A Better Algorithm for Random k-SAT

Amin Coja-Oghlan*

University of Edinburgh, School of Informatics, Edinburgh EH8 9AB, UK
acoghlan@inf.ed.ac.uk

Abstract. Let $\boldsymbol{\Phi}$ be a uniformly distributed random k-SAT formula with n variables and m clauses. We present a polynomial time algorithm that finds a satisfying assignment of $\boldsymbol{\Phi}$ with high probability for constraint densities $m/n < (1-\varepsilon_k)2^k \ln(k)/k$, where $\varepsilon_k \to 0$. Previously no efficient algorithm was known to find solutions with non-vanishing probability beyond $m/n = 1.817 \cdot 2^k/k$ [Frieze and Suen, Journal of Algorithms 1996].

1 Introduction

The k-SAT problem is well known to be NP-hard for $k \geq 3$. But this merely indicates that no algorithm can solve *all* possible inputs efficiently. Therefore, a significant amount of research has been conducted on *heuristics* for k-SAT, i.e., algorithms that solve 'most' inputs efficiently (where the meaning of 'most' depends on the scope of the respective paper). While some heuristics for k-SAT are very sophisticated, virtually all of them are based on at least one of the following basic paradigms.

Pure literal rule. If a variable x occurs only positively (resp. negatively) in the formula, set it to true (resp. false). Simplify the formula by substituting the newly assigned value for x and repeat.

Unit clause propagation. If the formula contains a clause that consists of only a single literal ('unit clause'), then set the underlying variable so as to satisfy this clause. Simplify and repeat.

Walksat. Initially pick a random assignment. Then repeat the following. While there is an unsatisfied clause, pick one at random, pick a variable occurring in the chosen clause randomly, and flip its value.

Backtracking. Assign a variable x, simplify the formula, and recurse. If the recursion fails to find a satisfying assignment, assign x the opposite value and recurse.

Heuristics based on these paradigms can be surprisingly successful (given that k-SAT is NP-hard) on certain types of inputs. However, it remains remarkably simple to generate formulas that elude all known algorithms/heuristics. Indeed, the simplest conceivable type of *random* instances does the trick: let $\boldsymbol{\Phi}$ denote a

* Supported by EPSRC grant EP/G039070/1.

S. Albers et al. (Eds.): ICALP 2009, Part I, LNCS 5555, pp. 292–303, 2009.

k-SAT formula over the variable set $V = \{x_1, \ldots, x_n\}$ that is obtained by choosing m clauses uniformly at random and independently from the set of all $(2n)^k$ possible clauses. Then for a large regime of densities m/n satisfying assignments are known to exist due to non-constructive arguments, but no efficient algorithm is known to find one.

To be precise, keeping k fixed and letting $m = \lceil rn \rceil$ for a fixed $r > 0$, we say that $\boldsymbol{\Phi}$ has some property *with high probability* ('w.h.p.') if the probability of the property tends to one as $n \to \infty$. Via the (non-algorithmic) second moment method [3,4] it can be shown that $\boldsymbol{\Phi}$ has a satisfying assignment w.h.p. if $m/n < (1 - \varepsilon_k)2^k \ln 2$. Here ε_k tends to 0 for large k. On the other hand, a simple first moment argument shows that no satisfying assignment exists w.h.p. if $m/n > 2^k \ln 2$. In summary, the threshold for $\boldsymbol{\Phi}$ being satisfiable is asymptotically $2^k \ln 2$.

Yet for densities m/n beyond $c \cdot 2^k/k$, where c is a constant (independent of k), no algorithm has been known to find a satisfying assignment in polynomial time with a probability that does not tend to zero. Using merely the Unit Clause rule yields a linear time algorithm that succeeds up to $m/n = c \cdot 2^k/k$ with $c \sim \mathrm{e}/2 \approx 1.36$. The best previous rigorous result, based on a somewhat more involved algorithm, achieved $c \sim 1.817$ (cf. Section 2). Conversely, many algorithms, including Pure Literal, Unit Clause, and DPLL, are known to fail or exhibit an exponential running time beyond $c \cdot 2^k/k$. There is experimental evidence that the same is true of Walksat. In effect, devising an algorithm to solve random formulas w.h.p. for densities m/n up to $2^k\omega(k)/k$ for *any* (howsoever slowly growing) $\omega(k) \to \infty$ has been a prominent open problem [3,4,8,15].

Theorem 1. *There are a sequence $\varepsilon_k \to 0$ and a polynomial time algorithm* `Fix` *such that* `Fix` *applied to a random formula $\boldsymbol{\Phi}$ with $m/n \leq (1-\varepsilon_k)2^k \ln(k)/k$ outputs a satisfying assignment w.h.p.*

`Fix` is a deterministic local search algorithm and runs in time $O(n+m)^{3/2}$. The recent paper [2] provides evidence that the density $m/n = 2^k \ln(k)/k$ may be a barrier for (at least) a large class of algorithms to find satisfying assignments in polynomial time. Hence, Theorem 1 may mark, at least up to the precise second order term hidden in the ε_ks, the end of the algorithmic road for random k-SAT. To explain this, we need to discuss a concept that originates from statistical physics.

A digression: replica symmetry breaking. For the last decade random k-SAT has been studied by statistical physicists via sophisticated, insightful, but mathematically non-rigorous techniques from the theory of spin glasses. Their results suggest that below the threshold density $2^k \ln 2$ for the existence of satisfying assignments various other phase transitions take place that affect the performance of algorithms.

To us the most important one is the *dynamic replica symmetry breaking* (dRSB) transition. Let $S(\boldsymbol{\Phi}) \subset \{0,1\}^V$ be the set of all satisfying assignments of the random formula $\boldsymbol{\Phi}$. Very roughly speaking, according to the dRSB hypothesis there is a density r_{RSB} such that for $m/n < r_{RSB}$ the correlations that shape the set $S(\boldsymbol{\Phi})$ are purely local, whereas for densities $m/n > r_{RSB}$ long range correlations occur. Furthermore, $r_{RSB} \sim 2^k \ln(k)/k$.

Confirming and elaborating on this hypothesis, we recently established a good part of the dRSB phenomenon rigorously [2]. In particular, we proved that there is a sequence $\varepsilon_k \to 0$ such that for $m/n > (1+\varepsilon_k)2^k \ln(k)/k$ the values that the solutions $\sigma \in S(\boldsymbol{\Phi})$ assign to the variables are mutually heavily correlated in the following sense. Let us call a variable x *frozen* in a satisfying assignment σ if any satisfying assignment τ such that $\sigma(x) \neq \tau(x)$ is at Hamming distance at least $\Omega(n)$ from σ. Then for $m/n > (1+\varepsilon_k)2^k \ln(k)/k$ in all but a $o(1)$-fraction of all solutions $\sigma \in S(\boldsymbol{\Phi})$ all but an ε_k-fraction of the variables are frozen w.h.p., where $\varepsilon_k \to 0$.

This suggests that on random formulas with density $m/n > (1+\varepsilon_k)2^k \ln(k)/k$ local search algorithms (such as Pure Literal, Unit Clause, or Walksat) are unlikely to succeed. For think of the *factor graph*, whose vertices are the variables and the clauses, and where a variable is adjacent to all clauses in which it occurs. Then a local search algorithm assigns a value to a variable x on the basis of the values of variables that have distance $O(1)$ from x in the factor graph. But in the random formula $\boldsymbol{\Phi}$ with $m/n > (1+\varepsilon_k)2^k \ln(k)/k$ assigning one variable x is likely to impose constraints on the values that can be assigned to variables at distance $\Omega(\ln n)$ from x in the factor graph (due to the occurrence of frozen variables).

The above discussion applies to 'large' values of k (say, $k \geq 8$). In fact, non-rigorous arguments as well as experimental evidence [5] suggest that the picture is quite different and rather more complicated for 'small' k (say, $k = 3, 4, 5$). In this case the various phenomena that occur at (or very near) the point $2^k \ln(k)/k$ for $k \geq 8$ appear to happen at vastly different points in the satisfiable regime, and in a different order. To keep matters as simple as possible we focus on 'large' k in this paper.

Notation. We let $V = V_n = \{x_1, \ldots, x_n\}$ be a set of propositional variables. For a set $Z \subset V$ let $\bar{Z} = \{\bar{x} : x \in Z\}$ contain the corresponding set of negative literals. If l is a literal, then $|l|$ signifies the underlying variable. Let $[\mu] = \{1, 2, \ldots, \mu\}$ for integers μ.

Let $\Omega_k(n, m)$ be the set of all k-SAT formulas over V. Throughout the paper we denote a random element of $\Omega_k(n, m)$ by $\boldsymbol{\Phi}$; unless otherwise specified, $\boldsymbol{\Phi}$ is uniformly distributed. We use the letter Φ to denote specific (i.e., non-random) elements of $\Omega_k(n, m)$. Further, Φ_i denotes the ith clause of Φ, and Φ_{ij} is the jth literal of Φ_i.

2 Related Work

Quite a few papers deal with efficient algorithms for random k-SAT, contributing either rigorous results, non-rigorous evidence based on physics arguments, or experimental evidence. Table 1 summarizes the part of this work that is most relevant to us. The best rigorous result prior to this work is due to Frieze and Suen [11], who proved that 'SCB' succeeds for densities $\eta_k 2^k/k$, where η_k increases to 1.817 as $k \to \infty$. SCB combines the shortest clause rule, which is a generalization of Unit Clause, with (very limited) backtracking.

Table 1. Algorithms for random k-SAT

Algorithm	Density $m/n < \cdots$	Success probability	Ref., year
Pure Literal	$o(1)$ as $k \to \infty$	w.h.p.	[14], 2006
Walksat, rigorous	$\frac{1}{6} \cdot 2^k/k^2$	w.h.p.	[9], 2009
Walksat, non-rigorous	$2^k/k$	w.h.p.	[16], 2003
Unit Clause	$\frac{1}{2}\left(\frac{k-1}{k-2}\right)^{k-2} \cdot \frac{2^k}{k}$	$\Omega(1)$	[7], 1990
Shortest Clause	$\frac{1}{8}\left(\frac{k-1}{k-3}\right)^{k-3} \frac{k-1}{k-2} \cdot \frac{2^k}{k}$	w.h.p.	[8], 1992
SCB	$\sim 1.817 \cdot \frac{2^k}{k}$	w.h.p.	[11], 1996
BP+decimation (non-rigorous)	$\mathrm{e} \cdot 2^k/k$	w.h.p.	[15], 2007

Montanari, Ricci-Tersenghi, and Semerjian [15] provide evidence that Belief Propagation guided decimation may succeed up to density $\mathrm{e} \cdot 2^k/k$. This algorithm is based on a very different paradigm than the others mentioned in Table 1. The basic idea is to run a message passing algorithm ('Belief Propagation') to compute for each variable the marginal probability that this variable takes the value true/false in a uniformly random satisfying assignment. Then, the decimation step selects a variable, assigns it the value true/false with the corresponding marginal probability, and simplifies the formula. Ideally, repeating this procedure will lead to a satisfying assignment, provided that Belief Propagation keeps yielding the correct marginals. Proving (or disproving) this remains a major open problem.

Survey Propagation is a modification of Belief Propagation that aims to approximate the marginal probabilities induced by a particular (non-uniform) probability distribution on the set of satisfying assignments [6]. It can be combined with a decimation procedure as well to obtain a heuristic for finding a satisfying assignment. Analyzing Survey Propagation guided decimation is a further outstanding open problem.

The discussion so far concerns the case of general k. In addition, a large number of papers deal with the case $k = 3$. Flaxman [10] provides a survey. Currently the best rigorously analyzed algorithm for random 3-SAT is known to succeed up to $m/n = 3.52$ [12,13]. This is also the best known lower bound on the 3-SAT threshold. Non-rigorous arguments suggest the threshold to be ≈ 4.267 [6]. As mentioned earlier, there is non-rigorous evidence that the structure of the set of all satisfying assignment evolves differently in random 3-SAT than in random k-SAT for 'large' k. This may be why experiments suggest that Survey Propagation guided decimation for 3-SAT succeeds for densities m/n up to 4.2 [6].

3 The Algorithm `Fix`

In this section we present the algorithm `Fix`. To establish Theorem 1 we will prove the following: for any $0 < \varepsilon < 0.1$ there is $k_0 = k_0(\varepsilon) > 3$ such that for all $k \geq k_0$ the algorithm `Fix` outputs a satisfying assignment w.h.p. when applied

to $\boldsymbol{\Phi}$ with $m = \lfloor n \cdot (1-\varepsilon) 2^k k^{-1} \ln k \rfloor$. Thus, we assume that k exceeds some large enough number k_0 depending on ε only. In addition, we assume throughout that $n > n_0$ for some large $n_0 = n_0(\varepsilon, k)$. We set $\omega = (1-\varepsilon)\ln k$ and $k_1 = \lceil k/2 \rceil$.

When applied to a k-SAT instance Φ the algorithm basically tries to 'fix' the all-true assignment by setting 'a few' variables $Z \subset V$ to false so as to satisfy all clauses. Obviously, the set Z will have to contain one variable from each clause consisting of negative literals only. The key issue is to pick 'the right' variables. To this end, the algorithm goes over the all-negative clauses in the natural order. If the present all-negative clause Φ_i does not contain a variable from Z yet, `Fix` (tries to) identify a 'safe' variable in Φ_i, which it then adds to Z. Here 'safe' means that setting the variable to false does not create new unsatisfied clauses. More precisely, we say that a clause Φ_i is *Z-unique* if Φ_i contains exactly one positive literal from $V \setminus Z$ and no negative literal whose underlying variable is in Z. Moreover, $x \in V \setminus Z$ is *Z-unsafe* if it occurs positively in a Z-unique clause, and *Z-safe* if this is not the case. Then in order to fix an all-negative clause Φ_i we prefer Z-safe variables.

To implement this idea, `Fix` proceeds in three phases. Phase 1 performs the operation described in the previous paragraph: try to identify a Z-safe variable in each all-negative clause. Of course, not every all-negative clause will contain one. In this case `Fix` just picks the variable in position k_1. This entails that the assignment constructed in Phase 1 will not satisfy *all* clauses. However, we will prove that the number of unsatisfied clauses is very small, and the purpose of Phases 2 and 3 is to deal with them. Before we come to this, let us describe Phase 1 precisely.

Algorithm 2. `Fix`(Φ)
Input: A k-SAT formula Φ. *Output:* Either a satisfying assignment or 'fail'.

1a. Let $Z = \emptyset$.
1b. For $i = 1, \ldots, m$ do
1c. If Φ_i is all-negative and contains no variable from Z
1d. If there is $1 \leq j < k_1$ such that $|\Phi_{ij}|$ is Z-safe, then pick the least such j and add $|\Phi_{ij}|$ to Z.
1e. Otherwise add $|\Phi_{i\,k_1}|$ to Z.

Let σ_Z be the assignment that sets all variables in $V \setminus Z$ to true and all variables in Z to false.

Proposition 3. *At the end of the first phase of* `Fix`($\boldsymbol{\Phi}$) *the following statements are true w.h.p.*

1. *We have* $|Z| \leq 4nk^{-1} \ln \omega$.
2. *At most* $(1 + \varepsilon/3)\omega n$ *clauses are Z-unique.*
3. *At most* $\exp(-k^{\varepsilon/8})n$ *clauses are unsatisfied under* σ_Z.

Since the probability that a random clause is all-negative is 2^{-k}, under the all-true assignment $(1+o(1))2^{-k}m \sim \omega n/k$ clauses are unsatisfied w.h.p. Hence, the outcome σ_Z of Phase 1 is already a lot better than the all-true assignment w.h.p.

Phase 2 deals with the clauses that are unsatisfied under σ_Z. The general plan is similar to Phase 1: we (try to) identify a set Z' of 'safe' variables that can be used to satisfy the σ_Z-unsatisfied clauses without 'endangering' further clauses. More precisely, we say that a clause Φ_i is (Z, Z')-*endangered* if there is no $1 \le j \le k$ such that the literal Φ_{ij} is true under σ_Z and $|\Phi_{ij}| \in V \setminus Z'$. In words, Φ_i is (Z, Z')-endangered if it is unsatisfied under σ_Z or it relies on one of the variables in Z' to be satisfied. Call Φ_i (Z, Z')-*secure* if it is not (Z, Z')-endangered. Phase 2 will construct a set Z' such that for all $1 \le i \le m$ either Φ_i is (Z, Z')-secure, or there are at least three indices $1 \le j \le k$ such that $|\Phi_{ij}| \in Z'$. To achieve this, we say that a variable x is (Z, Z')-*unsafe* if $x \in Z \cup Z'$ or there are indices $(i, l) \in [m] \times [k]$ such that the following two conditions hold:

a. For all $j \neq l$ we have $\Phi_{ij} \in Z \cup Z' \cup \overline{V \setminus Z}$.
b. $\Phi_{il} = x$.

(In words, x occurs positively in Φ_i, and all other literals of Φ_i are either positive but in $Z \cup Z'$ or negative but not in Z.) Otherwise we call x (Z, Z')-*safe*. `Fix` greedily tries to add as few (Z, Z')-unsafe variables to Z' as possible.

2a. Let Q consist of all $i \in [m]$ such that Φ_i is unsatisfied under σ_Z. Let $Z' = \emptyset$.
2b. While $Q \neq \emptyset$
2c. Let $i = \min Q$.
2d. If there are indices $k_1 < j_1 < j_2 < j_3 \le k - 5$ such that $|\Phi_{ij_l}|$ is (Z, Z')-safe for $l = 1, 2, 3$,
pick the lexicographically first such sequence and add the variables $|\Phi_{ij_1}|, |\Phi_{ij_2}|, |\Phi_{ij_3}|$ to Z'.
2e. else
let $k - 5 < j_1 < j_2 < j_3 \le k$ be the lexicographically first sequence such that $|\Phi_{ij_l}| \notin Z'$ and add $|\Phi_{ij_l}|$ to Z' $(l = 1, 2, 3)$.
2f. Let Q be the set of all (Z, Z')-endangered clauses that contain less than 3 variables from Z'.

Note that the While-loop gets executed at most $n/3$ times, because Z' gains three new elements in each iteration. Actually the final set Z' is fairly small w.h.p.:

Proposition 4. *The set Z' obtained in Phase 2 of `Fix`(Φ) has size $|Z'| \le nk^{-12}$ w.h.p.*

After completing Phase 2, `Fix` is going to set the variables in $V \setminus (Z \cup Z')$ to true and the variables in $Z \setminus Z'$ to false. This will satisfy all (Z, Z')-secure clauses. In order to satisfy the (Z, Z')-endangered clauses as well, `Fix` needs to set the variables in Z' appropriately. Since each (Z, Z')-endangered clause contains three variables from Z', this is essentially equivalent to solving a 3-SAT problem, in which Z' is the set of variables. As we shall see, w.h.p. the resulting formula is sufficiently sparse for the following 'matching heuristic' to succeed: set up a bipartite graph $G(\Phi, Z, Z')$ whose vertex set consists of the (Z, Z')-endangered clauses and the set Z'. Each (Z, Z')-endangered clause is adjacent to the variables from Z' that occur in it. If M is a matching in $G(\Phi, Z, Z')$ that

covers all (Z, Z')-endangered clauses, we construct an assignment $\sigma_{Z,Z',M}$ as follows: for each variable $x \in V$ we define

$$\sigma_{Z,Z',M}(x) = \begin{cases} \text{false} & \text{if } x \in Z \setminus Z' \\ \text{false} & \text{if } \{\Phi_i, x\} \in M \text{ for some } i \text{ and } x \text{ occurs negatively in } \Phi_i, \\ \text{true} & \text{otherwise.} \end{cases}$$

To be precise, Phase 3 proceeds as follows.

3. If $G(\Phi, Z, Z')$ has a matching that covers all (Z, Z')-endangered clauses, then compute an (arbitrary) such matching M and output $\sigma_{Z,Z',M}$. If not, output 'fail'.

Proposition 5. *W.h.p. $G(\boldsymbol{\Phi}, Z, Z')$ has a matching that covers all (Z, Z')-endangered clauses.*

Proof of Theorem 1. `Fix` is clearly a deterministic algorithm with running time $O(n+m)^{3/2}$ (if we use the Hopcroft-Karp algorithm to compute the matching in Phase 3). It remains to show that `Fix`$(\boldsymbol{\Phi})$ outputs a satisfying assignment w.h.p. By Proposition 5 Phase 3 will find a matching M that covers all (Z, Z')-endangered clauses w.h.p., and thus the output will be the assignment $\sigma = \sigma_{Z,Z',M}$ w.h.p. Assume that this is the case. Then σ sets all variables in $Z \setminus Z'$ to false and all variables in $V \setminus (Z \cup Z')$ to true, thereby satisfying all (Z, Z')-secure clauses. Furthermore, for each (Z, Z')-endangered clause $\boldsymbol{\Phi}_i$ there is an edge $\{\boldsymbol{\Phi}_i, |\boldsymbol{\Phi}_{ij}|\}$ in M. If $\boldsymbol{\Phi}_{ij}$ is negative, then $\sigma(|\boldsymbol{\Phi}_{ij}|) = \text{false}$, and if if $\boldsymbol{\Phi}_{ij}$ is positive, then $\sigma(\boldsymbol{\Phi}_{ij}) = \text{true}$. In either case σ satisfies $\boldsymbol{\Phi}_i$. □

In the next section we sketch the analysis of Phase 1, i.e., the proof of Proposition 3. The analysis of Phase 2 (Proposition 4) is based on very similar ideas (details omitted). Furthermore, the proof of Proposition 5 combines ideas from the analysis of Phase 1 with a first moment argument.

4 Analyzing Phase 1

In this section we let $0 < \varepsilon < 0.1$ and assume that $k \geq k_0$ for a sufficiently large $k_0 = k_0(\varepsilon)$. Moreover, we assume that $m = \lfloor (1-\varepsilon) 2^k k^{-1} \ln k \rfloor$ and that $n > n_0$ for some large enough $n_0 = n_0(\varepsilon, k)$. Let $\omega = (1-\varepsilon)\ln k$ and $k_1 = \lceil k/2 \rceil$.

It is worthwhile giving a brief intuitive explanation as to why Phase 1 'works'. Namely, let us just consider the *first* all-negative clause $\boldsymbol{\Phi}_i$ of the random input formula. Assume that $i = 1$. If we condition on $\boldsymbol{\Phi}_1$ being all-negative, the k-tuple of variables $(|\boldsymbol{\Phi}_{1j}|)_{j \in [k]}$ is uniformly distributed. Furthermore, at this point $Z = \emptyset$. Hence, a variable x is Z-safe unless it occurs as the unique positive literal in some clause. For any x the expected number of such clauses is $k 2^{-k} m/n \sim \omega$ (for in each clause there are k slots where to put x, the probability that x occurs in any slot is $1/n$, and the probability that x occurs positively and all other literals are negative is 2^{-k}). In fact, for each variable the number of such clauses is asymptotically Poisson. Consequently, the probability that x is Z-safe is $\exp(-\omega)$. Returning to the clause $\boldsymbol{\Phi}_1$, we conclude that the *expected* number of

indices $1 \leq j \leq k_1$ such that $|\boldsymbol{\Phi}_{1j}|$ is Z-safe is $k_1 \exp(-\omega)$. Since $\omega = (1-\varepsilon)\ln k$, we have $k_1 \exp(-\omega) \geq k^{\varepsilon}/3$. Indeed, the number of indices $1 \leq j \leq k_1$ so that $|\boldsymbol{\Phi}_{1j}|$ is Z-safe is binomially distributed, and hence the probability that there is no Z-safe $|\boldsymbol{\Phi}_{1j}|$ is at most $\exp(-k^{\varepsilon}/3)$. Thinking of k 'large' (in terms of ε), we see that there is a good chance that $\boldsymbol{\Phi}_1$ can be satisfied by setting some variable to false without creating any new unsatisfied clauses. Of course, this argument only applies to the first all-negative clause, and the challenge lies in dealing with the stochastic dependencies that arise in the course of the execution of the algorithm.

To this end, we need to investigate how the set Z computed in Phase 1 evolves over time. Thus, we will analyze the execution of Phase 1 as a stochastic process, in which Z corresponds to a sequence $(Z_t)_{t\geq 0}$ of sets. The time parameter t is the number of all-negative clauses for which either Step 1d or 1e has been executed. We will represent the execution of Phase 1 on input $\boldsymbol{\Phi}$ by a sequence of (random) maps

$$\pi_t : [m] \times [k] \to \{-1, 1\} \cup V \cup \bar{V}.$$

The map π_t is meant to capture the information that has determined the first t steps of the process. If $\pi_t(i,j) = 1$ (resp. $\pi_t(i,j) = -1$), then `Fix` has only taken into account that $\boldsymbol{\Phi}_{ij}$ is a positive (negative) literal, but not what the underlying variable is. If $\pi_t(i,j) \in V \cup \bar{V}$, then `Fix` has revealed the actual literal $\boldsymbol{\Phi}_{ij}$.

Let us define the sequence $\pi_t(i,j)$ precisely. Let $Z_0 = \emptyset$. Moreover, let U_0 be the set of all i such that there is exactly one j such that $\boldsymbol{\Phi}_{ij}$ is positive. Further, define $\pi_0(i,j)$ for $(i,j) \in [m] \times [k]$ as follows. If $i \in U_0$ and $\boldsymbol{\Phi}_{ij}$ is positive, then let $\pi_0(i,j) = \boldsymbol{\Phi}_{ij}$. Otherwise, let $\pi_0(i,j)$ be 1 if $\boldsymbol{\Phi}_{ij}$ is a positive literal and -1 if $\boldsymbol{\Phi}_{ij}$ is a negative literal. In addition, for $x \in V$ let $U_0(x)$ be the number of $i \in U_0$ such that x occurs positively in $\boldsymbol{\Phi}_i$. For $t \geq 1$ we define π_t as follows.

PI1. If there is no index $i \in [m]$ such that $\boldsymbol{\Phi}_i$ is all-negative but contains no variable from Z_{t-1}, the process stops. Otherwise let ϕ_t be the smallest such index.

PI2. If there is $1 \leq j < k_1$ such that $U_{t-1}(|\boldsymbol{\Phi}_{\phi_t j}|) = 0$, then choose the smallest such index; otherwise let $j = k_1$. Let $z_t = \boldsymbol{\Phi}_{\phi_t j}$ and $Z_t = Z_{t-1} \cup \{z_t\}$.

PI3. Let U_t be the set of all $i \in [m]$ such that $\boldsymbol{\Phi}_i$ is Z_t-unique. For $x \in V$ let $U_t(x)$ be the number of indices $i \in U_t$ such that x occurs positively in $\boldsymbol{\Phi}_i$.

PI4. For any $(i,j) \in [m] \times [k]$ let

$$\pi_t(i,j) = \begin{cases} \boldsymbol{\Phi}_{ij} & \text{if } (i = \phi_t \wedge j \leq k_1) \vee |\boldsymbol{\Phi}_{ij}| \in Z_t \\ & \qquad \vee (i \in U_t \wedge \pi_0(i,j) = 1), \\ \pi_{t-1}(i,j) & \text{otherwise.} \end{cases}$$

Let T be the total number of iterations before the process stops and define $\pi_t = \pi_T$, $Z_t = Z_T$, $U_t = U_T$, $U_t(x) = U_T(x)$, $\phi_t = z_t = 0$ for all $t > T$.

The process mirrors Phase 1 of `Fix` as follows. Step **PI1** selects the least index ϕ_t such that clause $\boldsymbol{\Phi}_{\phi_t}$ is all-negative but contains none of the variables Z_{t-1} that have been selected to be set to false so far. In terms of `Fix`, this corresponds to fast-forwarding to the next execution of Steps 1d–e. Since $U_{t-1}(x)$

is the number of Z_{t-1}-unique clauses in which variable x occurs positively, **PI2** applies the same rule as steps 1d–e of `Fix` to select the new element z_t to be included in the set Z_t. Step **PI3** then 'updates' the numbers $U_t(x)$. Finally, step **PI4** sets up the map π_t to represent the information that has guided the process so far: we reveal the first k_1 literals of the current clause $\boldsymbol{\Phi}_{\phi_t}$, all occurrences of the variable z_t, and all positive literals of Z_t-unique clauses.

The process **PI1**–**PI4** can be applied to any concrete k-SAT formula Φ (rather than the random $\boldsymbol{\Phi}$). It then yields a sequence $\pi_t[\Phi]$ of maps, variables $z_t[\Phi]$, etc. For each integer $t \geq 0$ we define an equivalence relation $\equiv_t$ on the set $\Omega_k(n,m)$ of k-SAT formulas by letting $\Phi \equiv_t \Psi$ iff $\pi_s[\Phi] = \pi_s[\Psi]$ for all $0 \leq s \leq t$. Let $\mathcal{F}_t$ be the σ-algebra generated by the equivalence classes of $\equiv_t$. Then (loosely speaking) a random variable $X(\boldsymbol{\Phi})$ is $\mathcal{F}_t$-measurable if its value is determined by time t.

Fact 6. For any $t \geq 0$ the random map π_t, the random variables ϕ_{t+1}, z_t, the random sets U_t and Z_t, and the random variables $U_t(x)$ for $x \in V$ are $\mathcal{F}_t$-measurable.

The first t steps of the process **PI1**–**PI4** are only driven by the information encoded in the map π_t, Hence, for (i,j) such that $\pi_t(i,j) = \pm 1$ the process has only taken into accout the *sign* of the literal $\boldsymbol{\Phi}_{ij}$ and the fact that $|\boldsymbol{\Phi}_{ij}| \notin Z_t$. But the process has been oblivious to the actual underlying variable $|\boldsymbol{\Phi}_{ij}|$. This implies the following.

Proposition 7. *Let $\mathcal{E}_t$ be the set of all pairs (i,j) such that $\pi_t(i,j) \in \{-1,1\}$. The conditional joint distribution of the variables $(|\boldsymbol{\Phi}_{ij}|)_{(i,j)\in\mathcal{E}_t}$ given $\mathcal{F}_t$ is uniform over $(V \setminus Z_t)^{\mathcal{E}_t}$. That is, for any map $f : \mathcal{E}_t \to V \setminus Z_t$ we have*

$$\mathrm{P}\left[\forall (i,j) \in \mathcal{E}_t : |\boldsymbol{\Phi}_{ij}| = f(i,j) | \mathcal{F}_t\right] = |V \setminus Z_t|^{-|\mathcal{E}_t|}.$$

In each step of the process **PI1**–**PI4** one variable z_t is added to Z_t. There is a chance that this variable occurs in several other all-negative clauses. Hence, the stopping time T should be smaller than the total number of all-negative clauses. To prove this, we need the following lemma.

Lemma 8. *W.h.p. the following is true for all $1 \leq t \leq \min\{T, n\}$: the number of indices $i \in [m]$ such that $\pi_t(i,j) = -1$ for all $1 \leq j \leq k$ is at most $2n\omega \exp(-kt/n)/k$.*

Proof. The proof illustrates the use of Proposition 7. Let $\mathcal{N}_{tij} = 1$ if $\pi_t(i,j) = -1$ and $t \leq T$, and let $\mathcal{N}_{tij} = 0$ otherwise. Let $t \leq n$, $\mu = \lceil \ln^2 n \rceil$, and let $\mathcal{I} \subset [m]$ be a set of size μ. Let $Y_i = 1$ if $t \leq T$ and $\pi_t(i,j) = -1$ for all $j \in [k]$, and let $Y_i = 0$ otherwise. Set $\mathcal{J} = [t] \times \mathcal{I} \times [k]$. If $Y_i = 1$ for all $i \in \mathcal{I}$, then $\mathcal{N}_{0ij} = 1$ for all $(i,j) \in \mathcal{I} \times [k]$ and $\mathcal{N}_{sij} = 1$ for all $(s,i,j) \in \mathcal{J}$. We will prove below that

$$\mathrm{E}\left[\prod_{(i,j)\in\mathcal{I}\times[k]} \mathcal{N}_{0ij} \cdot \prod_{(t,i,j)\in\mathcal{J}} \mathcal{N}_{tij}\right] \leq 2^{-k|\mathcal{I}|}(1-1/n)^{|\mathcal{J}|}, \text{ whence} \tag{1}$$

$$\mathrm{E}\left[\prod_{i\in\mathcal{I}} Y_i\right] \leq \lambda^{\mu}, \text{ where } \lambda = 2^{-k}\exp(-kt/n). \tag{2}$$

Let $Y = \sum_{i\in[m]} Y_i$. Then (2) entails that $\mathrm{E}[Y^\mu] \leq (1+o(1))(\lambda m)^\mu$. Therefore, Markov's inequality yields $\mathrm{P}[Y > 2n\omega\exp(-kt/n) \geq 1.9\lambda m] \leq 1.9^{-\mu}$, and thus the assertion follows from the union bound.

To complete the proof, we need to establish (1). Let

$$\mathcal{N}_0 = \prod_{(i,j)\in\mathcal{I}\times[k]} \mathcal{N}_{0ij},\ \mathcal{J}_s = \{(i,j) : (s,i,j) \in \mathcal{J}\},\ \text{and}\ \mathcal{N}_s = \prod_{(i,j)\in\mathcal{J}_s} \mathcal{N}_{sij}.$$

Since the signs of the literals $\boldsymbol{\Phi}_{ij}$ are mutually independent, we have $\mathrm{E}[\mathcal{N}_0] = 2^{-k|\mathcal{I}|}$. Furthermore, we will prove below that $\mathrm{E}[\mathcal{N}_s|\mathcal{F}_{s-1}] \leq (1-1/n)^{|\mathcal{J}_s|}$. Since $\mathcal{N}_s$ is $\mathcal{F}_s$-measurable for any s, we obtain

$$\mathrm{E}\left[\prod_{s=0}^{t}\mathcal{N}_s\right] = \mathrm{E}\left[\mathrm{E}\left[\mathcal{N}_t\middle|\mathcal{F}_{t-1}\right]\cdot\prod_{s=0}^{t-1}\mathcal{N}_s\right] \leq (1-1/n)^{|\mathcal{J}_t|}\cdot\mathrm{E}\left[\prod_{s=0}^{t-1}\mathcal{N}_s\right].$$

Proceeding inductively, we obtain (1).

Finally, we bound $\mathrm{E}[\mathcal{N}_s|\mathcal{F}_{s-1}]$ for $s \geq 1$. If $T < s$ or $\pi_{s-1}(i,j) \neq -1$ for some $(i,j) \in \mathcal{J}_s$, then $\mathcal{N}_s = \mathcal{N}_{sij} = 0$. Hence, suppose that $T \geq s$ and $\pi_{s-1}(i,j) = -1$ for all $(i,j) \in \mathcal{J}_s$. Then at time s **PI2** selects some variable $z_s \in V \setminus Z_{s-1}$, and $\mathcal{N}_{sij} = 1$ only if $|\boldsymbol{\Phi}_{ij}| \neq z_s$. As $\pi_{t-1}(i,j) = -1$ for all $(i,j) \in \mathcal{J}_s$, given $\mathcal{F}_{s-1}$ the variables $(|\boldsymbol{\Phi}_{ij}|)_{(i,j)\in\mathcal{J}_s}$ are independently uniformly distributed over $V\setminus Z_{s-1}$ by Proposition 7. Therefore, for each $(i,j) \in \mathcal{J}_s$ we have $|\boldsymbol{\Phi}_{ij}| = z_s$ with probability at least $1/n$. Hence, $\mathrm{E}[\mathcal{N}_s|\mathcal{F}_{s-1}] \leq (1-1/n)^{|\mathcal{J}_s|}$. □

Corollary 9. *W.h.p. we have $T < 4nk^{-1}\ln\omega$.*

Proof. Let $t_0 = 2nk^{-1}\ln\omega$ and let I_t be the number of indices i such that $\pi_t(i,j) = -1$ for all $1 \leq j \leq k$. By **PI2** $I_t \leq I_{t-1} - 1$ for all $t \leq T$. Consequently, if $T \geq 2t_0$, then $0 \leq I_T \leq I_{t_0} - t_0$, and thus $I_{t_0} \geq t_0$. But Lemma 8 entails that $I_{t_0} < t_0$ w.h.p. □

Let $\theta = \lfloor 4nk^{-1}\ln\omega\rfloor$. The next goal is to estimate the number of Z_t-unique clauses, i.e., the size of the set U_t. Using a similar (if slightly more involved) argument as in the proof of Lemma 8, we can infer the following.

Lemma 10. *W.h.p.* $\max_{0\leq t\leq T}|U_t| \leq (1+\varepsilon/3)\omega n$.

Let us think of the variables $x \in V\setminus Z_t$ as bins and of the clauses $\boldsymbol{\Phi}_i$ with $i \in U_t$ as balls. If we place each ball i into the (unique) bin x such that x occurs positively in $\boldsymbol{\Phi}_i$, then by Lemma 10 and Corollary 9 for $t \leq T$ the average number of balls in a bin is $\leq (1+\varepsilon/3)\omega n/|V\setminus Z_t| \leq (1-0.6\varepsilon)\ln k$ w.h.p. Hence, if the balls were thrown uniformly at random into the bins, we would expect

$$|V\setminus Z_t|\exp(-|U_t|/|V\setminus Z_t|) \geq (n-t)k^{0.6\varepsilon-1} \geq nk^{\varepsilon/2-1}$$

bins to be empty (i.e., $U_t(x) = 0$). The next corollary shows that this is accurate.

Corollary 11. *Let $\mathcal{Q}_t = |\{x \in V\setminus Z_t : U_t(x) = 0\}|$. W.h.p. we have*

$$\min_{t\leq T}\mathcal{Q}_t \geq nk^{\varepsilon/2-1}.$$

Now that we know that w.h.p. there are 'a lot' of variables $x \in V \setminus Z_{t-1}$ such that $U_t(x) = 0$, we expect that it is quite likely for clause $\boldsymbol{\Phi}_{\phi_t}$ to contain one. More precisely, we have the following.

Corollary 12. *Let $\mathcal{B}_t = 1$ if $\min_{j<k_1} U_{t-1}(|\boldsymbol{\Phi}_{\phi_t j}|) > 0$, $\mathcal{Q}_{t-1} \geq nk^{\varepsilon/2-1}$, $|U_t| \leq (1+\varepsilon/3)\omega n$, and $T \geq t$. Let $\mathcal{B}_t = 0$ otherwise. Then $\mathrm{E}[\mathcal{B}_t|\mathcal{F}_{t-1}] \leq \exp(-k^{\varepsilon/6})$ for all $1 \leq t \leq \theta$.*

Proof of Proposition 3. The definition of the process **PI1**–**PI4** mirrors the execution of the algorithm, i.e., the set Z obtained after Steps 1a–1d of `Fix` equals the set Z_T. Therefore, the first assertion is a consequence of Corollary 9 and the fact that $|Z_t| = t$ for all $t \leq T$. Furthermore, the second assertion follows directly from Lemma 10.

To prove the third claim, we need to bound the number of clauses that are unsatisfied under σ_{Z_T}. It is not difficult to see that the construction **PI1**–**PI4** ensures that for any $i \in [m]$ such that $\boldsymbol{\Phi}_i$ is unsatisfied under σ_{Z_T} one of the following is true.

a. There is $t \leq T$ such that $i \in U_{t-1}$ and z_t occurs positively in $\boldsymbol{\Phi}_i$.
b. There are $1 \leq j_1 < j_2 \leq k$ such that $\boldsymbol{\Phi}_{ij_1} = \boldsymbol{\Phi}_{ij_2}$.

Let $\mathcal{X}$ be the number of indices $i \in [m]$ such that a. occurs. We will show that

$$\mathcal{X} \leq n\exp(-k^{\varepsilon/7}) \qquad \text{w.h.p.} \tag{3}$$

Since the number of 'degenerate' $i \in [m]$ for which b. occurs is $O(\ln n)$ w.h.p. (by a simple first moment argument), (3) implies the third assertion.

To establish (3), let $\mathcal{B}_t$ be as in Corollary 12 and set $\mathcal{D}_t = \mathcal{B}_t \cdot U_{t-1}(z_t)$. Invoking Corollary 11 and Lemma 10, it is easy to show that $\mathcal{X} \leq \sum_{1\leq t\leq\theta} \mathcal{D}_t$ w.h.p. Further, the random variable $\mathcal{D}_t$ is $\mathcal{F}_t$-measurable and $\mathcal{D}_t = 0$ for all $t > \theta$. Let

$$\bar{\mathcal{D}}_t = \mathrm{E}[\mathcal{D}_t|\mathcal{F}_{t-1}] \text{ and } \mathcal{M}_t = \sum_{s=1}^{t} \mathcal{D}_s - \bar{\mathcal{D}}_s.$$

Then $\mathcal{M}_1, \ldots, \mathcal{M}_\theta$ is a martingale with $\mathrm{E}[M_\theta] = 0$. Azuma's inequality entails that $\mathcal{M}_\theta = o(n)$ w.h.p. Hence, w.h.p. $\sum_{1\leq t\leq\theta} \mathcal{D}_t = o(n) + \sum_{1\leq t\leq\theta} \bar{\mathcal{D}}_t$.

We claim that $\bar{\mathcal{D}}_t \leq 2\omega\exp(-k^{\varepsilon/6})$ for all $1 \leq t \leq \theta$. For by Corollary 12 we have $\mathrm{E}[\mathcal{B}_t|\mathcal{F}_{t-1}] \leq \exp(-k^{\varepsilon/6})$. Moreover, given $\mathcal{F}_{t-1}$ we have $\pi_{t-1}(\phi_t, k_1) = -1$, whence z_t is uniformly distributed over $V \setminus Z_{t-1}$ (by Proposition 7). Since $\mathcal{B}_t = 1$ implies $|U_{t-1}| \leq (1+\varepsilon/3)\omega n$, the conditional expectation of $U_{t-1}(z_t)$ is

$$\leq |U_{t-1}|/|V \setminus Z_{t-1}| \leq (1+\varepsilon/3)\omega n/(n-t) \leq 2\omega.$$

Combining these estimates, we obtain that w.h.p.

$$\sum_{1\leq t\leq\theta} \mathcal{D}_t \leq 2\omega\exp(-k^{\varepsilon/2}/3)\theta + o(n) \leq n\exp(-k^{\varepsilon/7}).$$

Thus, (3) follows from the fact that $\mathcal{X} \leq \sum_{1\leq t\leq\theta} \mathcal{D}_t$ w.h.p. □

References

1. Achlioptas, D., Beame, P., Molloy, M.: Exponential bounds for DPLL below the satisfiability threshold. In: Proc. 15th SODA, pp. 139–140 (2004)
2. Achlioptas, D., Coja-Oghlan, A.: Algorithmic barriers from phase transitions. In: Proc. 49th FOCS, pp. 793–802 (2008)
3. Achlioptas, D., Moore, C.: Random k-SAT: two moments suffice to cross a sharp threshold. SIAM Journal on Computing 36, 740–762 (2006)
4. Achlioptas, D., Peres, Y.: The threshold for random k-SAT is $2^k \ln 2 - O(k)$. Journal of the AMS 17, 947–973 (2004)
5. Ardelius, J., Zdeborova, L.: Exhaustive enumeration unveils clustering and freezing in random 3-SAT. Phys. Rev. E 78, 040101(R) (2008)
6. Braunstein, A., Mézard, M., Zecchina, R.: Survey propagation: an algorithm for satisfiability. Random Structures and Algorithms 27, 201–226 (2005)
7. Chao, M.-T., Franco, J.: Probabilistic analysis of a generalization of the unit-clause literal selection heuristic for the k-satisfiability problem. Inform. Sci. 51, 289–314 (1990)
8. Chvátal, V., Reed, B.: Mick gets some (the odds are on his side). In: Proc. 33th FOCS, pp. 620–627 (1992)
9. Coja-Oghlan, A., Feige, U., Frieze, A., Krivelevich, M., Vilenchik, D.: On smoothed k-CNF formulas and the Walksat algorithm. In: Proc. 20th SODA, pp. 451–460 (2009)
10. Flaxman, A.: Algorithms for random 3-SAT. Encyclopedia of Algorithms (2008)
11. Frieze, A., Suen, S.: Analysis of two simple heuristics on a random instance of k-SAT. Journal of Algorithms 20, 312–355 (1996)
12. Hajiaghayi, M., Sorkin, G.: The satisfiability threshold of random 3-SAT is at least 3.52. IBM Research Report RC22942 (2003)
13. Kaporis, A., Kirousis, L., Lalas, E.: The probabilistic analysis of a greedy satisfiability algorithm. Random Structures and Algorithms 28, 444–480 (2006)
14. Kim, J.H.: Poisson cloning model for random graph (preprint, 2006)
15. Montanari, A., Ricci-Tersenghi, F., Semerjian, G.: Solving constraint satisfaction problems through Belief Propagation-guided decimation. In: Proc. 45th Allerton (2007)
16. Semerjian, G., Monasson, R.: A study of pure random walk on random satisfiability problems with "Physical" methods. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 120–134. Springer, Heidelberg (2004)