Mick Gets Some (the Odds Are on His Side)

V. Chvátal

Computer Science Department
Rutgers University
New Brunswick, NJ 08903, USA

B.Reed

Forschungsinstitut f. Diskrete Mathematik
Universität Bonn
Bonn, D-5300 Germany

Abstract

Consider a randomly generated boolean formula F (in the conjunctive normal form) with m clauses of size k over n variables; k is fixed at any value greater than 1, but n tends to ininity and $m = (1 + o(1))cn$ for some c depending only on k. It is easy to see that F is unsatisfiable with probability $1 - o(1)$ whenever $c > (\ln 2)2^k$; we complement this observation by proving that F is satisfiable with probability $1 - o(1)$ whenever $c < (0.25)2^k/k$; in fact, we present a linear-time algorithm that satisfies F with probability $1 - o(1)$. (This result is a continuation of work by Chao and Franco.) In addition, we establish a threshold for 2-SAT: if $k = 2$ then F is satisfiable with probability $1 - o(1)$ whenever $c < 1$ and unsatisfiable with probability $1 - o(1)$ whenever $c > 1$.

0 Introduction

A *truth assignment* is a mapping f that assigns 0 or 1 to each variable in its domain. The *complement* $\overline{x}$ of each such variable x is defined by $f(\overline{x}) = 1 - f(x)$ for all truth assignments f; both x and $\overline{x}$ are called *literals*. A *clause* is a set of (distinct) literals and a *formula* is a family of (not necessarily distinct) clauses. A truth assignment f *satisfies* a clause C if and only if $f(w) = 1$ for at least one literal w in C; the assignment *satisfies* a formula F if and only if it satisfies every clause in F. (Thus a formula is precisely that which is often called a Boolean expression in the conjunctive normal form.) A formula is called *satisfiable* if it is satisfied by at least one truth assignment; otherwise it is called *unsatisfiable*.

Let us call a clause C *ordinary* if there is no variable x such that both x and $\overline{x}$ belong to C; let n, m, k be positive integers and let X be a set of size n. The *random formula with m clauses of size k over n variables* consists of independent random variables $C_1, C_2, \ldots, C_m$ such that each C_i is distributed uniformly over the $\binom{n}{k}2^k$ ordinary clauses of size k with variables coming from X.

Throughout, we reserve n for the number of variables, m for the number of clauses, k for the clause size, and c for any positive constant depending only on k; the asymptotic symbol $o(1)$ will always mean a quantity tending to zero as n tends to infinity.

The expected number of truth assignments that satisfy a random formula is $2^n(1 - 2^{-k})^m$; hence a random formula with $(1 + o(1))cn$ clauses of size k over n variables such that

$$c > (\ln 2) \cdot 2^k \tag{1}$$

is unsatisfiable with probability $1 - o(1)$. This observation has been made by Franco and Paull [6], by Simon et al. [8], by Chvátal and Szemerédi [3], and possibly by others; our purpose is to complement it by the following result.

THEOREM 1. If $k \geq 2$ then a random formula with $(1 + o(1))cn$ clauses of size k over n variables such that

$$c < \frac{1}{4} \cdot \frac{2^k}{k} \tag{2}$$

is satisfiable with probability $1 - o(1)$. □

(The assumption that $k \geq 2$ is crucial: it is an easy exercise to show that, with probability $1 - o(1)$, a random formula with $m(n)$ clauses of size one over n variables is unsatisfiable whenever $m(n)n^{-1/2} \to \infty$ and satisfiable whenever $m(n)n^{-1/2} \to 0$.)

In fact, we prove something better than Theorem 1: rather than merely establishing that the random formula is satisfiable with probability $1-o(1)$, we show how to find quickly an actual truth assignment that satisfies it with probability $1 - o(1)$. Our algorithm, which we call SC, will be described in the next section;

0-8186-2900-2/92 $3.00 © 1992 IEEE

it runs in time $O(n + km)$, linear in the size of the input, and its performance guarantee goes as follows.

THEOREM 2. Let k be an integer such that $k \geq 3$ and let c be a nonnegative number such that

$$c < \frac{1}{8}\left(\frac{k-1}{k-3}\right)^{k-3}\frac{k-1}{k-2}\cdot\frac{2^k}{k}. \qquad (3)$$

(We adopt the convention that $0^0 = 1$, and so the right-hand side of (3) equals 2/3 when $k = 3$.) Then algorithm SC, given a random formula with $(1 + o(1))cn$ clauses of size k over n variables, finds a truth assignment that satisfies this formula with probability $1 - o(1)$. □

Since

$$\left(\frac{k-3}{k-1}\right)^{k-3}\frac{k-2}{k-1} < e^{-2}\frac{(k-1)(k-2)}{(k-3)^2} \leq 3e^{-2} < \frac{1}{2}$$

whenever $k \geq 5$, Theorem 2 implies Theorem 1 except when $k = 2$. We shall deal with the exceptional case separately.

THEOREM 3. A random formula with $(1+o(1))cn$ clauses of size two over n variables such that

$$c < 1$$

is satifiable with probability $1 - o(1)$. □

We do not have to worry about finding quickly an actual truth assignment that satisfies the random formula in Theorem 3 with probability $1-o(1)$: for every satisfiable formula F with $k = 2$, a truth assignment satifying F can be found in time polynomial in the size of the input. This has been pointed out by Cook [4]; later on, two linear-time algorithms were designed by Even, Itai and Shamir [5] and by Aspvall, Plass, and Tarjan [1].

Incidentally, now we may replace the proportionality constant 1/4 in (2) by 3/8 : the trick is to appeal to Theorem 2 in case $k \geq 4$ and to Theorem 3 (with one literal removed at random from each clause when $k = 3$) in case $k \leq 3$. However, we are not concerned with maximizing the proportionality constant: as we see it, the more interesting question is whether the gap between the two orders of magnitude, 2^k in (1) for unsatisfiability and $2^k/k$ in (2) for satisfiability, can be narrowed.

Perhaps this gap can be closed completely in the sense that, for every integer k greater than 1, there is a constant c_k such that a random formula with $(1 + o(1))cn$ clauses of size k over n variables is satisfiable with probability $1 - o(1)$ whenever $c < c_k$ and unsatisfiable with probability $1-o(1)$ whenever $c > c_k$. However, we can prove this only when $k = 2$.

THEOREM 4. A random formula with $(1+o(1))cn$ clauses of size two over n variables such that

$$c > 1$$

is unsatisfiable with probability $1 - o(1)$. □

After this paper was written, Valerie King informed us that our Theorems 3 and 4 were obtained independently by Goerdt [7] and by de la Vega [11].

Our work on Theorem 2 was inspired by earlier work of Chao and Franco [2]. They analyzed two algorithms, which they call UC and GUC, that attempt to satisfy a prescribed formula. (Our algorithm SC is a close relative of UC and GUC.) Two of their results pertaining to our Theorem 2 go as follows.

[2, Theorem 4]: Let k be an integer such that $k \geq 3$ and let c be a positive number such that

$$c < \frac{1}{2}\left(\frac{k-1}{k-2}\right)^{k-2}\cdot\frac{2^k}{k}.$$

Then algorithm UC, given a random formula with $(1+ o(1))cn$ clauses of size k over n variables, finds a truth assignment that satisfies this formula with probability at least $\varepsilon + o(1)$ for some positive ε. □

[2, Theorem 6]: Let k be an integer such that $4 \leq k \leq 40$ and let c be a positive number such that

$$c < 0.46125\left(\frac{k-1}{k-2}\right)^{k-2}\cdot\frac{2^k}{k+1} - 1.$$

Then algorithm GUC, given a random formula with $(1 + o(1))cn$ clauses of size k over n variables, finds a truth assignment that satisfies this formula with probability $1 - o(1)$. □

(The notation that Chao and Franco use in [2] differs from ours: their r is our n and their n is our m. Our choice was motivated by the convention of graph and hypergraph theory, where n usually stands for the number of vertices and m stands for the number of edges.)

1 Algorithms

The algorithms UC, GUC, and SC follow the same scheme, where the truth assignment is constructed in n iterations with precisely one variable per iteration set at 1 or 0. Setting the value of a literal w in a formula F at 1 amounts to removing from F all the clauses that contain w and deleting $\overline{w}$ from all the clauses that contain $\overline{w}$. We shall let $F[w]$ denote the resulting formula; with this notation, the common scheme of UC, GUC, and SC can be described as follows:

```
$F_0 = F$;
for $t = 1$ to $n$ do
    choose a literal $w_t$ whose value has not yet
    been set;
    $w_t = 1; F_t = F_{t-1}[w_t]$
end
```

The three algorithms differ only in their choice of w_t.

Trivially, the truth assignment constructed by this scheme satisfies F if an only if F_n includes no null clause (a clause of size zero); trivially, a null clause that appears in some F_t will persist in $F_{t+1}, \ldots, F_n$. These observations suggest a policy for choosing w_t: if F_{t-1} includes a clause C of size one, say $C = \{v\}$, then we must set $v = 1$ at some time in the future in order to satisfy F, and so we may just as well choose $w_t = v$ now. That is what algorithm UC (short for unit clause) does: w_t is a literal that occurs in a clause of size one in F_{t-1} if such a clause is present and w_t is chosen arbitrarily otherwise. Thus algorithm UC evades the danger of a null clause at the last moment; algorithm GUC (short for generalized UC) evades it already from far away: there, w_t is a literal that occurs in a clause of the smallest size in F_{t-1}. Our algorithm SC (short for small clause) comes between the two: we consider a clause dangerous if and only if its size is at most two and we consider clauses of size one more dangerous than clauses of size two. Thus we let w_t be a literal that occurs in a clause of size one in F_{t-1}; if no such clause is present then we let w_t be a literal that occurs in a clause of size two in F_{t-1}; if no clause of size one or two is present in F_{t-1} then we choose w_t arbitrarily.

Actually, these informal decriptions of UC, GUC, and SC are not quite correct: ties in the choice of w_t have to be broken with care in order to justify the probabilistic performance guarantees. Chao and Franco use randomization in their choice of w_t in both UC and GUC; so do we in SC, but our way is slightly different from theirs.

Let us assume that the input is represented by a matrix with m rows and k columns, whose i-th row lists the k literals that appear in the i-th clause of F. The output will be represented by an array of length n that lists the values of the n variables; to initialize this array, we fill each of its n cells with a symbol signifying that the value of the variable has not yet been set.

To keep the running time within $O(n + km)$, we first construct $2n$ linked lists that enumerate, for each of the $2n$ literals w, all the clauses that contain w; each of these clauses is represented simply by a pointer to the corresponding row of the input matrix. These linked lists will be called *incidence lists*. With each $i = 1, 2, \ldots, m$, we associate an integer $size[i]$ which monitors the current size of the i-th input clause (and therefore is initialized as k) and a bit $sat[i]$ which tells us whether the i-th input clause has been satisfied (and therefore is initialized as *false*). In addition, we make use of two stacks: STACK1 lists all the clauses whose current size is 1 (and possibly also some clauses of size 0) and STACK2 lists all the clauses whose current size is 2 (and possibly also some clauses of size at most 1). Again, each clause on these stacks is represented by a pointer to the corresponding row of the input matrix.

To choose w_t, we first attempt to find a clause of size 1 in F_{t-1} by popping STACK1 until an i with $size[i] = 1$ is found or the stack becomes empty; if there is no such clause, then we attempt to find a clause of size 2 in F_{t-1} by popping STACK2 until an i with $size[i] = 2$ is found or the stack becomes empty. If a clause of size 1 or 2 is found, then we let w_t be a randomly chosen entry (from the uniform distribution) in the corresponding row of the input matrix whose value has not yet been set; else we let w_t be the first variable in the output array whose value has not yet been set. (In the latter case, w_t can be found by moving a pointer through the output array, starting from the position where it has been left previously.)

To replace F_{t-1} by F_t, we first scan the incidence list of w and set $sat[i] = true$ for each i on this list. Then we scan the incidence list of $\overline{w}$ and, for each i on this list such that $sat[i] = false$, we decrement $size[i]$ by 1; if $size[i]$ drops to 2, then we push i on STACK2; if $size[i]$ drops to 1 then we push i on STACK1. (If $size[i]$ drops to 0 then we might just as well return a failure message and stop. However, the analysis is smoother if we carry on even in these circumstances.)

For the proof of Theorem 2, it is crucial to note that SC maintains randomness in the following sense: if C is any clause of size s in F_{t-1} other than the small clause from which w_t is selected then $w_t \epsilon C$ with probability $s/2(n-t+1)$ and $\overline{w}_t \epsilon C$ with probability $s/2(n-t+1)$ independently of what happens between w_t and all the remaining clauses. (To see this, it may help to imagine all the entries of the input matrix concealed in the beginning and all the entries equal to w_j or $\overline{w}_j$ disclosed as soon as w_j is selected.) This invariant would be also maintained if we first preprocessed the input by permuting the k entries in each row at random and independently of all the other rows to ensure that the rows are independent identically distributed random variables with the distribution uniform over its $n(n-1)\dots(n-k+1)2^k$ possible values; then we could choose each w_t from a small clause simply as the first entry in the corresponding row of the input matrix whose value has not yet been set. The resulting implementation may be easier to see through as it resorts to randomization only in the preprocessing step and proceeds in a deterministic fashion afterwards; the implementation we chose is marginally faster.

The analysis of SC in our proof of Theorem 2 applies equally well to the implementation of GUC where k stacks of clauses are used to select w_t (and whose running time is also $O(n+km)$). Stating Theorem 2 as we do highlights the features of GUC that suffice to guarantee a high rate of success on random inputs with

$$(1+o(1))\gamma\cdot\frac{2^k}{k}n$$

clauses of size k over n variables for some positive γ independent of k. In this context, it may be interesting to note that UC has no chance of success at a comparable rate: for every choice of a positive integer k and a positive number c there is a positive ε such that UC, given a random input with $(1+o(1))cn$ claues of size k over n variables, fails with probability at least $\varepsilon+o(1)$. (The expected number of iterations in which precisely two new clauses of size one appear is at least δn for some positive δ and so their actual number a is at least $\delta n/2$ with probability greater than $\delta/2$. The probability that the two new clauses of size one clash in none of these a iterations is at most $e^{-\delta/4}$.)

2 Analysis

PROOF OF THEOREM 2. Let us begin by explaining where the right-hand side of (3) comes from: keeping c below this bound amounts to keeping the expected number of input clauses that shrink to size two after precisely t iterations below 1 for all $t=1,2,\dots,n$. More precisely, let p_t denote the probability that a fixed input clause shrinks to size two after precisely t iterations and let ε be any positive number such that

$$\frac{1+\varepsilon}{1-2\varepsilon}c<\frac{1}{8}\left(\frac{k-1}{k-3}\right)^{k-3}\frac{k-1}{k-2}\cdot\frac{2^k}{k}.$$

Since

$$p_t=\frac{\binom{t-1}{k-3}\binom{n-t}{2}}{\binom{n}{k}}\cdot\frac{1}{2^{k-2}}=\frac{\binom{t-1}{k-3}\binom{n-t}{2}}{\binom{n-1}{k-1}}\cdot\frac{k}{2^{k-2}n},$$

p_t is maximized when

$$\frac{k-3}{k-1}n\le t\le\frac{k-3}{k-1}n+1,$$

and so

$$p_t\le(1+\varepsilon)\binom{k-1}{2}\left(\frac{k-3}{k-1}\right)^{k-3}\left(\frac{2}{k-1}\right)^2\cdot\frac{k}{2^{k-2}n}$$
$$<\frac{1-\varepsilon}{m}$$

for all sufficiently large n.

The actual number a_t of input clauses that shrink to size two after precisely t iterations is likely to differ from its expected value by quite large amounts for quite a few values of t. Nevertheless, the average of the fluctuating values of a_t over every sufficiently large block of values of t is likely to stay below 1: we claim that, with probability $1-o(1)$,

$$\sum_{t=r}^{s}a_t\le s-r\text{ whenever }s-r\ge 5\varepsilon^{-2}\ln n.$$

To justify this claim, we rely on the well-known inequality

$$\sum_{i\ge x}\binom{m}{i}p^i(1-p)^{m-i}\le\left(\frac{pm}{x}\right)^x\left(\frac{m-pm}{m-x}\right)^{m-x}\tag{4}$$

which implies that

$$\sum_{i\ge x}\binom{m}{i}p^i(1-p)^{m-i}\le\left(\frac{pm}{x}\right)^x e^{x-pm}\le\exp\left(-\frac{\varepsilon^2x}{2}\right)$$

whenever $pm\le(1-\varepsilon)x$. In particular, if r and s are fixed so that $1\le r\le s\le n$ then the probability of

$$\sum_{t=r}^{s}a_t\ge s-r+1$$

is at most $\exp(-\varepsilon^2(s-r+1)/2)$, which is $o(n^{-2})$ whenever $s-r \geq 5\varepsilon^{-2}\ln n$; since there are $\binom{n+1}{2}$ choices of r and s, the desired conclusion follows.

In addition, the individual values of a_t are likely to be 0 or 1 for all t close enough to n: we claim that, with probability $1-o(1)$,

$$a_t \leq 1 \text{ whenever } n-n^{3/4} < t \leq n.$$

To justify this claim, note first that

$$p_t m = O\left(\left(\frac{n-t}{n}\right)^2\right).$$

Again, we shall rely on (4), which implies that

$$\sum_{i\geq 2}\binom{m}{i}p^i(1-p)^{m-i} \leq \left(\frac{epm}{2}\right)^2.$$

In particular, if t is fixed then the probability of $a_t \geq 2$ is $O(((n-t)/n)^4)$; since there are at most $1+n^{3/4}$ choices of t with $n-n^{3/4} < t \leq n$, the desired conclusion follows.

Now observe that the number of clauses of size one or two in F_t is at most b_t with $b_0, b_1, \ldots, b_n$ defined recursively by

$b_0 = 0$ and $b_s = \max\{b_{s-1}-1, 0\}+a_s$ whenever $s \geq 1$.

An easy induction on s shows that

$$b_s \leq 1 + \max_r \sum_{t=r}^{s}(a_t - 1)$$

(we may set $r = s$ if $b_{s-1} = 0$ and let r be the maximizer for b_{s-1} otherwise). Hence, with probability $1-o(1)$,

$$b_t \leq 5\varepsilon^{-2}\ln n \text{ for all } t \tag{5}$$

and

$$b_t \leq 1 \text{ whenever } n-n^{2/3} \leq t \leq n. \tag{6}$$

We shall complete the proof by showing that, with probability $1-o(1)$, the clauses of size at most two in F_t remain distributed in a certain special way throughout the run of the algorithm. Let us say that F_t is *well-behaved* if

(i) the multigraph whose vertices are the variables in F_t and whose edges are the clauses of size two in F_t contains no cycles (including cycles of length two),

(ii) for each component Q of this multigraph (including the single-vertex components) there is at most one clause C of size one in F_t such that the variable involved in C is a vertex of Q,

(iii) there are no clauses of size zero in F_t.

We propose to prove that F_t remains well-behaved throughout the run of the algorithm with probability $1-o(1)$.

For this purpose, let G_t denote the formula consisting of all the clauses of size at most two in F_t, let V_t denote the set of variables involved in the clauses of $G_{t-1}[w_t]$, and let N_t denote the set of clauses of size two that appear in $G_t - G_{t-1}$. Now consider the following events.

C_t: the multigraph formed by N_t contains a cycle.

P_t: the multigraph formed by N_t contains a path with both endpoints in V_t.

If G_t is well-behaved then $G_{t-1}[w_t]$ is well-behaved; hence F_t ceases to be well-behaved if and only if at least one of events C_t, P_t takes place. It follows that F_j is not well-behaved with probability at most

$$\sum_{t=1}^{j}\text{Prob}(C_t) + \sum_{t=1}^{j}\text{Prob}(P_t).$$

To bound this quantity from above, consider events

B_t: $|V_t| \leq 2b$ and $|N_t| \leq b$ with $b = 5\varepsilon^{-2}\ln n$.

From (5), we have

$$\text{Prob}\,(B_1B_2\ldots B_n) = 1-o(1)\,;$$

in addition,

$$\text{Prob}\,(C_t|B_t) \leq \sum_{i=2}^{n-t}(n-t)^i b^i \binom{n-t}{2}^{-i} < \frac{8b^2}{(n-t-1)^2}$$

and

$$\text{Prob}\,(P_t|B_t) \leq \sum_{i=1}^{n-t-1}(2b)^2(n-t)^{i-1}b^i\binom{n-t}{2}^{-i} < \frac{16b^3}{(n-t)(n-t-1)}$$

whenever $n-t-1 \geq 4b$. Setting $j = n - \lfloor n^{2/3}\rfloor$ we conclude that F_j is well-behaved with probability $1-o(1)$. If (6) holds then F_j includes at most one clause of size at most two and at most one new clause of size at most two appears in each subsequent iteration. Hence F_t remains well-behaved throughout $t = j+1, j+2, \ldots, n$ with probability $1-o(1)$. □

PROOF OF THEOREM 3. By a *bicycle*, we shall mean a formula with (at least two) distinct variables $x_1, x_2, \ldots, x_s$ and clauses $C_0, C_1, \ldots, C_s$ that have the following structure: there are literals $w_1, w_2 \ldots, w_s$ such that each w_r is x_r or $\overline{x}_r$, each C_r with $0 < r < s$ is $\{\overline{w}_r, w_{r+1}\}$, and $C_0 = \{u, w_1\}$, $C_s = \{\overline{w}_s, v\}$ with literals u, v chosen from $x_1, x_2, \ldots, x_s, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_s$. We propose to show that

(i) a random formula with $(1+o(1))cn$ clauses of size two over n variables such that $c < 1$ contains a bicycle with probability $o(1)$,

(ii) every unsatisfiable formula with all clauses of size two contains a bicycle.

To prove (i), let p denote the probability that a random formula with m clauses of size two over n variables contains a bicycle. Since

$$\begin{aligned} p &\leq \sum_{s=2}^{n} n^s 2^s (2s)^2 m^{s+1} \left(\frac{1}{4\binom{n}{2}}\right)^{s+1} \\ &= \frac{2m}{n(n-1)} \sum_{s=2}^{n} s^2 \left(\frac{m}{n-1}\right)^s, \end{aligned}$$

we have $p = O(n^{-1})$ whenever $m = (1 + o(1))cn$ with $c < 1$.

To prove (ii), we appeal to the work of Aspvall, Plass, and Tarjan [1]. They associate a directed graph $G(F)$ with every formula F whose clauses have size two. The vertices of $G(F)$ are the $2n$ literals arising from the n variables of F; each clause$\{u, v\}$ in F gives rise to directed edges $\overline{u} \to v$ and $\overline{v} \to u$ in $G(F)$. Theorem 1 of [1] asserts that F is unsatisfiable if and only if some variable and its complement belong to the same strongly connected component of $G(F)$. In particular, if F is unsatisfiable then $G(F)$ contains a directed walk $w_0 w_1 \ldots w_i$ such that $w_0 = w_i$ and $w_t = \overline{w}_0$ for some t. Choosing a walk that minimizes t, we find that all the literals $w_1, w_2, \ldots, w_t$ are distinct and none of them is the complement of another. Let s be the largest subscript such that all the literals $w_1, w_2, \ldots, w_s$ are distinct and none of them is the complement of another. Then the clauses $C_0, C_1, \ldots, C_s$ with $C_r = \{\overline{w}_r, w_{r+1}\}$ form a bicycle. □

PROOF OF THEOREM 4. First, consider just a set of n variables; let t be an integer depending on n in such a way that

$$t/\log n \to \infty \quad \text{but} \quad t/n^{1/9} \to 0. \tag{7}$$

By a *snake*, we shall mean a sequence of distinct literals $w_1, w_2, \ldots, w_s$ with $s = 2t - 1$ such that no w_i is the complement of another. With each snake A, we shall associate the set F_A of $s + 1$ clauses $\{\overline{w}_r, w_{r+1}\}$, $0 \leq r \leq s$, such that $w_0 = w_{s+1} = \overline{w}_t$. Note that F_A is unsatisfiable: every truth assignment that satisfies F_A must have $w_i \leq w_j$ whenever $i \leq j$, and so it must have $\overline{w}_t = w_0 \leq w_t \leq w_{2t} = \overline{w}_t$.

Now let $p_i(n)$ denote the probability that, for a fixed snake A and a snake B chosen at random from the uniform distribution, sets F_A and F_B share precisely i clauses. A crucial part of our argument is the claim that, for all sufficiently large n,

$$p_i(n) < \frac{1300t^9}{n} \left(\frac{1}{2n}\right)^i \tag{8}$$

whenever $1 \leq i \leq t - 1$ and

$$p_i(n) < 18tn \left(\frac{1}{2n}\right)^i \tag{9}$$

whenever $1 \leq i \leq 2t$.

In justifying this claim, it will be convenient to view F_A as a graph with vertices $x_1, x_2, \ldots, x_s$ (each x_r being the variable such that w_r is x_r or $\overline{x}_r$) and edges $x_r x_{r+1}$, $0 \leq r \leq s$, such that $x_0 = x_{s+1} = x_t$. When F_A and F_B are both viewed as graphs, we let F_{AB} denote their intersection with all isolated vertices removed; for a fixed snake A, we let $N(i,j)$ denote the number of snakes B such that F_{AB} has i edges and j vertices. To obtain an upper bound on $N(i,j)$, we set up a production line that manufactures all the snakes counted by $N(i,j)$: (i) choose j terms of A for membership in F_{AB}, (ii) choose j terms of B for membership in F_{AB}, (iii) assign values to the j terms in $B \cap F_{AB}$, (iv) assign values to the $s - j$ terms in $B - F_{AB}$. As for (i), members of F_{AB} can be selected from A by first deciding whether $x_0 x_1$ belongs to F_{AB} or not and then placing a marker at each x_r with $1 \leq r \leq s$ such that precisely one of $x_{r-1}x_r$ and $x_r x_{r+1}$ belongs to F_{AB}. Observing that there are at least $2(j - i) - 1$ and at most $2(j - i) + 2$ markers, we conclude that there are at most $2\binom{s+3}{2j-2i+2}$ choices in (i), and at most that many choices in (ii). To analyze (iii), let k denote the number of connected components in F_{AB}. Components that are paths may be mixed and matched with their counterparts in up to $k!$ ways and each of these components may flip. In addition, there may be a unique component that is not a path; this component may be mapped onto its counterpart in at most $2t$ ways. We conclude that there are at most $tk!\,2^k$ choices in (iii). Trivially, there are at most $(2n)^{s-j}$ choices in (iv).

If $1 \leq i \leq t-1$ then $k = j - i$; hence

$$N(i,j) \leq 4\binom{2t+2}{2j-2i+2}^2 t(j-i)!\,2^{j-i}(2n)^{s-j}$$
$$\leq 4t(2t+2)^{4(j-i)+4}\,2^{j-i}(2n)^{s-j}$$

and

$$\begin{aligned} p_i(n) &\leq \frac{1}{\binom{n}{s}s!2^s}\sum_{j\geq i+1} N(i,j) \\ &< \frac{5t(2t+2)^4}{(2n)^i}\sum_{j\geq i+1}\left(\frac{(2t+2)^4}{n}\right)^{j-i} \\ &< \frac{1300t^9}{n}\left(\frac{1}{2n}\right)^i, \end{aligned}$$

which is (8). If the upper bound on i is dropped then we can maintain only $k \leq j - i + 2$; now

$$N(i,j) \leq 4\binom{2t+2}{2j-2i+2}^2 t(j-i+2)!\,2^{j-i+2}(2n)^{s-j}$$
$$\leq 16t(2t+2)^{4(j-i)+4}2^{j-i}(2n)^{s-j}$$

and

$$\begin{aligned} p_i(n) &\leq \frac{1}{\binom{n}{s}s!2^s}\sum_{j\geq i-1} N(i,j) \\ &< \frac{17t(2t+2)^4}{(2n)^i}\sum_{j\geq i-1}\left(\frac{(2t+2)^4}{n}\right)^{j-i} \\ &< 18tn\left(\frac{1}{2n}\right)^i, \end{aligned}$$

which is (9).

The rest is straightforward. We propose to show that, with probability $1 - o(1)$, a random formula F with $(1+o(1))cn$ clauses of size two over n variables such that $c > 1$ includes all the clauses of some F_A. The proof is a routine application of the second moment method: writing $X = \sum X_A$ with $X_A = 1$ if each clause of F_A appears precisely once in F and $X_A = 0$ otherwise, we shall prove that

$$E(X^2) \leq (1+o(1))E(X)^2. \qquad (10)$$

The desired conclusion will follow from (10) by substituting $E(X)$ for d in the Chebyshev inequality,

$$\text{Prob}\left(|X - E(X)| \geq d\right) \leq \frac{E(X^2) - E(X)^2}{d^2}.$$

To prove (10), consider arbitrary snakes A and B. We have $E(X_A) = E(X_B) = f(2t)$ with

$$f(x) = \binom{m}{x}x!\left(\frac{1}{4\binom{n}{2}}\right)^x\left(1 - \frac{x}{4\binom{n}{2}}\right)^{m-x};$$

if F_A and F_B share precisely i clauses then $E(X_AX_B) = f(4t-i)$. Since $m = O(n)$, we have

$$f(x) = (1+o(1))\left(\frac{m}{2n(n-1)}\right)^x$$

uniformly in every range $x = O(n^\alpha)$ with $\alpha < 1/2$; hence (7) guarantees that

$$E(X_AX_B) = (1+o(1))E(X_A)E(X_B)\left(\frac{2n(n-1)}{m}\right)^i$$

uniformly in the range $0 \leq i \leq 2t$. Since

$$E(X^2) = \sum_A\sum_B E(X_AX_B)$$

and

$$E(X)^2 = \sum_A\sum_B E(X_A)E(X_B),$$

it follows that

$$E(X^2) = (1+o(1))E(X)^2\sum_{i=0}^{2t}p_i(n)\left(\frac{2n(n-1)}{m}\right)^i.$$

Finally, from (8) and (7) we have

$$\sum_{i=1}^{t-1}p_i(n)\left(\frac{2n(n-1)}{m}\right)^i < \frac{1300t^9}{n}\sum_{i=1}^{t-1}\left(\frac{n-1}{m}\right)^i$$
$$= o(1);$$

from (9) and (7) we have

$$\sum_{i=t}^{2t}p_i(n)\left(\frac{2n(n-1)}{m}\right)^i < 18tn\sum_{i=t}^{2t}\left(\frac{n-1}{m}\right)^i$$
$$= o(1).$$

□

Acknowledgements

Chvátal was supported by the Office of Naval Research grant N0001492J1375; Reed by the Alexander-von Humboldt Stiftung.

References

[1] B. Aspvall, M.F. Plass, and R.E. Tarjan, "A linear-time algorithm for testing the truth of certain quantified Boolean formulas," *Inform. Process. Let.*, Vol. 8, pp. 121-123, 1979.

[2] M.-T. Chao and J. Franco, "Probabilistic analysis of a generalization of the unit-clause literal section heuristic for the k-satisfiability problem," *Inform. Sci.*, Vol. 51, pp. 289-314, 1990.

[3] V.Chvátal and E. Szemerédi, "Many hard examples for resolution," *J. Assoc. Comput. Mach.*, Vol. 35, pp. 759-768, 1988.

[4] S.A. Cook, "The complexity of theorem proving procedures," *Proc. 3rd Ann. ACM Symp. Theory Comput.*, pp. 151-158, 1971.

[5] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comput.*, Vol. 5, pp. 691-703, 1976.

[6] J. Franco and M. Paull, "Probabilistic analysis of the Davis-Putnam procedure for solving the satisfiability problem," *Discrete Appl. Math.*, Vol. 5, pp. 77-87, 1983.

[7] A. Goerdt, "A threshold for unsatisfiability," manuscript, 1991.

[8] J.-C. Simon, J. Carlier, O. Dubois, and O. Moulines, "Étude statistisque de l'existence de solutions de problèmes SAT, application aux systèmes-experts," *C.R. Acad. Sci. Paris. Sér. I Math.*, Vol. 302, pp. 283-286, 1986.

[9] R. Stones, *12* × *5*, PS402, London, 1964.

[10] R. Stones, *Out of our heads*, PS429, London, 1965.

[11] W. F. de la Vega, "On random 2-SAT," manuscript, 1992.