\alpha||_\infty \leq w$ is monotone with respect to addition of new clauses. Thus one can define the *terminator threshold* θ_n^w as the density for which a terminator α, $||\alpha||_\infty \leq w$ exists with probability $1/2$.

Claim 5.1 *A CNF $\mathcal{C}$ with m clauses and n variables has some terminator iff* $0 \notin$ *convex-hull*$(\{A_i^{\mathcal{C}} : i = 1, \ldots, m\})$.

Proof: Think of a terminator α as the normal of a hyperplane in $\mathbb{R}^n$ passing through zero. This hyperplane partitions $\mathbb{R}^n$ into two parts. $\langle A_i^{\mathcal{C}}, \alpha \rangle > 0$ iff the point $A_i^{\mathcal{C}}$ lies in the positive half of $\mathbb{R}^n$. Thus $\langle A_i^{\mathcal{C}}, \alpha \rangle > 0, i = 1, \ldots, m$ iff zero is not in the convex hull of the points. □

Füredi proved the following general theorem (he gave a tighter bound than presented here, but the form we quote is sufficient for our purposes). A set of points $P \subset \mathbb{R}^n$ is *symmetric* if $p \in P \Rightarrow (-p) \in P$.

Theorem 5.2 *[19] Let $\{P_n \subset \mathbb{R}^n\}_{n \in \mathbb{N}}$ be an infinite family of finite symmetric sets of points. Suppose $(2+\epsilon)n$ points are selected uniformly at random from P_n. Then*

$$\lim_{n \to \infty} \Pr[0 \notin \textit{ convex-hull of points}] = 0$$

In our case P_n is the symmetric set of $\{-1, 0, 1\}$-valued points with support size three. Thus, by Füredi's theorem when the clause density is greater than 2, whp there is no terminator. Notice this upper-bound on the terminator threshold holds for any k-CNF, even for non-constant k (e.g. $k = n$). Combining Theorem 4.1 with Füredi's Theorem gives for $k = 3$ the following bounds:

$$1.63 \leq \theta_n^\infty \leq 2$$

We leave the resolution of the terminator threshold for $k = 3$ as an interesting open problem.

For the case of 2-CNFs we can bound the terminator threshold from above by 1, because this is the satisfiability threshold for random 2-CNFs (and a terminator implies satisfiability). It seems reasonable to conjecture that for $k = 2$ the satisfiability and terminator threshold coincide. This could be used to prove that for random 2-CNFs below the satisfiability threshold, `RWalkSAT` terminates in linear time (as opposed to the quadratic upper bound guaranteed for any satisfiable 2-CNF by [24]).

6. Tightness of Terminator Based Bounds

In this section we show that the upper bound derived by the terminator method is tight, even for pure CNFs. We present pure CNFs such that the running time of `RWalkSAT` on them is exponential in the number of variables, and also lower bounded by the terminator weight.

Theorem 6.1 *For arbitrarily large n, there exist pure 3-CNFs over n variables, with total terminator weight $\geq 2^{n/2}$, and the running time of* `RWalkSAT` *on them is $2^{\epsilon n}$ for some $\epsilon > 0$.*

Proof: Use the following formula, which is a slight variation of the X-DAG contradiction used in [6].

Definition 6.2 *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 implies the existence of a terminator of weight 3^n, and it is not hard to see that any terminator must have weight 2^n at least. 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. $\square$

7. Acknowledgments

We thank Madhu Sudan for many useful discussions. We thank Bart Selman and Andrew Parkes for valuable information on the empirical results regarding RWalkSAT and Balint Virag for his help with the analysis of martingales. We thank Jon Feldman for providing the code for running the LP simulations (Appendix B), and Jeong Han Kim for allowing us to include the upper bound on the terminator threshold (Section 5) in the paper. The second author thanks Rocco Servedio, Salil Vadhan and Dimitris Achlioptas for helpful discussions.

References

[1] D. Achlioptas. Setting two variables at a time yields a new lower bound for random 3-SAT. In *Proc. 32nd STOC*, pp 28-37 (2000).

[2] D. Achlioptas. Lower Bounds for random 3-SAT via Differential Equations. In *Theoretical Computer Science* 285(1-2) (2001) 159-185.

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

[5] S. Baumer, R. Schuler. Improving a probabilistic 3-SAT algorithm by Dynamic Search and Independent Clause Pairs. ECCC report No. 10, 2003. Also presented at SAT 2003.

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

[7] M.T. Chao, J. Franco. Probabilistic Analysis of two heuristics for the 3-Satisfiability problem. In *Infor. Sci.* 51 (1990). 289-314.

[8] V. Chvátal, B. Reed. Mick Gets Some (the odds are on his side). In *FOCS '92* pp 620-627.

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

[10] E. Danstin, A. Goerdt, E. A. Hirsch, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schoning. A Deterministic $2 - \frac{2}{k+1}$ algorithm for k-SAT based on local search. In *Theoretical Computer Science*, 223 (1-2):1-72, 1999.

[11] O. Dubois, Y. Boufkhad, J. Mandler. Typical random 3-SAT formulae and the satisfiability threshold. In *Proc. 11th SODA* , pp. 126-127 (2000). Available at `ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/2003/TR03-007/index.html`

[12] 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.

[13] A. Frieze, S. Suen. Analysis of two simple heuristics for random instances of k-SAT. In *J. Algorithms* 20 (1996) 312-355.

[14] T. Hofmeister, U. Schoning, R. Schuler O. Watanabe. Probabilistic 3-SAT Algorithm Further Improved. In *Proc. 19th STACS*, LNCS 2285:193-202, 2002.

[15] K. Iwama, S. Tamaki. Improved Bounds for 3-SAT. In *ECCC report No. 53*, 2003.

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

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

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

[19] Z. Füredi. Random Polytopes in the d-Dimensional Cube. In *Discrete Computational Geometry* (1) pp. 315-319 (1986)

[20] A. 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).

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

[22] 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.

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

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

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

[26] D. Rolf 3-SAT in $RTIME(O(1.32793^n))$ - Improving Randomized Local Search by Initializing Strings of 3-Clauses. In *ECCC report No. 54*, 2003.

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

[28] Selman, B. Personal Communication.

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

[30] 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. Proofs

In this section we prove theorems 4.4 and 4.6. Our starting point is the following theorem and lemma proved implicitly in [12]. The lemma is a slight generalization of lemma 4.4 in [12], so we provide its proof. (The original lemma 4.4 of [12] only needed expansion factor of $3/2$, whereas we need a constant fraction more than $3/2$. The proof is essentially the same).

Theorem A.1 *[12] For every $\Delta < 1.63$ there exists an integer d such that with high probability for $\mathcal{C} \sim \mathbb{F}_\Delta^n$, $|\mathcal{C}_d| \leq \frac{n}{600\Delta^2}$.*

Lemma A.2 *[12] 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 (of Lemma A.2): 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 $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$.

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

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

Notice that if $\mathcal{C}_d$ is a $(|\mathcal{C}_d|, \frac{3}{2}+\epsilon)$-expander then every subset of $\mathcal{C}_d$ (including $\mathcal{C}_d$ itself) has at least ϵ *unique neighbors* (i.e. literals appearing in exactly one clause), and these unique neighbors are pure. Thus, $\mathcal{C}_d$ is $O(\log n)$-pure. Hence $\mathcal{C}$ is $O(\log n)$-pure (remember d is a constant) and this proves theorem 4.4. In order to prove theorem 4.6 we need the following lemma from [9].

Lemma A.3 *[9] 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 A.3, for $\Delta = 1.63$ and $c = 7/4$. By theorem 4.4, $|\mathcal{C}_d| \leq n/(600\Delta_0^2)$, for some constant d. By lemma A.2, $\mathcal{C}_d$ is a $(|\mathcal{C}_d|, \epsilon)$-boundary expander, for some $\epsilon > 0$. Remove an additional d' layers from $\mathcal{C}$ (each containing at least an $\epsilon/3$ fraction of the remaining clauses) so that $|\mathcal{C}_{d+d'}| \leq \delta n$, and by lemma A.3 this remaining CNF is (with high probability) a $(|\mathcal{C}_{d+d'}, 7/4)$-expander. This proves theorem 4.6.

B. Terminator Threshold - Empirical Results via Linear Programming

Acknowledgements The code used in this section was written by Jon Feldman. We thank him for assisting us with the empirical tests and allowing us to include these results in our paper.

A nice property of a terminator is that if it exists, it can be found by linear programming in polynomial time, and if it doesn't exist, we can efficiently find a proof of this fact.

Since by definition a terminator contains a satisfying assignment, this provides us with an efficient approach for SAT-solving using linear programming. As far as we know, this is the first use of linear programming in this context.

Empirical Threshold Lower Bound As discussed earlier (Section 5), we know that the terminator threshold must lie between 1.63 and 2 (for 3-CNFs). Empirically it seems the terminator threshold is at least 1.8. This is based on the fact that only 5% of the inputs we tested failed to have a terminator, out of a sample of 120 random CNFs with 4,000-4,100 variables each. On instances with a smaller number of variables (450-1,000) the failure rate is significantly higher (around 25%). We believe this is an effect of small instance size and asymptotically at density 1.8 a terminator exists.

Terminator weight vs. Running Time There seems to be a large gap between the empirical terminator weight (that bounds from above the running time of `RWalkSAT`) and the actual running time of `RWalkSAT`. Whereas the existence of a terminator is decidable in polynomial time, finding the *minimal weight* terminator seems to be a much harder problem. Thus, it is not clear if the empirical gap displayed below (between terminator weight and `RWalkSAT` running time) really exists, or is merely an artifact of our limited computational power.

Recall the data of figure 1 showing an empirical running time of $\leq n$ on instances with n variables (and clause density ≤ 2.5). The following figure 2 shows an empirical upper bound on the average termination weight for instances with 450-1,000 variables, at clause densities $1.65, 1.7$ and 1.75. For each instance size between 450 and $1,000$ (jumps of 50 variables), we tested thirty random instances and their average termination weight is plotted in figure 2.

Experimental Methods For each instance $\mathcal{C}$ we searched for a small weight terminator as follows. First we searched for a solution for the linear program $A^{\mathcal{C}} \cdot \alpha > 0$ with a *random* objective function. Having found a terminator α (that is minimal with respect to the random objective function), we minimized its ℓ_1-norm by solving the linear program with the *new* objective function $\mathrm{sign}(\alpha)$. For each instance $\mathcal{C}$ we repeated this two-phase process three times, each time starting with a fresh random objective function. Finally we took the minimum value (ℓ_1-norm) as an upper bound on the terminator weight for $\mathcal{C}$.

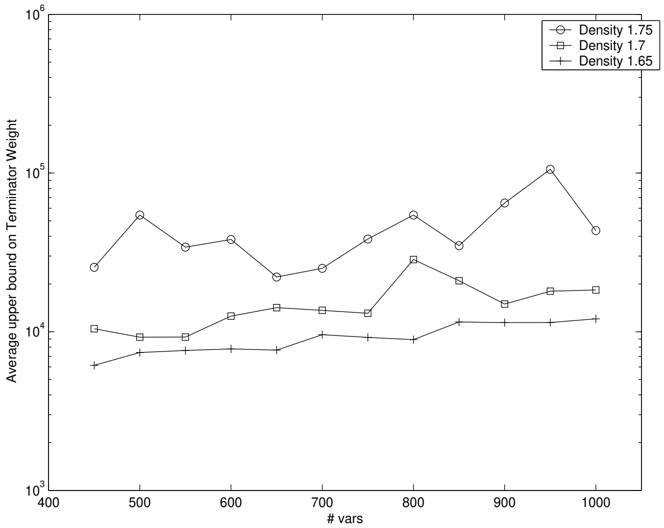

Figure 2. Empirical upper bounds on the average termination weight. Average taken over thirty random instance per variable size.