Probabilistic Analysis of Satisfiability Algorithms

John Franco*

Department of Computer Science
University of Cincinnati
Cincinnati, OH 45221-0030

October 2, 2008

1 Introduction

Probabilistic and average-case analysis can give useful insight into the question of what algorithms for testing satisfiability might be effective and why. Under certain circumstances, one or more structural properties shared by each of a family or class of expressions may be exploited to solve such expressions efficiently; or structural properties might force a class of algorithms to require superpolynomial time. Such properties may be identified and then, using probabilistic analysis, one may argue that these properties are so common that the performance of an algorithm or class of algorithms can be predicted for most of a class of expressions. Perhaps most important, sometimes an analysis provides the intutition needed to suggest improved algorithms.

A classic example is the work on resolution for random (n, m, k)-CNF expressions. By a random (n, m, k)-CNF expression we mean a CNF expression of m clauses, each chosen uniformly at random and with replacement from among the set of $2^k\binom{n}{k}$ elementary disjunctions of k literals on a set of n Boolean variables and their complements. Simple variants of the well known Davis-Putnam-Logemann-Loveland algorithm (DPLL) [32] which never reassign a value to a variable once it is set can solve large random (n, m, k)-CNF expressions with probability tending to 1 if they are generated with $m/n < c_k 2^k/k$, where c_k is a constant plus a term of complexity $o(k)$. Since variables are assigned at most one time, those variants run in polynomial time. The probabilistic analyses leading to these results help explain what is actually happening as the algorithms operate, and actually suggest improvements to those algorithms. This has led to remarkable progress in understanding this class of algorithms [63]. The prospect of further progress along these lines is reasonably good.

*Supported in part by DoD contract MDA 904-02-C-1162.

On the other hand, resolution is remarkably poor in trying to prove that no solution exists on random (n, m, k)-CNF expressions which have no solution. In particular, the probability that the length of a shortest resolution proof is bounded by a polynomial in n when $m/n > c'_k 2^k$, c'_k some constant, and $\lim_{m,n\to\infty} m/n = O(1)$ tends to 0 [24, 13]. Again, the analysis illuminates the reason. But, if the number of clauses in a random (n, m, k)-CNF expression is great enough, that expression can, with high probability, efficiently be shown to have no solution. For example, if $m/n = \Theta(n^{k-1})$, the probability that there exist 2^k clauses containing the same variables spanning all different literal complementation patterns tends to 1 and since such a pattern can be identified in polynomial time and certifies there is no solution, random expressions can be proven not to have a solution with probability tending to 1 in this case. Considerable investigation of resolution proof size to find the relationship between m and n that defines the crossover from hard to easy expressions, although brilliant, has not progressed considerably. The best bound on the crossover is not much different from $m/n = \Theta(n^{k-1})$ [12]. This lack of success has motivated consideration of alternatives to resolution. One such alternative is to cast SAT as a Hitting Set problem and sum the number of variables that are *forced* to be set to 1 and the number that are forced to be set to 0; if this sum can be shown to be greater than n in polynomial time, a “short” proof of unsatisfiability follows. This idea has been applied to random (n, m, k)-CNF expressions and yields a hard-easy crossover no worse than $m = n^{k/2+o(1)}$, a considerable improvement over resolution results [52]. This is an example of how probabilistic analysis has driven the search for improved, alternative algorithms.

But it has been difficult for some to take probabilistic results seriously. The main drawbacks of relying on random (n, m, k)-CNF results for practical problems are: 1) some distribution of input expressions must be assumed and a chosen distribution may not represent reality very well; 2) results are usually sensitive to the choice of distribution; 3) the state of analytical tools is such that distributions yielding to analysis are typically symmetric with independent components; 4) few algorithms have yielded to analysis.

Despite these drawbacks, probabilistic results can be a useful supplement to worst-case results, which can be overly pessimistic for NP-complete problems, in understanding algorithmic behavior. One reason for this is that CNF expressions which are presented to a SAT solver are typically translated from another logic form which may even be a binarization of integer variable expressions. Such translations tend to smear or garble original domain-specific structural relationships and make the resulting CNF translation look somewhat like a random expression. We have noticed this happen on some problems related to circuit verification, for example, which are very difficult for advanced SAT solvers such as Chaff.

Hence, in order to better understand the nature of expressions that are hard for current SAT solvers, it seems reasonable to study the relationship between hardness and random expressions. There has been much work on this subject in recent years (see, for example, [1] and [40] for a bibliography), most focusing on random (n, m, k)-CNF expressions. As m and n tend to infinity with limiting ratio $m/n \rightarrow \alpha$, probabilistic analysis and experimental results have provided evidence for the existence of a phase transition at some $\alpha = r_k$. That is, when in the limit $m/n < r_k$, the probability that a random (n, m, k)-CNF expression is satisfiable tends to 1 and when $m/n > r_k$ that probability tends to 0. Results show, for each fixed k, there is a sharp threshold [47], with associated critical ratio $r_k \geq 2^{k-2}/k$ [23] (it is known that $\lim_{k\rightarrow\infty} r_k \rightarrow c\cdot 2^k$, c constant [4, 5], but the result of those papers is not close to the actual threshold when k is around 3). It has also been observed that random expressions become harder for SAT solvers when generated with values of m and n where the ratio m/n is close to r_k and easier when m/n is distant from r_k: the more distant being easier.

These results and observations have suggested a relationship between hardness and threshold. Further investigation has identified long "backbones," or chains of inferences, to be a good candidate for the underlying cause of the sharp thresholds and poor algorithm performance near the thresholds since it appears to be the high density of well-separated "almost solutions" induced by the backbones that lead to thrashing in search algorithms [22]. In [77] and other articles it has been suggested that there is a strong connection between the "order" of threshold sharpness, that is whether the transition is smooth or discontinuous, and hardness.

Recent advances [28, 30] have revealed the importance of minimal monotonic structures to the existence of sharp transitions. Those results have been used, for example, to show how limited most succinctly defined polynomial-time classes of SAT are. Notable examples of such classes are Horn [33, 58], renameable Horn [73], q-Horn [16, 15], extended Horn [18], SLUR [89], balanced [25], and matched [46], to name a few. These classes have been studied partly in the belief that they will yield some distinction between hard and easy problems. For example, in [16] a satisfiability index is presented such that a class with index greater than $1 + \epsilon$, for any positive constant ϵ, is NP-complete but the q-Horn class has satisfiability index 1. Thus, it seems that q-Horn is situated right at the point delineating hard and easy satisfiability problems. This hypothesis has been tested somewhat using m/n as a scale for determining the boundaries, in a probabilistic sense, of q-Horn and other classes: it has been found that a random (n, m, k)-CNF expression is q-Horn with probability tending to 0 if $m/n > 2/k(k-1)$ and that the probability that a random (n, m, k)-CNF expression is q-Horn is bounded away from 0 if $m/n < 1/k(k-1)$ [46]. Similar results have been obtained for other polynomial-time solvable classes. They illuminate the fact that most instances of such classes are satisfiable since their extent on the m/n scale is far below the r_k satisfiability threshold. Since their boundaries, in a probabilistic sense, are so distant from the threshold, all the polynomial-time classes mentioned above may be considered *extremely* easy, especially when compared to the good probabilistic performance shown for polynomial-time incomplete algorithms in the range $m/n < c_k \cdot 2^k/k$ [23].

But the limitations of these classes seem to be related to thresholds. Except for the matched class, the classes above, including q-Horn, are "vulnerable" to cyclic clause structures, any one of which prevents an expression containing such a structure from being a member of the class. These structures have the recently discovered minimality and monotonic properties which are necessary for sharp thresholds and are defined in [28]. So, it seems to find challenging polynomial-time solvable classes it is advisable to look for classes which are not so vulnerable: that is, those for which expressions cannot be excluded by adding certain minimal monotonic structural components. This is a case where probabilistic tools might prove useful in the development of broad succinctly definable polynomial-time SAT classes.

Hopefully, at this point the reader is convinced that probabilistic analysis can be an important tool in understanding and coping with SAT expressions. In what follows we review some notable probabilistic results, the mathematical tools they are based on, and suggest how these results add to our intuition about SAT.

2 CNF Expressions

All results in this chapter apply to Boolean expressions in Conjunctive Normal Form (CNF). Important elements of CNF expressions as well as the definition of CNF expressions are presented here for completeness.

Let v be a Boolean variable (in what follows we use variable to mean Boolean variable). Then v can be assigned a value from $\{0, 1\}$. A *positive literal* is a variable. A *negative literal* is the complement of a variable: that is, it takes value 1 if and only if its corresponding variable takes value 0. We use the symbol $\bar{\ }$ to represent the operation of complementing a literal: that is, if l is a literal, then $\bar{l}$ is its complement. Thus, if v is a positive literal, its complement is denoted by $\bar{v}$ and if $\bar{v}$ is a negative literal its complement is $\bar{\bar{v}}$ which is also v. Whether a literal is complemented or not is referred to as its *polarity*.

A *clause* is a disjunction of literals and evaluates to 1 if and only if one of its literals has value 1. In this chapter we represent a clause as a set of literals. The *width* of a clause is the number of literals it contains. A clause of width 1 is called a *unit clause*.

A *CNF* expression is a set of clauses. An expression evaluates to 1 if and only if all its clauses evaluate to 1. We use C to represent a clause, ϕ to represent a CNF expression, and V to represent the set of variables existing in ϕ as positive or negative literals. If $\exists C \in \phi : l \in C$ and $\forall C \in \phi, \bar{l} \notin C$ then l is called a *pure literal*. An assignment of values to V is called a *truth assignment* or *assignment*. If T is an assignment, the notation $\phi(T)$ is used to mean the value of ϕ under T. If there exists an assignment M such that $\phi(M) = 1$, then M is called a *model* for ϕ and ϕ is said to have a solution. In this chapter we represent a model M

as a set of variables: the value of all variables in M is considered to be 1 and the value of all other variables is considered to be 0.

We use the term k-CNF to mean any of a family of CNF expressions all of whose clauses have width k. It is well known that 2-CNF expressions are solved in polynomial time [38]. Therefore, we will only be interested in generating k-CNF expressions where $k \geq 3$.

Several generators of CNF expressions have been studied but one has withstood the test of time and has received much more attention than the rest. Because of this, we focus attention on that generator in this chapter. This generator (from [45]) has three parameters, n, m, and k. An expression returned by this generator contains m clauses generated uniformly, independently, and with replacement from the set of all elementary disjunctions of k literals on a set of n Boolean variables and their complements. Such an expression is referred to as a random (n, m, k)-CNF expression.

3 Basic Tools of Analysis

In this section we discuss some basic probabilistic tools which are commonly applied to Satisfiability algorithms.

3.1 First moment method

The first moment method is used to provide an upper bound on the probability that a specified property P holds on a class of random structures. Let X be a positive real-valued random variable. Then

$$Pr(X \geq 1) = \int_{t=1}^{\infty} p_X(t)dt \leq \int_{t=1}^{\infty} tp_X(t)dt \leq E\{X\}$$

where we have used p_X to denote the distribution density function of X and $E\{X\}$ to denote the mean of X. The above is known as Markov's inequality. It says the mean of X is an upper bound on the probability that X takes value at least 1. Suppose X takes value at least 1 if and only if a random structure has property P and takes value 0 otherwise. Then the mean is an upper bound on the probability that a random structure has property P. The bound will be useful if the mean is small.

For example, let P be the property that there exists a model for a random (n, m, k)-CNF expression ϕ and suppose we wish to find the probability that P holds in ϕ. Define X to be the *number* of models for ϕ. The probability that P holds is identical to the probability that $X \geq 1$. By Markov's inequality, this is bounded by $E\{X\}$. Index all possible assignments of variables to values (there are 2^n such assignments). Define indicator random variables $I_1, ..., I_{2^n}$ such that

I_i has value 1 if and only if the ith assignment is a model for ϕ and has value 0 otherwise. For all i, the probability that $I_i = 1$ is $(1 - 2^{-k})^m$ since $(1 - 2^{-k})$ is the probability that a clause has value 1 under the ith assignment and clauses are constructed uniformly, independently, and with replacement. Then, since $X = I_1 + ... + I_{2^n}$,

$$E\{X\} = \sum_{i=1}^{2^n} E\{I_i\} = \sum_{i=1}^{2^n} Pr(I_i = 1) = 2^n(1 - 2^{-k})^m.$$

Thus, the probability that P holds is bounded from above by $2^n(1 - 2^{-k})^m$. This bound is quite useful if

$$m/n \quad > \quad -\ln(2)/\ln(1 - 2^{-k}) \tag{1}$$

for then $\lim_{n,m\to\infty}(2(1 - 2^{-k})^{m/n})^n \to 0$, and by the first moment method, a random $(n, m, 3)$-CNF expression has no model with probability tending to 1 (with increasing m and n) if $m/n > 5.19$.

3.2 Second moment method

The second moment method is used to prove that a specified property P holds on a class of random structures with high probability as the size parameter of the class tends to ∞. It is applied here in two major ways: to determine bounds, in probability, on the running time of a proposed algorithm for finding a model; and to determine a bound on the probability that a random expression is a member of a particular class of expressions and therefore exhibits certain properties which may be exploited by an algorithm for finding a model.

For a given random structure (in our case, a random (n, m, k)-CNF expression ϕ), a *witness* w is a substructure (in our case, a witness may be a set of clauses, $w \subseteq \phi$) whose presence implies that the structure has property P. Let W be the set of all possible witnesses for the class. The idea is to prove the probability that a randomly chosen structure *fails to contain any witness* tends to zero with increasing size parameter.

Let w also represent the *event* that $w \subseteq \phi$: which meaning is intended will be clear from context. Usually the set of witnesses W is chosen so its elements are *symmetric*: i.e., for any pair $w, z \in W$, there is an automorphism of the probability space that maps w to z. We shall assume this is the case. Thus $p = Pr(w)$ is independent of w. Let I_w be the indicator random variable that has value 1 if event w occurs and 0 otherwise. Then,

$$E\{I_w\} = p \quad \text{and} \quad \text{var}(I_w) = E\{(I_w - p)^2\} = p(1 - p).$$

Define the random variable $X = \sum_{w\in W} I_w$, and let $\mu = E\{X\} = |W|p$ and $\sigma^2 = \text{var}(X)$. A special case of the Chebyshev inequality states that

$$Pr(X = 0) \quad \leq \quad \frac{\sigma^2}{\mu^2}. \tag{2}$$

Thus it suffices to show that this ratio tends to zero as the size parameter increases (in our case, as $n \to \infty$).

If the events w are independent, then $\sigma^2 = |W|p(1-p) = O(\mu)$, and it becomes sufficient to show that $\mu \to \infty$ as $n \to \infty$. But the events w are usually not independent. In that case the crucial aspect of the second moment method is to show that, although the events w *are not* independent, the dependencies are weak enough that $\sigma^2 = o(\mu^2)$.

To analyze σ^2 in the case of random (n, m, k)-CNF expressions, we introduce the following notation:

$A(w)$ is the set of witnesses having some clause in common with w, other than w itself;

We can now state a basic lemma of the second moment method.

Lemma 1 (Alon and Spencer [10, Ch. 4.3, Cor. 3.5]) *If:*

1. *the elements of W are symmetric,*
2. *$\mu \to \infty$ as $n \to \infty$,*
3. *$\sum_{z \in A(w)} Pr(z|w) = o(\mu)$ for an arbitrary $w \in W$,*

then $Pr(P) \to 1$ as $n \to \infty$. $\Box$

Observe that bending the definition of "witness" slightly to mean a model for ϕ[1] and correspondingly letting X be the number of models for ϕ, as before, one obtains $\sigma^2 = (m^2/n)O(\mu^2)$ [19]. Since an "interesting" lower bound is no less than $m/n = 1$ from Page 24, the second moment method cannot be applied to obtain a meaningful lower bound on the probability that a random expression has a model when X is the number of models. However, recent work, presented in Section 6.3, shows how to achieve a very good bound using the second moment method.

3.3 Markovian analysis and approximation by differential equations

Probabilistic analysis was originally applied to search algorithms, most notably variants of DPLL which is shown in Figure 1, to gain intuition about how an

[1]A model is merely a collection of unit clauses encompassing literals associated with all variables and a disjunction of all such representations of models can be added to ϕ without changing it. The resulting expression can easily be translated to a CNF expression.

effective search heuristic should be designed. More recently, it has been used to determine lower bounds on the satisfiability threshold of random (n, m, k)-CNF expressions. The variants that we would like to analyze are completely deterministic, their heuristic variable setting choices depending only on the reduced CNF expression implied by the current assignment of values to variables. Moreover, the search process induced by any of the variants jumps between states, where each state represents a reduced CNF expression and the partial assignment of values that causes it[2]. Thus, a search process may be modeled as a Markovian process.

Given DPLL with some search heuristic, if one can calculate the probability that terminal states representing models are reached, then one can determine the probability that a random (n, m, k)-CNF expression has a model or even the probability that a model will be found in some given number of jumps. Currently, this task is too ambitious due to the large number of states in question. Hence, analysis is typically applied to some other, simpler algorithm for which the states of the corresponding search space are a tiny subset of the states of the search space for DPLL: if the probability that a model is found by the simpler algorithm tends to 1 (or even is bounded by a constant in case we are looking for thresholds - see below) then the probability that DPLL finds a model also tends to 1.

Whereas DPLL may unassign and reassign values to a variable many times, the algorithms which have best yielded to analysis on random (n, m, k)-CNF expressions iteratively choose a variable, assign a value to it, then never change that value up to termination of the algorithm. We will call this class of algorithms *straight line* algorithms. All straight line algorithms are *incomplete*: that is, they may terminate without providing an answer. Although the number of states is reduced for straight line algorithms, the greatest reduction in the state space comes from coalescing into one state all states which have some attributes in common: for example, the clause counts of every possible clause width from 1 up to k, and the count of assigned variables. This particular coalescence is quite important and below we refer to it as the *spectral coalescence of states*.

Now the state space is small enough to work with but calculating state probabilities is trickier because, for a given search algorithm, the distribution of expressions at each coalesced state may depend on how that state was reached. Showing that such dependence does not exist, or is so weak that the results are not affected by it, is a crucial part of the probabilistic analysis of search algorithms. This requirement limits the type of algorithms that may be considered[3]. A general search algorithm typically introduces strong dependencies when reassigning values to variables. The dependence problem can be controlled to some

[2]The reduced expression is the original expression minus the clauses containing at least one literal which has value 1 in the partial assignment and minus literals which have value 0 in the partial assignment. In Figure 1 the lines $\phi_{d+1} \leftarrow \{c - \{\bar{v}\} : c \in \phi_d, v \notin c\}$ create reduced expressions.

[3]Observe that the expression generator also has an effect on state dependence so we are also limited in the types of generators chosen for analysis.

Procedure DPLL (ϕ)
Input: a CNF expression ϕ.
Output: either "unsatisfiable" or a model for ϕ.

```
begin
    $d := 0$;
    $M_d := \emptyset$;
    $\phi_d := \phi$;
    $V_P := \emptyset$.
    repeat the following until some statement outputs a value {
        if $\phi_d = \emptyset$ then return $M_d$;  // $M_d$ is a model for $\phi$
        if $\emptyset \in \phi_d$ then {            // $M_d$ falsifies a clause
            repeat the following until encountering a $v$ that is "tagged" {
                if $V_P = \emptyset$ then return "unsatisfiable";
                pop $v \leftarrow V_P$;
                $d := d - 1$;
            }
            push $V_P \leftarrow \bar{v}$;
            untag $v$;
            if $v$ is a complemented literal then $M_{d+1} := M_d \cup \{\bar{v}\}$;
            $\phi_{d+1} := \{C - \{\bar{v}\} : C \in \phi_d, v \notin C\}$;
            $d := d + 1$;
        } else {
            if there exists a pure literal in $\phi_d$ then
                choose a pure literal $v$;
            else if there is a unit clause in $\phi_d$ then
                choose a literal $v$ in a unit clause and "tag" $v$;
            else
                choose a literal $v$ in $\phi_d$ and "tag" $v$;
            push $V_P \leftarrow v$;
            if $v$ is an uncomplemented literal then $M_{d+1} := M_d \cup \{v\}$;
            $\phi_{d+1} := \{C - \{\bar{v}\} : C \in \phi_d, v \notin C\}$;
            $d := d + 1$;
        }
    }
end;
```

Figure 1: *DPLL algorithm for CNF expressions. If M_d is output, a model for ϕ has been found, otherwise no model exists for ϕ.*

extent by straight line algorithms and this is why this class of algorithms is so widely considered.

Myopic algorithms

Most performance results on random (n, m, k)-CNF expressions have been obtained for myopic algorithms. A straight line algorithm is called *myopic* [3] if, under the spectral coalescence of states, the distribution of expressions corresponding to a particular coalesced state can be expressed by its spectral components alone: that is, by the *number* of clauses of width i, for all $1 \leq i \leq k$, and the *number* of assigned variables. Thus, given random (n, m, k)-CNF expressions, the distribution of expressions for the coalesced "start" state is determined by the distribution of such expressions; and the distribution for the coalesced state corresponding to j assigned variables and $m_1, m_2, ..., m_k$ clauses of width $1, 2, ..., k$ is that of random $(m_1, n-j, 1)$-CNF, $(m_2, n-j, 2)$-CNF,..., $(m_k, n-j, k)$-CNF expressions, respectively for each clause width.

To determine whether an algorithm is myopic it is sufficient to show that no information about remaining clauses and literals, other than number, is revealed after assigning a value to a variable and eliminating clauses satisfied and literals falsified by that value. This is the case if variables are chosen at random and assigned values at random. In Section 6.1 more illuminating examples are given.

But some choices of variables will prevent an algorithm from being myopic. For example, if a pure literal is always chosen when one exists among remaining clauses we will not have a myopic algorithm because the distribution of number of occurrences in clauses depends on whether a pure literal is chosen or not. To see this consider the set of expressions containing 6 variables and 4 clauses, each of width 3. Label the variables $v_1, v_2, ..., v_6$. For each expression D let $p_D(e_1, e_2, e_3)$ be the probability that v_4,v_5, or v_6 is chosen first *and* the new expression resulting from the elimination of clauses (no literals are falsified since the chosen variable is a pure literal) is $\{\{v_1, v_2, v_3\}, \{\langle e_1, v_1\rangle, \langle e_2, v_2\rangle, \langle e_3, v_3\rangle\}, *\}$ where $\langle e, v\rangle$ means variable v occurs as an uncomplemented literal if $e = 1$ and as a complemented literal if $e = 0$, and $*$ means arbitrary. Let $\sigma(e_1, e_2, e_3) = \sum_{|D|\equiv 4} p_D(e_1, e_2, e_3)$. Obviously, $\sigma(e_1, e_2, e_3)$ should be independent of e_1, e_2, e_3 if choosing pure literals is myopic. More specifically, $\sigma(1, 1, 1)$ and $\sigma(0, 0, 0)$ should be equal. But, since v_1,v_2, nor v_3 can be pure in the latter case but cannot be pure in the former case, this requires that the pure literal selection heuristic would have to ignore v_1, v_2, and v_3 and pick only from v_4, v_5, or v_6. Reversing the roles of v_1, v_2, v_3 and v_4, v_5, v_6, the pure literal selection heuristic would have to ignore v_4, v_5, and v_6. Then no literals can be chosen! Thus, any form of the pure literal heuristic[4] cannot be myopic. Another way to choose literals which leads to non-myopic algorithms is via the well known "Johnson heuristic" where clauses are weighted by the number of literals they contain

[4]For example, if some weighting scheme is applied to the set of pure literals existing in an expression. For more details on this see [87]

and a literal is given a weight which is the sum of the weights of the clauses containing it.

In Section 6.1 some examples of myopic algorithms are discussed. What they have in common is that choosing variables can be based on the number of occurrences of literals in remaining clauses but ties must be broken randomly.

Differential equations to approximate discrete processes

The idea of using differential equations to approximate discrete random processes goes back at least to [72] and its application to the analysis of algorithms goes back to [66]. Given an initial system of m k literal clauses taken from a set of n variables, the process we will approximate is the movement of clauses out of the system as they become satisfied and the movement of i literal clauses down to $i-1$ literal clauses as literals are falsified during iterations of a straight line algorithm. To this end let $m_i(j)$ be the number of clauses containing i literals at the start of the jth iteration of a straight line algorithm. Initially, $m_k(1) = m$ and $m_i(1) = 0$ for $0 < i < k$. Observe a coalesced state is represented as the vector $\langle j, m_1(j), m_2(j), ..., m_k(j)\rangle$.

The following theorem from [1] (based on Theorem 2 of [97]) is used to approximate clause flows by differential equations. Hypothesis (i) ensures that $m_i(j)$ does not change too quickly from iteration to iteration of an algorithm; hypothesis (ii) tells us what we expect the rate of change of $m_i(j)$ to be and involves functions which are calculated from a knowledge of what the algorithm is doing; and hypothesis (iii) ensures that this rate of change does not change too quickly. For a random variable X, it is said $X = o(f(n))$ *always* if $\max\{x : Pr(X = x) \neq 0\} = o(f(n))$. The term "uniformly" refers to the convergence implicit in the $o(.)$ terms. By function $f(u_1, ..., u_k)$ *satisfies a Lipschitz condition on* $D \subseteq \mathbb{R}^j$ it is meant there exists a constant $L > 0$ such that $|f(u_1, ..., u_j) - f(v_1, ..., v_j)| \leq L \sum_{i=1}^{j} |u_i - v_i|$ for all $(u_1, ..., u_j)$ and $(v_1, ..., v_j)$ in D.

Theorem 2 *Let $m_i(j)$ be a sequence of real-valued random variables, $0 < i \leq k$ for some fixed k, such that for all i, all j, and all n, $|m_i(j)| \leq Bn$ for some constant B. Let $\mathbf{H}(j) = \langle\langle m_1(1), ..., m_k(1)\rangle, ..., \langle m_1(j), ..., m_k(j)\rangle\rangle$ be the state history of sequences.*

Let $I = \{\langle c_1, ..., c_k\rangle : Pr(\langle m_1(1), ..., m_k(1)\rangle = \langle c_1 n, ..., c_k n\rangle) \neq 0$ for some $n\}$.

Let D be some bounded connected open set containing the intersection of $\{\langle s, c_1, ..., c_k\rangle : s \geq 0\}$ with a neighborhood of $\{\langle 0, c_1, ..., c_k\rangle : \langle c_1, ..., c_k\rangle \in I\}$.

Let $f_i : \mathbb{R}^{k+1} \mapsto \mathbb{R}$, $0 < i \leq k$, and suppose that for some function $m = m(n)$

(i) *for all i and uniformly over all $j < m$*

$$Pr(|m_i(j+1) - m_i(j)| > n^{1/5} | \mathbf{H}(j)) = o(n^{-3}) \quad \textit{always};$$

(ii) *for all i and uniformly over all $j < m$*

$$E\{m_i(j+1)-m_i(j)|\mathbf{H}(j)\} = f_i(j/n, m_1(j)/n, ..., m_k(j)/n)+o(1) \quad \textit{always};$$

(iii) *for each i, the function f_i is continuous and satisfies a Lipschitz condition on D.*

Then

(a) *for $\langle 0, \hat{z}(1), ..., \hat{z}(k)\rangle \in D$ the system of differential equations*

$$\frac{dz_i}{ds} = f_i(s, z_1, ..., z_k), 0 < i \leq k$$

has a unique solution in D for $z_i : \mathbb{R} \mapsto \mathbb{R}$ passing through $z_i(0) = \hat{z}(i)$, $0 < i \leq k$, and which extends to points arbitrarily close to the boundary of D;

(b) *almost surely,*

$$m_i(j) = z_i(j/n) \cdot n + o(n),$$

uniformly for $0 \leq j \leq \min\{\sigma n, m\}$ and for each i, where $z_i(j)$ is the solution in (a) with $\hat{z}(i) = m_i(j)/n$, and $\sigma = \sigma(n)$ is the supremum of those s to which the solution can be extended.

□

How to use the solution to the differential equations

Theorem 2 is useful particularly because the differential equations are developed using *expectations* of clause counts and flows which in many cases are relatively easy to compute. Moreover, these expectations in the discrete world are translated to actual flow and count values in the solution to corresponding differential equations. Thus, the solution found in Theorem 2(b) for $m_2(j)$, say, does not deviate significantly from $E\{m_2(j)\}$ asymptotically. This is significant because in that case the function $m_2(j)$ is often enough to predict where a unit-clause-based algorithm will be successful probabilistically, in some sense.

Theorem 3 (*from [1]*) *Let A be a myopic algorithm that always chooses to satisfy a unit clause, when one exists among non-satisfied clauses. Let U_j be the event that on iteration j of A there are no empty or unit clauses existing among remaining non-satisfied clauses. Suppose*

$$m_2(j) < (1-\delta)(n-j)$$

for all $1 \leq j < (1-\epsilon)n$, $0 < \epsilon$ and $0 < \delta$ fixed, almost always. Then, there exists $\rho = \rho(\delta, \epsilon)$, $\rho > 0$, such that $Pr(U_{(1-\epsilon)n}) > \rho$. □

Theorem 3 can be applied directly to the result of Theorem 2(b) but there are two mysteries that need some clarification first. For one thing, Theorem 3 only applies to $j < (1-\epsilon)n$. This problem is disposed of on an ad-hoc basis. For example, in Section 6.1, page 34 the following is obtained for a particular myopic algorithm:

$$m_i(j) = \frac{1}{2^{k-i}}\binom{k}{i}\left(1-\frac{j}{n}\right)^i\left(\frac{j}{n}\right)^{k-i} m.$$

Then, using the binomial theorem,

$$\sum_{i=2}^{k} m_i(j) \quad = \quad \left(\left(1-\frac{j}{2n}\right)^k - \left(\frac{j}{2n}\right)^k - k\left(1-\frac{j}{n}\right)\left(\frac{j}{2n}\right)^{k-1}\right)m. \qquad (3)$$

Let $m/n = 1/\lambda$ and suppose $\lambda < 1$[5] and $1/2^k < \lambda$[6]. Set $\epsilon = \lambda/\binom{k}{2}$. After substituting $1-\epsilon$ for j/n on the right side of (3), expanding terms, and collecting powers of ϵ, it can be seen that (3) is bounded from above by $8\binom{k}{2}(\lambda/\binom{k}{2})^2 m/2^k = 8\lambda n/2^k\binom{k}{2} = (8/2^k)\epsilon n \le \epsilon n$. In other words, the number of width 2 and greater clauses remaining when $j = (1-\epsilon)n$ is no greater than the number of unset variables, with high probability. By Theorem 3 there are no unit clauses or empty clauses remaining with bounded probability. But, assuming the algorithm is myopic and inputs are random (n, m, k)-CNF expressions, one may randomly remove all but two literals from each clause resulting in a random $(n, m, 2)$-CNF expression. Such an expression is satisfiable with high probability and can be taken care of trivially.

The second mystery concerns the bound ρ which is only guaranteed to be a constant and may not be close to 1. In the case where the analysis is used to determine a bound on the probability of the existence of a model, finding a constant bound is all that is needed to prove the probability tends to 1. This is discussed in Section 3.5.

3.4 Sharp thresholds, minimality, and monotonicity

Let $X = \{x_1, ..., x_{n_e}\}$ be a set of n_e elements. Let A_X, a subset of the power set of X (denoted 2^X), be called a *property*. Call A_X a *monotone property* if for any $s \in A_X$, if $s \subset s'$, then $s' \in A_X$. Typically, A_X follows from a high-level description. For example, let $X = \mathcal{C}_{k,n}$ be the set of all non-tautological, width k clauses that can be constructed from n variables. Thus $n_e = 2^k\binom{n}{k}$ and any $s \in 2^{\mathcal{C}_{k,n}}$ is a k-CNF expression. Let $\mathcal{U}NSAT_{\mathcal{C}_{k,n}}$ be the property that a k-CNF expression constructed from n variables has no model. That is, any

[5]The case $\lambda > 1$ is not interesting in this context since the simple minded strategy of randomly removing all but two literals from every clause and applying a 2-CNF algorithm to the result succeeds with high probability in that case.

[6]Straight line algorithms for finding models will do poorly if $1/2^k \ge \lambda$ since almost all expressions have no models in that case.

$s \in \mathcal{UNSAT}_{\mathcal{C}_{k,n}}$ has no model and any $s \in 2^{\mathcal{C}_{k,n}} \setminus \mathcal{UNSAT}_{\mathcal{C}_{k,n}}$ has a model. If $s \in \mathcal{UNSAT}_{\mathcal{C}_{k,n}}$ and $c \in \mathcal{C}_{k,n}$ such that $c \notin s$, then $s \cup \{c\} \in \mathcal{UNSAT}_{\mathcal{C}_{k,n}}$ so the property $\mathcal{UNSAT}_{\mathcal{C}_{k,n}}$ is monotone.

An element $s \in A_X$ is said to be *minimal* if for all $s' \subset s$, $s' \in 2^X \setminus A_X$. For any $0 \leq p \leq 1$ and any monotone property $A_X \subset 2^X$ define

$$\mu_p(A_X) = \sum_{s \in A_X} p^{|s|}(1-p)^{n_e - |s|}$$

to be the probability that a random set has the monotone property. For the property $\mathcal{UNSAT}_{\mathcal{C}_{k,n}}$ (among others), s is a set of clauses, hence this probability measure does not match that for what we call random k-CNF expressions but comes very close with $p = m/(2^k\binom{n}{k}) \approx m \cdot k!/(2n)^k$.

Observe that $\mu_p(A_X)$ is an increasing function of p. Let $p_c(X)$ denote that value of p for which $\mu_p(A_X) = c$. If for any small, positive ϵ, $\lim_{|X|\to\infty}(p_{1-\epsilon}(X) - p_\epsilon(X))/p_{1/2}(X) = 0$ then A_X is said to have a sharp threshold. If $\lim_{|X|\to\infty}(p_{1-\epsilon}(X) - p_\epsilon(X))/p_{1/2}(X) > 0$ then A_X is said to have a coarse threshold. The following criteria for sharp thresholds is found in [29] and is developed from [47].

Theorem 4 *Let A_X be a monotone property such that $\lim_{|X|\to\infty} p_{1/2}(X) \to 0$. If the following two conditions are satisfied, then A_X has a sharp threshold.*

1. *For all $0 < c < 1$ and for all positive integers λ,*

$$\lim_{|X|\to\infty} \mu_{p_c(X)}(s' \subseteq s,\ s' \text{ is minimal for } A_X,\ \text{and } |s'| \leq \lambda) \to 0.$$

2. *For all $0 < c < 1$, for all positive integers λ, and for all $s^* \in 2^X \setminus A_X$ with $|s^*| = \lambda$,*

$$\lim_{|X|\to\infty} \mu_{p_c(X)}(s \in A_X, s \setminus s^* \in 2^X \setminus A_X | s^* \subseteq s) \to 0.$$

□

The first condition of Theorem 4 says that elements of A_X of bounded size have negligible probability of appearing. The second condition says the probability that a given s is in A_X is not affected much by conditioning on an element of bounded size that is not in A_X. But there are better conditions for sharp thresholds when one is dealing with k-CNF. Let $A_{\mathcal{C}_{k,n}}$ be a monotone property on width k CNF expressions in the sense that if s is a set of clauses verifying such a property then so does any set s' of clauses containing s. Let E be a set of values[7].

[7] In some applications $E = \{0, 1\}$, in some E has more than two elements.

Theorem 5 (adaptation from [30]) *If the following three conditions are satisfied, then the monotone property $A_{\mathcal{C}_{k,n}}$ has a sharp threshold.*

1. *For all $0 < c < 1$, $p_c(\mathcal{C}_{k,n}) = O(n^{1-k})$.*
2. *For all s minimal for $A_{\mathcal{C}_{k,n}}$, the number of variables of s is no greater than $(k-1) \cdot |s| - 1$.*
3. *For all $0 < c < 1$, for all t, for all $(\delta_1, ..., \delta_t) \in E^t$, and all $\gamma > 0$,*

$$\mu_{p_c(\mathcal{C}_{k,n})}(s \in 2^{\mathcal{C}_{k,n}} \setminus A^{\delta}_{\mathcal{C}_{k,n}}, |C^{\delta}_s| \geq \gamma \cdot n^{k-1}) \to 0$$

where $A^{\delta}_{\mathcal{C}_{k,n}}$ denotes the property $A_{\mathcal{C}_{k,n}}$ with the assignment $v_1 = \delta_1, ..., v_t = \delta_t$; and C^{δ}_s denotes the set of clauses C having at least one variable in $\{v_1, ..., v_t\}$ and is such that $s \cup C \in A^{\delta}_{\mathcal{C}_{k,n}}$.

$\square$

Theorem 5 may be used in a variety of ways depending on the monotone property $A_{\mathcal{C}_{k,n}}$. For example, the property that random (n, m, k)-CNF expressions are members of some polynomial time solvable class can be shown to have a sharp threshold for a number of polynomial time solvable classes. Two such illustrations are given in [31] for the classes of hidden Horn and of q-Horn expressions[8]

3.5 Lifting constant probability bounds to almost surely bounds

If a straight line algorithm is shown to find a model for a random expression with constant probability for a range of densities, this implies a random expression has a model, almost surely, for that range of densities. The appropriate theorem is the following [47].

Theorem 6 *Let $p_k(m, n)$ denote the probability that a random (n, m, k)-CNF expression has a model. For every $k \geq 2$ there exists a sequence $r_k(n)$ such that for any $0 < \epsilon$,*

$$\lim_{n\to\infty} p_k((r_k(n) - \epsilon)n, n) = 1 \text{ and } \lim_{n\to\infty} p_k((r_k(n) + \epsilon)n, n) = 0.$$

$\square$

[8]The monotone property $A_{\mathcal{C}_{k,n}}$ has to do with the emergence of structures which cause an expression *not* to be q-Horn or hidden Horn. See Section 4 for class definitions and Section 6.6 for some results and their meaning.

Theorem 6 says for each k there is a sharp threshold $r_k(n)$ at some density for every n such that a random expression almost surely has a model below the threshold and almost surely does not have a model above it. It follows that proving a model exists with probability bounded from below by a constant for a range of densities is sufficient to imply that a model exists almost surely for the same range of densities.

4 Some Efficiently Solvable Classes of CNF Expressions

All members of some easily identified, and in several cases well-known, classes of CNF expressions can be solved efficiently. For these it is natural to ask how often one encounters such an expression, whether expressions of a class can be recognized efficiently, or whether they even need to be. Some of these classes are incomparable meaning, for each of a pair of classes, there exists an expression which is a member of one but not the other. Nevertheless, probabilistic analysis can reveal some interesting properties of expressions in these classes such as:

1. What weakness does the class possess? That is, what critically distinguishes the class from more difficult classes?

2. Is one class much larger than another incomparable class?

In this section a few examples of polynomial time solvable classes are defined. In Section 6.6 probabilistic results on these classes will be reviewed. We omit discussion of a large number of polynomial time solvable classes such as 2-CNF, extended Horn [18], simple extended Horn [92], CC-balanced expressions [25], renameable Horn [73], and many others because we are interested here in how to use probability to make some statement about the relative size of such classes.

Definition 7 *A CNF expression is* Horn *if every clause it contains has at most one uncomplemented literal. A CNF expression is* hidden Horn *if reversing the polarity of the literals associated with some subset of variables (called a* switch set*) causes it to be Horn.* $\square$

This class is widely studied, in part because of its close association with Logic Programming. Namely, a Horn clause $\{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_i, y\}$ is equivalent to the rule $v_1 \wedge v_2 \wedge \ldots \wedge v_i \rightarrow y$ or the implication $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i \rightarrow y$ (where association is from right to left). However, the notion of causality is generally lost when translating from rules to Horn sets. Horn sets can be solved in linear time using unit resolution [33, 58, 91] (see algorithm UR, Figure 3). An important property of Horn sets is that a model, if one exists, which is minimal in the number of variables set to 1 is a *unique* minimum model with respect to 1 (this is referred to as the *minimal model* below). This property has important implications in, among other things, efficient algorithms for some other classes of CNF expressions.

Definition 8 *A CNF expression is a member of the class* SLUR *if, for all possible sequences of selected variables v, the* SLUR *algorithm of Figure 2 does not give up on that expression.* □

This class was developed in [89] as a generalization of other classes including Horn, extended Horn [18], simple extended Horn [92], and CC-balanced expressions [25]. It is peculiar in that it is defined based on an algorithm rather than on properties of expressions. The algorithm of Figure 2, called SLUR for Single Lookahead Unit Resolution, selects variables sequentially and arbitrarily and considers a one-level lookahead, under unit resolution, of both possible values that the selected variable can take. Observe that due to the definition of this class, the question of class recognition is avoided.

The origin of the class called q-Horn is [16, 15] but it is also a special case of work considered in [94, 95]. Represent a CNF expression $\phi = \{c_1, c_2, ..., c_m\}$ as a $(0, \pm 1)$ matrix A^ϕ, columns indexed on variables, rows indexed on clauses, and such that $A^\phi_{i,j} = 1$ if literal $v_j \in c_i$, $A^\phi_{i,j} = -1$ if literal $\bar{v}_j \in c_i$ and $A^\phi_{i,j} = 0$ if $v_j \notin c_i$ and $\bar{v}_j \notin c_i$.

Definition 9 *Suppose some of the columns of A^ϕ can be scaled by -1 and the rows and columns permuted so that there is a partition of the resulting matrix into quadrants*

$$\left(\begin{array}{c|c} A^1 & E \\ \hline D & A^2 \end{array}\right)$$

where submatrix A^1 has at most one $+1$ entry per row, submatrix D contains only -1 or 0 entries, submatrix A^2 has no more than two nonzero entries per row, and E has only 0 entries. Then ϕ is q-Horn. □

Such a partition, if it exists, can be found in linear time. To solve a q-Horn expression perform the partition, solve A^1 (a Horn expression), cancel rows in D and A^2 which are satisfied by the minimal model for A^1 (or return "unsatisfiable" if there is no model), solve A^2 less the canceled rows (a 2-CNF expression) and return models for A^1 and A^2 (or "unsatisfiable" if A^2 has no model).

Expressions in the class q-Horn had been thought to be close to what might be regarded as the largest easily definable class of polynomially solvable propositional expressions due to results in [16]. Let $\{v_1, v_2, ..., v_n\}$ be a set of variables, and P_k and N_k, $P_k \cap N_k = \emptyset$, be subsets of $\{1, 2, ..., n\}$ such that the kth clause in ϕ is given by $\vee_{i \in P_k} v_i \vee_{i \in N_k} \bar{v}_i$. Construct the following system of inequalities:

$$\sum_{i \in P_k} \alpha_i + \sum_{i \in N_k} (1 - \alpha_i) \quad \leq \quad Z, \quad (k = 1, 2, ..., m), \text{ and} \tag{4}$$

```
Procedure SLUR($\phi$):
Input: a CNF expression $\phi$.
Output: a model $M$ for $\phi$, if one exists, or “unsatisfiable,” or “give up.”

begin
    $M := \emptyset$;
    $\phi :=$ UR($\phi, M$);
    if $\emptyset \in \phi$ then return (“unsatisfiable”);
    $level := 0$;
    repeat the following while $\phi \neq \emptyset$ {
        choose arbitrarily a variable $v \in \phi$;
        $M_1 := M$;
        $M_2 := M$
        $\phi_1 :=$ UR($\{C - \{v\} : C \in \phi, \bar{v} \notin C\}, M_1$)
        $\phi_2 :=$ UR($\{C - \{\bar{v}\} : C \in \phi, v \notin C\}, M_2$)
        if $\emptyset \in \phi_1$ and $\emptyset \in \phi_2$ {
            if $level = 0$ then return (“unsatisfiable”);
            else return (“give up”);
        } else {
            arbitrarily choose $i$ so that $\emptyset \notin \phi_i$;
            $\phi := \phi_i$;
            $M := M_i$;
            if $i = 2$ then $M := M \cup \{x\}$
        }
        $level := 1$;
    }
    return ($M$);
end;
```

Figure 2: SLUR algorithm.

```
Procedure UR($\phi, M$): // Applies unit resolution until exhaustion
Input: a CNF expression $\phi$ and partial assignment $M$.
Output: a CNF expression and an updated $M$.

begin
    repeat the following {
        let $\{l\} \in \phi$ be a unit clause;
        if $l$ is an uncomplemented literal then $M := M \cup \{l\}$;
        $\phi := \{C - \{complement(l)\} : C \in \phi, l \notin C\}$;
    } while $\emptyset \notin \phi$ and there is a unit clauses in $\phi$;
    return ($\phi$);
end;
```

Figure 3: Unit resolution algorithm.

$$0 \leq \alpha_i \leq 1 \qquad (i = 1, 2, ..., n). \tag{5}$$

where $Z \in R^+$. The minimum value of Z that satisfies (4) and (5) is the *satisfiability index* for ϕ. If ϕ's satisfiability index is no greater than 1, then it is q-Horn. However, the class of all expressions with a satisfiability index greater than $1 + 1/n^\epsilon$, for any fixed $\epsilon < 1$, is NP-complete.

The last class of this section is called the Matched class [46][9]. For a given expression ϕ, define G^ϕ to be an undirected bipartite graph with vertex sets C^ϕ, whose elements are the clauses of ϕ, and V^ϕ, whose elements are the variables of ϕ, and edge set $E^\phi = \{\langle v, c\rangle : v \in V^\phi, c \in C^\phi, \text{ and variable } v \text{ is in clause c}\}$. A matching in G^ϕ is a disjoint subset of edges $B \subset E^\phi$. A maximum matching in G^ϕ is a matching in G^ϕ containing the maximum possible number of edges. A total matching in G^ϕ is a matching in G^ϕ where every $c \in C^\phi$ is in some edge $e \in B$.

Definition 10 *A CNF expression ϕ is a member of the Matched class if G^ϕ has a total matching.* $\square$

It is well known that, due to an augmenting path algorithm of Edmonds, a maximum matching can be found for bipartite graphs in polynomial time. This implies a total matching can be found for G^ϕ in polynomial time, if one exists.

It is straightforward to show that the SLUR, q-Horn, and Matched classes are incomparable.

5 A Digest of Probabilistic Results

Goldberg was among the first to investigate the frequency with which DPLL algorithms return a model quickly or return a short proof that no model exists. He provided an average-case analysis of a variant of DPLL which does not handle pure literals or unit clauses [53]. The analysis was based on a different parameterized distribution that generates random (n, m, p)-CNF expressions.

Definition 11 *A random (n, m, p)-CNF expression contains m clauses, each generated independently according to a random process described as follows. Let V be a set of n variables and let $0 \leq p \leq 1/2$. For every variable $v \in V$ independently and uniformly do one of the following: add literal v to clause C with probability p, add literal $\bar{v}$ to C with probability p, add neither v nor $\bar{v}$ to C with probability $1 - 2p$.* $\square$

Goldberg showed that, for random (n, m, p)-CNF expressions, the DPLL variant has average complexity for bounded from above by $O(m^{-1/\log(p)}n)$ for

[9]The observation that a total matching implies a model was credited to Adam Rosenberg by Tovey [93].

any fixed $0 < p < 1$. This includes the "unbiased" sample space when $p = 1/3$ and all expressions are equally likely. Later work [54] showed the same average-case complexity even if pure literals are handled as well. The scientific and engineering communities are often interested in refutation proofs, but Goldberg made no mention of the frequency of unsatisfiable random (n, m, p)-CNF expressions over the parameter space of that distribution.

However, Franco and Paull [45] pointed out that large sets of random (n, m, p)-CNF expressions, for fixed $0 < p < 1/2$, are dominated by easily *satisfiable* expressions: that is, a random assignment of values to the variables of a random expression is a model for that expression with high probability. This result is refined somewhat in [41] where it is shown that a random assignment is a model for a random (n, m, p)-CNF expression with high probability if $p > \ln(m)/n$ and a random (n, m, p)-CNF expression is unsatisfiable with high probability if $p < \ln(m)/2n$. In the latter case, a "proof" of unsatisfiability is trivially found with high probability because a random (n, m, k)-CNF expression for this range of p usually contains at least one empty clause, which can easily be located, and implies unsatisfiability. The case $p = c\ln(m)/n$, $1/2 \le c \le 1$, was looked at in [44] where it was shown that a random (n, m, p)-CNF expression is satisfiable with high probability if $\lim_{n,m\to\infty} m^{1-c}/n^{1-\epsilon} < \infty$, for any $0 < \epsilon < 1$. These results show two things regarding random (n, m, p)-CNF expressions: 1) there seems to be some threshold, probably sharp, for the probability that a random expression is satisfiable and 2) expressions generated on both sides of the threshold are usually trivially solved because they either contain empty clauses or can be satisfied by a random assignment. In other words, only a small region of the parameter space is capable of not being dominated by trivially solved satisfiable or unsatisfiable expressions: namely, when the average clause width is $c\ln(m)/n$, $1/2 \le c \le 1$. Because of this, random (n, m, p)-CNF generators are not considered interesting by many.

Nevertheless, there has been significant interest in Goldberg's distribution and the results of some of this work is summarized in the two dimensional chart of Figure 4 which partitions the entire parameter space of that distribution: the vertical axis measures the average clause width $(p \cdot n)$ and the horizontal axis measures the density (m/n). Each result is presented as a line through the chart with a perpendicular arrow. Each line is a boundary for the algorithm labeling the line and the arrow indicates that the algorithm has polynomial average time performance in that region of the parameter space that is on the arrow side of the line (constant factors are ignored for simplicity). Goldberg's result appears in the upper right corner, labeled **Goldberg**, occupying only a boundary of the parameter space (hence no arrow) and shows the algorithm analyzed by Goldberg has polynomial average time performance if p is a constant. Iwama's counting algorithm [59], labeled **Counting**, does better. If clauses are resolved only if the pivot appears in two (remaining) clauses then a null clause verifying unsatisfiablity will be obtained in polynomial average time below the lines labeled **Limited Resolution** [43]. If pure literal reductions are added to the algorithm analyzed by Goldberg then polynomial average time is achieved below the line labeled **Pure Literals** as well as above the **Goldberg** line [84]. But an improved result, shown by the lines labeled **Unit Clauses**, is obtained merely by repeatedly satisfying unit clauses until a null clause is generated (or the algorithm gives up) [42]. Results for Search Rearrangement Backtracking [85] are disappointing (shown bounded by the two lines labeled **Backtracking**) but a slight variant, always choosing variables that are in a positive clause (if there is no positive clause, satisfiability is determined by setting the remaining variables to 0), is fantastic as is shown by the line labeled **Probe Order Backtracking** [83]. Observe that the combination of probe order backtracking and unit clauses yield polynomial average time almost everywhere for random (n, m, p)-CNF expressions. Now we turn our attention to the possibly more robust random (n, m, k)-CNF expressions.

Franco and Paull in [45] (see [82] for corrections) also considered the probabilistic performance of Goldberg's variant of DPLL for random (n, m, k)-CNF expressions. They showed that for all $k \geq 3$ and every fixed $m/n > 0$, with probability $1 - o(1)$, the variant takes an *exponential* number of steps to report a result: that is, either to report all ("cylinders" of) models, or that no model exists. The first upper bound on r_k, namely the smallest value of m/n such that the expected number of (n, m, k)-CNF models tends to 0, was also presented in that paper: the probability that a random expression is unsatisfiable is $1 - o(1)$ if $m/n > -\ln(2)/\ln(1 - 2^{-k})$ (the result of Section 3.1 which is approximately $0.69 \cdot 2^k$).

Later, in a series of two papers [20, 21], Chao and Franco presented some useful insights which influenced the lower bound probabilistic analysis of Satisfiability and Coloring algorithms in following years (for example, [2, 6, 7, 8, 23, 48]). Unlike upper bounds, which are probabilistic counting arguments, they produced lower bounds for r_k which are algorithmic. The algorithms they considered are shown in Figure 5 as a single algorithm, called SCA, with one parameter s. SCA performs exactly like DPLL until either a model is found or the first backtrack is attempted and, in the latter case, the algorithm gives up. Thus, a positive result for algorithm SCA is also a positive result for DPLL. The analysis of this algorithm was intended primarily to determine conditions under which efficient performance was likely and secondarily to determine a lower bound for r_k. It was based on considering the "flow" of clauses between levels of clause sets, where level i consists of all clauses of i literals at any iteration j of SCA. The technique of flow analysis is discussed in detail in Section 6.1.

Using flows, in [20] it is shown that the Unit Clause (UC) algorithm (Algorithm SCA, Figure 5 with $s = 1$) has positive probability of finding a model for random (n, m, k)-CNF expressions when $m/n < 8/3 = 2.66..$ and, when combined with a "majority" rule, for $m/n < 2.9$. In [21] the Generalized Unit Clause (GUC) algorithm (Algorithm SCA with $s = k$) is shown to find a model with bounded probability when $m/n < (0.77)2^k((k-1)/(k-2))^{k-2}/(k+1)$ and $4 \leq k \leq 40$ and with probability $1 - o(1)$ when $m/n < (0.46)2^k((k-1)/(k-2))^{k-2}/(k+1)$ and $4 \leq k \leq 40$. This was improved by

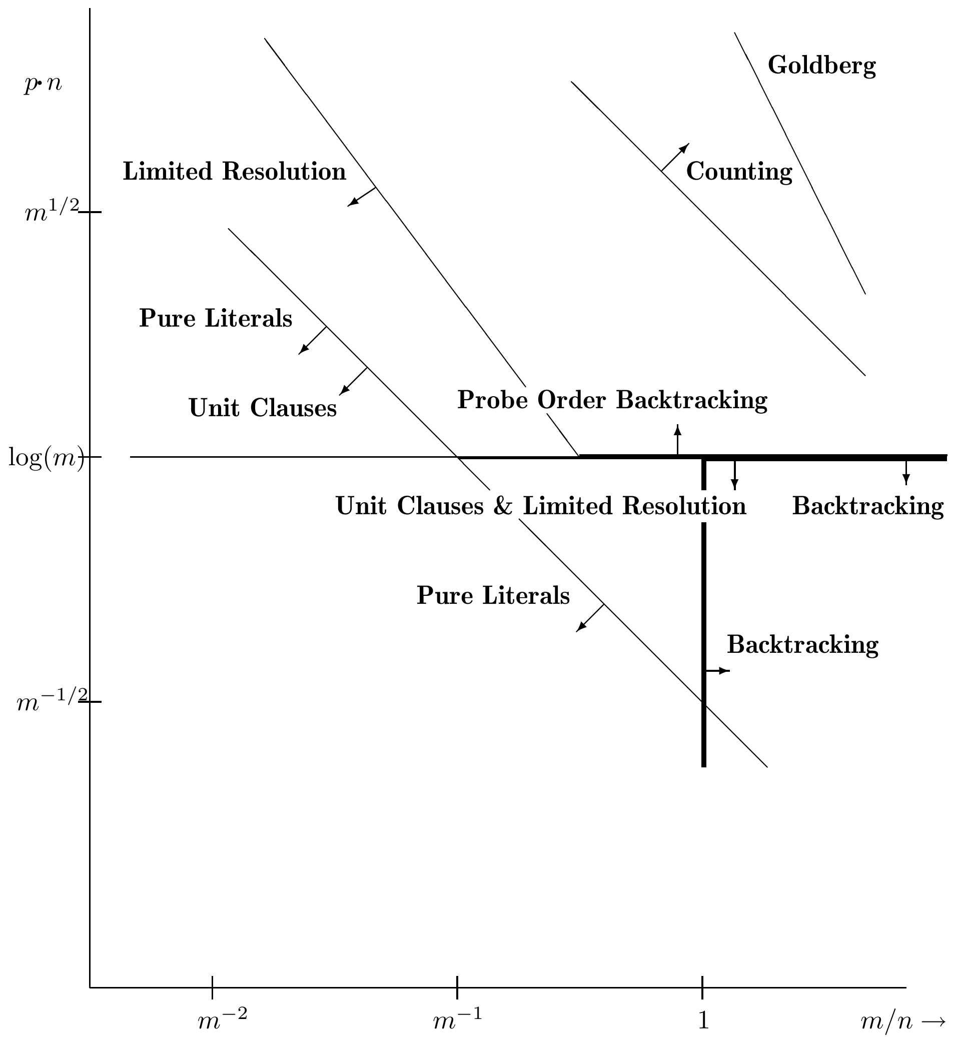

Figure 4: The parameter space of Goldberg's distribution partitioned by polynomial average time solvability. Pick a point in the parameter space. Locate the lines with names of algorithms and arrows on the side of the line facing the chosen point. Random formulas generated with parameters set at the specified point are solved in polynomial average time by the named algorithms.

```
Procedure SCA ($\phi, s$):
Input: a CNF expression $\phi$, integer $s$.
Output: either a model for $\phi$ or "give up".

begin
    $M := \emptyset$;
    repeat the following until some statement outputs a value {
        if $\phi = \emptyset$ then return $M$;
        if $\emptyset \in \phi$ then return "gives up";
        for $i := 0$ to $k$ do $\mathcal{C}_i := \{C : C \in \phi$ and $|C| = i\}$;
        $q := \min\{i : \mathcal{C}_i \neq \emptyset\}$;
        $L_1 := \{l : \exists c \in \mathcal{C}_q$ such that $l \in c\}$;
        $L_2 := \{l : \exists c \in \phi$ such that $l \in c\}$;
        if $q \leq s$ then choose $l$ randomly from $L_1$;
        else choose $l$ randomly from $L_2$;
        if $l$ is an uncomplemented literal then $M := M \cup \{l\}$;
        $\phi := \{c - \{\bar{l}\} : c \in \phi$ and $l \notin c\}$;
    }
end;
```

Figure 5: *Smallest Clause Algorithm for CNF expressions.*

Chvátal and Reed [23] who showed that the Shortest Clause (SC) algorithm (Algorithm SCA with $s = 2$) finds a model with probability $1 - o(1)$ when $m/n < (0.125)2^k((k-1)/(k-3))^{k-3}/k$ and $3 \leq k$. Observe that, combined with previous results, this tells us random (n, m, k)-CNF expressions are nearly always satisfiable if $m/n < c_1 2^k/k$ and nearly always unsatisfiable if $m/n > c_2 2^k$ for some positive constants c_1 and c_2 and sufficiently large n.

According to a result of Friedgut (see Theorem 6) there is a sharp satisfiability threshold r_k for every $k \geq 3$, and we know from the above results that it must be no greater than some constant times 2^k but no less than some constant times $2^k/k$. But where, precisely, is it? The answer is interesting for various reasons including that it may provide some insight about the performance of DPLL variants. These variants seem to do well up to m/n about equal to $2^k/k$ but after that, for fixed m/n, their performance seems to suffer. Thus, a threshold result of order 2^k suggests a rather "hard" region for DPLL algorithms. The question has been essentially solved recently. Before this, nearly all concerned researchers had believed the threshold is of order 2^k and for this reason it became known in 1999 as the "*Why* 2^k?" problem. In 2002 Achlioptas and Moore [4] showed that nearly everyone was right, using the second moment method on a variant of Satisfiability which bounds it in some way. Details of the analysis are instructive and are presented in Section 6.3.

The most active research on that portion of the parameter space where expressions with models are numerous has considered the cases $k = 2$ and $k = 3$. Since 2-CNF expressions are polynomial time solvable, for $k = 2$ the issue can only be whether the threshold exists and, if so, where is it. Chvátal and Reed [23], Goerdt [51] and Fernandez de la Vega [39] independently answered these questions: they determined $r_2 = 1$. It is important to observe that 2-CNF expressions being solvable in polynomial time [26] means that there is a *simple* characterization of unsatisfiable 2-CNF expressions. Indeed, both [23] and [51] make full use of this characterization as they proceed by focusing on the emergence of the "most likely" unsatisfiable random (n, m, k)-CNF expressions. Also using this characterization, Bollobas et al. [14] recently completely determined the "scaling window" for random $(n, m, 2)$-CNF expressions, showing that the transition from satisfiability to unsatisfiability occurs for $m = n + \lambda n^{2/3}$ as λ goes from $-\infty$ to $+\infty$.

For $k = 3$, less progress has been made: the value of r_3 has not been established although ever improving bounds for it have progressively been reported. Since $r_2 = 1$, $1 \leq r_3$ follows trivially. Broder, Frieze and Upfal [17] proved that the pure literal heuristic alone, with no backtracking, almost always sets all the variables for $m/n \leq 1.63$. They used Martingales to offset distribution dependency problems as pure literals are uncovered. (Martingales will not be discussed here since more recent techniques appear to be more powerful). Mitzenmacher [76] showed that this bound is tight for the pure literal heuristic. Frieze and Suen determined the probability of success of Algorithms SC and GUC. In particular, they showed that for $m/n < 3.003..$, both heuristics succeed with positive probability. Moreover, they proved that a modified version of GUC, which performs a very limited form of backtracking, succeeds almost surely for such values of m/n. Years later, Achlioptas [2] showed, using a flow analysis, that for $m/n \leq 3.145$, a random (n, m, k)-CNF expression is satisfiable with probability $1 - o(1)$ by changing SC slightly to choose two literals at a time coming from a clause with two unfalsified literals remaining. In [3] Achlioptas and Sorkin take this approach to the limit with the discovery of an algorithm which almost surely finds a model if $m/n < 3.26$. By *take to the limit* we mean that no other myopic algorithm for Satisfiability can perform better probabilistically. However, Kaporis et.al. [63] found a workaround to this "barrier" and analyzed a simple non-myopic greedy algorithm for Satisfiability. A description is outlined in Section 6.2, controls the dependence problem by considering a different generator of expressions such that probabilistic results also hold for random $(n, m, 3)$-CNF expressions. Their result is that there exists an algorithm for Satisfiability which almost always finds a model when $m/n < 3.42$. With this analysis machinery in place, and considering the results of numerous experiments, it will not be long before better results are reported. Because an analysis tends to direct algorithm development, it is hoped future probabilistic results will reveal new generally useful search heuristics, or at least explain the mechanics of existing search heuristics.

Algorithmic methods, applied successfully to DPLL on the satisfiable side, do not help in finding good upper bounds on r_k. Instead, successful results, based on counting, are due to the application of the first and second moment methods. It was stated earlier that a simple bound of $r_k < -\ln(2)/\ln(1-2^{-k})$ is obtained from the first moment method applied to the number of models of a random expression, later referred to as N_M, and the second moment method on N_M does not give a better bound because the variance is too large. However, the first moment method can yield better results if the count being measured is changed. For example, from the above $r_3 < 5.19$ but in [62] the count is allowed to be reduced due to the observation that, with high probability, a large number of variables have no effect on models. Taking the expectation of N_M assuming an appropriately reduced number of variables leads to $r_3 < 4.762$. A weaker bound using the same idea but on a slight variation of the generation of random 3-CNF expressions, was obtained independently in [37]. But a more clever idea, introduced independently in [67] and [34] is based on identifying, for a given expression ϕ, a set of "critical" models that is far smaller than the complete set of models of ϕ. The expected number of critical models is a tighter upper bound on the probability that an expression is satisfiable than the expected number of models and it can be found with some cleverness as follows.

Call a truth assignment a *critical model* for CNF expression ϕ if it is a model for ϕ and flipping exactly the value of one 0-valued variable to 1 results in an assignment that is not a model for ϕ. A critical model can be obtained from any model by flipping 0 values to 1 until such flips can only result in an assignment which is not a model. Therefore, every satisfiable expression has at least one critical model and a bound on the expected number of critical models is a bound on the probability there exists a model. Given a critical model M and a specific flip, there are $\binom{n}{k}$ possible clauses which are excluded from ϕ and there is at least one of $\binom{n-1}{k-1}$ possible clauses containing the flipped variable which must be included to cause ϕ to be falsified when that variable is flipped. Hence, the probability that an assignment is a model is $(1-2^{-k})^m$ and the probability that a clause is the one which causes ϕ to be falsified given ϕ is satisfied by M is $\binom{n-1}{k-1}/(2^k-1)\binom{n}{k}$ which is approximately $k/((2^k-1)n)$. From the latter, the probability that M is critical but a given single flip results in not satisfying ϕ is $1-(1-k/((2^k-1)n))^m$. If s is the number of possible single flips for M, then the probability that M is critical given it is a model for ϕ is bounded from above[10] by $(1-(1-k/((2^k-1)n))^m)^s$. The probability that M is critical given s flips are possible is $(1-2^{-k})^m(1-(1-k/((2^k-1)n))^m)^s$ and this is also the expected value of the indicator variable which takes value 1 if and only if the corresponding M is critical. The expected number of critical models is then

$$\begin{aligned}(1-2^{-k})^m \sum_{s=0}^{n} \binom{n}{s} (1-(1-k/((2^k-1)n))^m)^s \\ = (1-2^{-k})^m(2-(1-k/((2^k-1)n))^m)^n.\end{aligned}$$

[10]A few details concerning dependence must be added here to verify the bound.

Setting this equal to 1 provides the bound (approximately)

$$r_k \quad < \quad -\frac{\ln(2 - e^{-(k/(2^k-1))r_k})}{\ln(1-2^{-k})}. \tag{6}$$

Comparing Inequalities 1 and 6 we see the difference is only in the exponential term. For $k = 3$ the upper bound given by Inequality 6 is $r_3 < 4.667$. However, a generalization of this method results in a bound $r_3 < 4.601$. In [36] a still further improvement to $r_3 < 4.506$ is reported. Experimental results suggest that $r_3 = 4.26$. For more information on this area of research the reader is referred to [35].

Returning to algorithms, pessimistic results have been reported for resolution-based algorithms, including all DPLL variants, in the case of random unsatisfiable expressions. Resolution is a general procedure used primarily to certify that a given CNF expression has no model. The idea predates the often cited work reported in [86]. Let c_1 and c_2 be clauses such that there is exactly one variable v that occurs negated in one clause and unnegated in the other. Then, the *resolvent* of c_1 and c_2, denoted by $\mathcal{R}^{c_1}_{c_2}$, is a clause which contains all the literals of c_1 and all the literals of c_2 except for v and its complement. That is, $\mathcal{R}^{c_1}_{c_2} = \{l : l \in c_1 \cup c_2 \setminus \{v, \bar{v}\}\}$. The variable v is called a *pivot* variable.

The usefulness of resolvents derives from the following Lemma, which is straightforward to prove.

Lemma 12 *Let ϕ be a CNF expression. Suppose there exists a pair $c_1, c_2 \in \phi$ of clauses such that $\mathcal{R}^{c_1}_{c_2} \notin \phi$ exists. Then the CNF expression $\phi \cup \{\mathcal{R}^{c_1}_{c_2}\}$ is functionally equivalent to ϕ.* $\square$

A resolution algorithm for CNF expressions, making use of Lemma 12 and the fact that a clause containing no literals cannot be satisfied by any truth assignment, is presented in Figure 6. If the algorithm outputs "unsatisfiable" then the set of all resolvents generated by the algorithm is a *resolution refutation* for the input expression, certifying that it has no model. If the output is a set of variables, then the assignment obtained by setting those variables to 1 and all others to 0 is a model for the input expression.

In [27] it is shown that if there is a short DPLL refutation for a given unsatisfiable expression, then there must be a short resolution refutation for the same expression. Therefore, resolution is potentially more efficient than DPLL. However, finding a *shortest* resolution refutation is generally hard. Moreover, the more restricted nature of DPLL algorithms, namely finding all resolvents involving a particular pivot variable v and then removing clauses containing v and $\bar{v}$ from ϕ, seems to result in a better intuitive grasp of effective search heuristics and this seems to be the reason DPLL is preferred to resolution, in general.

In any case, the pessimistic probabilistic results for resolution, which also apply to DPLL, begin with establishing the root cause: namely, expression sparse-

```
Procedure Resolution ($\phi$)
Input: a CNF expression $\phi$.
Output: either “unsatisfiable” or a model for $\phi$.

begin
    $M_1 := \emptyset$;
    $M_2 := \emptyset$;
    repeat the following until some statement below outputs a value {
        if $\emptyset \in \phi$ then return “unsatisfiable”;
        if there are two clauses $C_1, C_2 \in \phi$ such that $\mathcal{R}^{C_1}_{C_2} \notin \phi$ exists then
            $\phi := \phi \cup \{\mathcal{R}^{C_1}_{C_2}\}$;
        else {
            repeat the following while there is a pure literal $l \in \phi$ {
                if $l$ is an uncomplemented literal then $M_1 := M_1 \cup \{l\}$;
                $\phi := \{C : C \in \phi, l \notin C\}$;
            }
            repeat the following while there is an uncomplemented clause $C \in \phi$ {
                choose variable $v \in C$;
                reverse the polarity of all occurrences of $v$ and $\bar{v}$ in $\phi$;
                $M_2 := M_2 \cup \{v\}$;
            }
            $M := M_1 \cup M_2$;
            return $M$;
        }
    }
end;
```

Figure 6: *Resolution algorithm for CNF expressions.*

ness [24]. Without getting too technical at this time, sparseness is a measure of the number of times pairs of clauses have a common literal or complementary pair of literals; any "moderately large" subset $\mathcal{C}$ of clauses taken from a sparse expression must contain a "large" number of variables that occur exactly once in $\mathcal{C}$.

Sparse expressions force a superpolynomial number of resolution steps for the following reason. Let $\mathcal{P}^\phi$ be the minimum set of clauses that is the result of repeated applications of the resolution rule starting from a sparse unsatisfiable CNF expression ϕ and ending with the null clause. By sparsity, any moderately sized subset $\mathcal{C}$ of clauses taken from ϕ must contain a large number of variables that occur exactly once in $\mathcal{C}$. This forces at least one clause of high width to exist in $\mathcal{P}^\phi$. But, almost all "short" resolution refutations contain no long clauses after eliminating all clauses satisfied by a particular small random partial assignment ρ. Moreover, resolution refutations for almost all unsatisfiable random (n, m, k)-CNF expressions with clauses satisfied by ρ removed are sparse and, therefore, must have at least one high width clause. Consequently, almost all unsatisfiable random (n, m, k)-CNF expressions have long resolution refutations. Ideas leading up to a concise understanding of this phenomenon can be found in [11, 12, 13, 24, 50, 56, 96]. Despite considerable tweaking, the best we can say right now is that the probability that the length of a shortest resolution proof is bounded by a polynomial in n when $m/n > c'_k \cdot 2^k$, c'_k some constant, and $\lim_{m,n\to\infty} m/n^{(k+2)/(4-\epsilon)} < 1$, where ϵ is some small constant, tends to 0 and when $\lim_{m,n\to\infty} m/n > (n/\log(n))^{k-2}$ tends to 1. These results are restated with a little more detail in Section 6.4.

The failure of resolution has led to the development of new techniques for certifying unsatisfiability. A major success with respect to probabilistic analysis is reported in [52] and described in Section 6.5. In essence, for a random (n, m, k)-CNF expression ϕ, one finds a bound $N^+_{m,n}$ on the number of variables needed to be set to 1 to satisfy clauses containing only uncomplemented literals and a bound $N^-_{m,n}$ on the number of variables needed to be set to 0 to satisfy clauses containing only complemented literals. If $N^+_{m,n} + N^-_{m,n} > n$ then at least one variable would have to be set to both 1 and 0 to satisfy the given expression. Since this is impossible if ϕ is satisfiable, this test can provide certification that ϕ is unsatisfiable. In [52] spectral techniques are used to obtain bounds sufficient to show that certifying unsatisfiability in polynomial time can be accomplished with high probability when $\lim_{m,n\to\infty} m/n > n^{k/2-1+o(1)}$ Results on resolution are discussed further in Section 6.4. Figure 7 summarizes probabilistic results on algorithms for random (n, m, k)-CNF expressions.

The probabilistic analysis of properties of CNF expressions can develop insights into the nature of "hard" problems as well as the potential effectiveness of preprocessing expressions before search algorithms are applied. For example, consider checking whether a given expression is a member of a polynomial time solveable class before search. If so, the search can be eliminated entirely. Such a strategy might result in a drastic reduction of overall execution time. How-

	Efficient in Probability - $m/n < ...$		
Algorithm	$k = 3$	$k > 3$ $(Pr > c)$	$k > 3$ $(Pr \to 1)$
Goldberg	none	none	none
Pure Literals (Broder...)	1.63	not reported	not reported
UC (unit clauses)	2.66	$\frac{2^k}{k}\frac{1}{2}\left(\frac{k-1}{k-2}\right)^{k-2}$	none
UC + majority	2.9	not determined	not determined
GUC (shortest clauses)	3.003	$\frac{3\cdot 2^k}{4k}\left(\frac{k-1}{k-2}\right)^{k-2}\left(\frac{k}{k+1}\right)$	$0.46\frac{2^k}{k}\left(\frac{k-1}{k-2}\right)^{k-2}\left(\frac{k}{k+1}\right)$
SC (1,2 width clauses)	3.003	$O(2^k/k)$	$\frac{2^k}{8k}\left(\frac{k-1}{k-3}\right)^{k-3}\left(\frac{k-1}{k-2}\right)$
SC (2 literals at a time)	3.145	$O(2^k/k)$	$O(2^k/k)$
Best Myopic	3.26	$O(2^k/k)$	$O(2^k/k)$
Greedy (Kaporis...)	3.42	$O(2^k/k)$	$O(2^k/k)$

	Efficient in probability - $m/n > ...$		
Algorithm	$k = 3$	$k > 3$ $(Pr > c)$	$k > 3$ $(Pr \to 1)$
Restricted Width	0.66n	—	$n^{k-2}\left(\frac{2^k}{2k!}\right)$
DPLL variant	$n/\log(n)$	—	$n^{k-2}\left(\frac{1}{\log(n)}\right)^{k-2}$
Hitting Set	$n^{1/2+\epsilon}$	—	$n^{k/2-1+o(1)}$

Figure 7: Probabilistic behavior of algorithms on random (n, m, k)-CNF expression as a function of density (m/n). The top table, except for Goldberg's algorithm, applies to incomplete algorithms which either find a model or give up. Such algorithms perform well in probability for densities (m/n) below a certain quantity. Table entries indicate known bounds on those quantities. Two types of results are shown for $k > 3$: probability of success bounded from below by a constant (column heading includes $Pr > c$) and probability of success tending to 1 (column heading includes $Pr \to 1$). Much work has been done for the case $k = 3$ and those results are in a separate column. In that case, the probability of success tends to 1 only for the algorithm of Broder. However, the addition of a limited form of backtracking brings GUC and SC into that category as well. Table entries $O(2^k/k)$ have not been published but either are obvious or have been communicated personally. These are interesting in light of the result that the Satisfiability threshold is $O(2^k)$. The bottom table applies to algorithms which intend to verify unsatisfiability.

ever, this seems unlikely to succeed for any of the well known efficiently solvable classes due to probabilistic studies of random expressions such as those described in Section 6.6. According to those results the likelihood of the existence of instances of such classes is extremely low among reasonably hard random expressions. The analysis explains why: namely the highly likely presence of specific cyclic structures (this is explained in Section 6.6).

Insights developed from a probabilistic analysis also can be quite surprising. For example, using density (m/n) as a parameter, it is found that several well studied, different, complex polynomially solvable classes are rare when $m/n > 4/(k^2 - k)$ [46]. On the other hand, some polynomial time solvable classes which are apparently so trivial that they have been all but neglected in the literature, and are not vulnerable to cyclic structures, are common even out to $m/n = 1$. See Section 6.6 for details.

Before leaving this section we mention the recent important non-rigorous contributions of the statistical physics community leading to a better understanding of the nature of hard problems and proposals for algorithms that can deal with them. A discussion of this topic is left to Section 6.8.

6 The Probabilistic Analysis of Algorithms

In this section we present examples of analysis using the tools of Section 3. We choose examples which represent, without complication, ideas that have led to progressively better results. This means, in some cases, the results obtained here are not the "best" known.

6.1 Myopic algorithms for satisfiable expressions

For densities where random expressions are satisfiable with high probability the behavior of some DPLL variants can be understood adequately by means of performance results on corresponding straight line algorithms. A first wave of many results were obtained for myopic algorithms of the form of Algorithm SCA in Figure 5, known as the Shortest Clause Algorithm because each variable assignment is chosen to satisfy a non-satisfied clause of least width. The study of these algorithms is motivated by the observation that if a straight line algorithm is to succeed, then it cannot allow an accumulation of unit clauses since too many unit clauses increases the likelihood of a pair of unit clauses which have opposite polarity and that would terminate the algorithm without success. Intuitively, eliminating shortest clauses first should then be a priority. Probabilistic analysis confirms this intuition to some extent: results for the class of shortest clause first algorithms are quite good compared to other heuristics, for example the pure literal heuristic alone. In addition, the analysis is based on *clause-flows* and illuminates the algorithmic mechanics that cause successful behavior. The remainder of this section is a high-level presentation of the analysis of several straight line algorithms. The main intention is to focus on motivation and intuition with a minimum of details.

A clause-flow model can be used to analyze the mechanics of many straight line algorithms. In particular, in this section we concentrate on the family identified as Algorithm SCA in Figure 5 and known as the *shortest clause algorithms*. Let ϕ be a random (n, m, k)-CNF expression. The shortest clause algorithms choose an unassigned literal on each iteration. Such a choice causes some clauses of ϕ to disappear because they have become satisfied and some clauses to have a literal removed because they have become falsified by the implicit assignment. Thus, at the start of the jth iteration of SCA some non-satisfied, remaining clauses have k literals, some $k-1$ literals and so on. Define $C_i^{\phi}(j)$ to be the set of clauses of ϕ that have exactly i literals at the start of the jth iteration (henceforth, the superscript will be dropped for simplicity). Define $w_i(j)$ to be the number of i-literal clauses added to $C_i(j)$ as a result of choosing a literal on the jth iteration. That is, $w_i(j)$ is the flow of clauses into $C_i(j)$ due to picking a literal on the jth iteration. Define $z_i(j)$ to be the number of clauses eliminated from $C_i(j)$ (satisfied) as a result of choosing a literal on the jth iteration. That is, $z_i(j)$ is the flow out of $C_i(j)$ due to picking a literal on the jth iteration. Let $m_i(j) = |C_i(j)|$. Figure 8 shows these clause sets, represented by ovals, and clause flows, represented by arcs with arrows indicating flow direction.

The success of a shortest clause algorithm depends critically on what is happening to $w_0(j)$. If $w_0(j) > 0$ for any j, any shortest clause algorithm stops and gives up because some clause has just had all its literals falsified by the current partial truth assignment. In turn, $w_0(j)$ can be controlled by keeping complememtary pairs of clauses out of $C_1(j)$, for all j, since, if such a pair exists in $C_1(j)$ for some $j = j'$, then eventually $w_0(j) > 0$ for some $j = j^* > j'$. Complementary pairs may be kept out of $C_1(j)$, for all j, by preventing a significant accumulation of unit clauses over time since such an accumulation tends to raise the probability that a complementary pair exists. Choosing a unit clause literal first, if one exists, does this by acting like a "pump" that attempts to immediately discharge all clauses which flow into $C_1(j)$. Unfortunately, it is not usually the case that more than one unit clause can be discharged at a time. Therefore, by choosing unit clause literals first one is unable to prevent an accumulation in $C_1(j)$ when $w_1(j) > 1$ over a significant range of j.

An important approach to the analysis of shortest clause algorithms is to model the clause flows and accumulations as a system of differential equations and determine under what conditions $max_j\{w_1(j)\} = 1$. Those conditions mark the boundary of good probabilistic performance. In what follows we try this idea for Algorithm SCA given in Figure 5 with $s = 1$. This algorithm is called in the literature UC for Unit Clause.

Whether the flows can be modeled according to the approach stated above depends on two things: 1) the clauses in $\mathcal{C}_i(j)$, for any i and j, should be statistically independent and uniformly distributed; 2) conditions whereby markovian

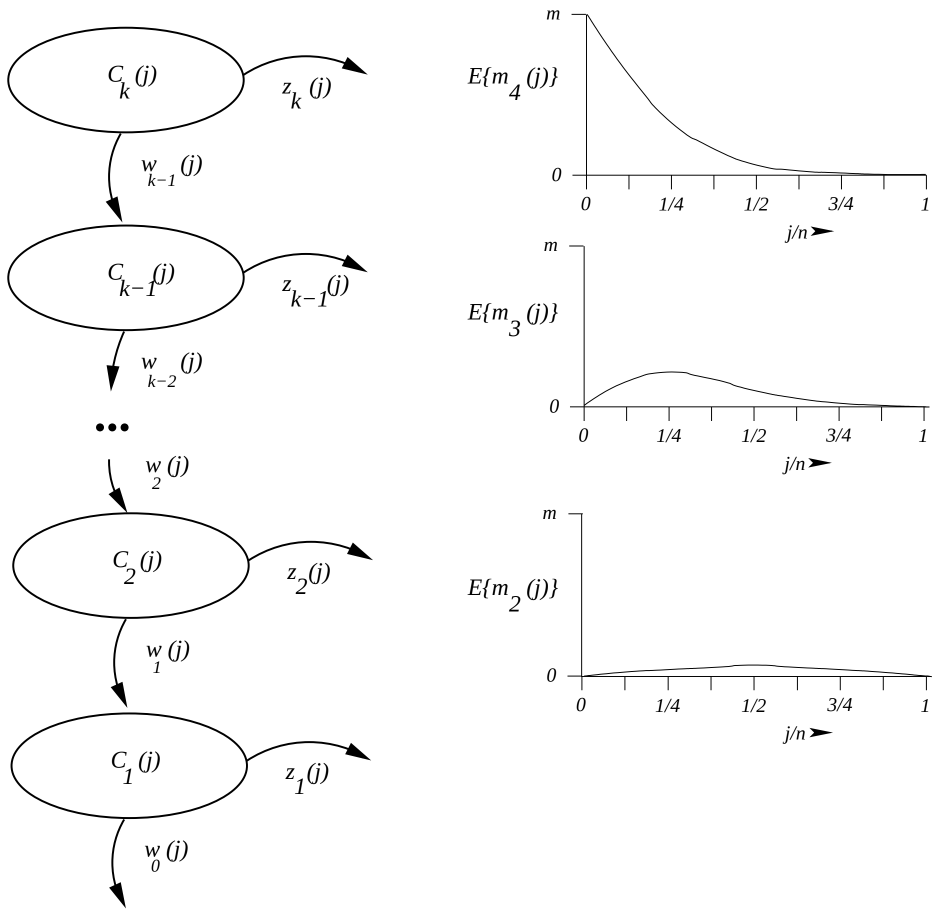

Figure 8: Clause sets and flows for a linear algorithm. On the left is a schematic representation. On the right are plots of the expected number of clauses in $\mathcal{C}_i(j)$ versus j/n for the case $k = 4$ with $i = 2, 3, 4$.

processes may be modeled as differential equations should be satisfied. In the first case, clauses entering $\mathcal{C}_i(j)$ are independent of clauses existing in $\mathcal{C}_i(j)$ and of each other, and are uniformly distributed since they were so in $\mathcal{C}_{i+1}(j-1)$ and the chosen literal is selected randomly from the appropriate set of free literals. Also, conditioned on the event that at least one unit clause leaves $\mathcal{C}_1(j)$ when $|\mathcal{C}_1(j)| > 0$, clauses leave $\mathcal{C}_i(j)$ independently as well. This establishes that Algorithm SCA with $s = 1$ is myopic (see Section 3.3). In the second case, even as n grows, the flows $w_i(j)$ and $z_i(j)$ are binomially distributed with means that are less than km/n which is bounded by a constant. This is enough to satisfy the conditions of Theorem 2 so there is no loss in modeling the discrete flows and accumulations by a system of differential equations.

With this out of the way we can write, for $1 \le i \le k$, $0 < j < n$,

$$m_i(j+1) = m_i(j) + w_i(j) - z_i(j).$$

Taking expectations gives

$$E\{m_i(j+1)\} = E\{m_i(j)\} + E\{w_i(j)\} - E\{z_i(j)\}$$

which can be written

$$E\{m_i(j+1)\} - E\{m_i(j)\} \quad = \quad E\{w_i(j)\} - E\{z_i(j)\}. \tag{7}$$

But, due to the statistical independence of clauses at every j, and the fact that ϕ is originally constructed from a set of n variables, we have, for all $2 \le i \le k$, $0 < j < n$,

$$\begin{aligned} E\{z_i(j)\} &= E\{E\{z_i(j)|m_i(j)\}\} \\ &= E\left\{\frac{i * m_i(j)}{n-j}\right\} = \frac{i * E\{m_i(j)\}}{n-j}. \end{aligned}$$

Also, for all $1 \le i < k$, $0 < j < n$,

$$\begin{aligned} E\{w_i(j)\} &= E\{E\{w_i(j)|m_{i+1}(j)\}\} \\ &= E\left\{\frac{(i+1) * m_{i+1}(j)}{2(n-j)}\right\} = \frac{(i+1) * E\{m_{i+1}(j)\}}{2(n-j)} \quad \text{and} \\ E\{w_k(j)\} &= 0. \end{aligned}$$

After substituting these results into Equation 7 and verifying conditions in Theorem 2, we can write the corresponding difference equations as follows: for $2 \le i < k$,

$$\begin{aligned} E\{m_i(j+1) - m_i(j)\} &= \frac{(i+1) * E\{m_{i+1}(j)\}}{2(n-j)} - \frac{i * E\{m_i(j)\}}{n-j} \quad \text{and} \\ E\{m_k(j)\} &= -\frac{k * E\{m_k(j)\}}{n-j}. \end{aligned}$$

The corresponding differential equations are

$$\begin{aligned}\frac{d\bar{m}_i(x)}{dx} &= \frac{(i+1)*\bar{m}_{i+1}(x)}{2(1-x)n} - \frac{i*\bar{m}_i(x)}{(1-x)n} \quad \text{and} \\ \frac{d\bar{m}_k(x)}{dx} &= -\frac{k*\bar{m}_k(x)}{(1-x)n}\end{aligned}$$

where $\bar{m}_i(x)$ is $z_i(x)$ of Theorem 2.

Boundary conditions, assuming m clauses of k literals in ϕ initially, are $\bar{m}_k(0) = m/n$ and $\bar{m}_i(0) = 0$ for all $1 \leq i < k$. The solution to the equations with these boundary conditions is, for all $2 \leq i \leq k$

$$\bar{m}_i(x) = \frac{1}{2^{k-i}}\binom{k}{i}(1-x)^i\,(x)^{k-i}\,m/n.$$

Thus,

$$E\{m_i(j)\} = \frac{1}{2^{k-i}}\binom{k}{i}(1-j/n)^i\,(j/n)^{k-i}\,m.$$

Plots of these functions for $k = 4$ are given in Figure 8.

The important flow is given by

$$E\{w_1(j)\} \;=\; \frac{E\{m_2(j)\}}{n-j} = \frac{1}{2^{k-2}}\binom{k}{2}(1-j/n)(j/n)^{k-2}(m/n).$$

By Theorem 3 and since $|E\{m_2(j)\} - m_2(j)| < \beta$, for any $0 < \beta$, almost always, from Theorem 2(b), Algorithm UC succeeds with probability bounded from below by a constant if

$$E\{w_1(j)\} = \frac{E\{m_2(j)\}}{n-j} < \frac{m_2(j)+\beta}{n-j} = \frac{m_2(j)}{n-j} + o(1) < 1 - \epsilon + o(1)$$

for any $0 < \epsilon$. That is, the algorithm succeeds as long as the average rate of production of unit clauses is no greater than 1 per iteration at any iteration.

Taking the derivative with respect to j and setting to 0 yields a maximum for $E\{w_1(j)\}$ at $j = j^* = \frac{k-2}{k-1}n$. The value of $E\{w_1(j^*)\}$ is less than 1 if

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

We can now conclude the following.

Theorem 13 *Algorithm* UC *determines a given random* (n, m, k)*-CNF expression has a model with probability bounded from below by a constant if*

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

□

Observe that for $k = 3$, UC succeeds with bounded probability when $m/n < 2.666$. By Theorem 6 almost all random $(n, m, 3)$-CNF expressions have at least one model if $m/n < 2.666$.

A feature of the flow analysis outlined above is it reveals mechanisms that suggest other, improved heuristics. For example, since increasing z flows decreases w flows, the following adjustment to Algorithm UC is suggested: if there are no unit clauses, choose a literal l randomly from ϕ, then compare the number of occurrences of l with the number of occurrences of $\bar{l}$ and choose the literal that occurs most often (this tends to increase z flows at the expense of w flows). This isn't quite good enough, however, since $C_i(j)$ clauses are approximately twice as influential as $C_{i+1}(j)$ clauses. This is because, roughly, one clause is accounted for in $z_i(j)$ for every two clauses in $z_{i+1}(j)$. Thus, it is better to compare *weights* of literals where the weight of a literal is given by:

$$\omega(l) = \sum_{c \in \phi : l \in c} 2^{-|c|}.$$

An analysis of such a heuristic is not known to us but the algorithm, called UCL, for 3-CNF expressions shown in Figure 9 using the original idea of counting clauses, comes fairly close.

Whereas in the case of Algorithm UC applied to random $(n, m, 3)$-CNF expressions a model can be found with bounded probability when $m/n < 2.66$, in the case of Algorithm UCL we have the following

Theorem 14 *Algorithm* UCL *determines a given random* $(n, m, 3)$*-CNF expression has a model with probability bounded from below by a constant if* $m/n < 2.9$. $\square$

An improved analysis and improved bound to $m/n < 3.001$ is given in [1].

The reader may be curious about why the number of occurrences of literals in $C_2(j)$ was not taken into account in Algorithm UCL. Flow analysis tells us that it is unnecessary. Suppose that, in the case $C_1(j) = \emptyset$, the $j + 1$st literal is chosen on the number of times it occurs in both $C_3(j)$ and $C_2(j)$. Assume the most optimistic case: the literal appears in more clauses of both $C_3(j)$ and $C_2(j)$ than its complement (then the flow into $C_1(j + 1)$ is minimized since the number of two and three literal clauses removed due to the $j + 1$st chosen literal is maximized). Let $E\{w_1^*(j)\}$ denote the new average flow of clauses into $C_1(j)$. Then

$$E\{w_1^*(j)\} = E\{w_1(j)\} - h_1(j)(1 - E\{w_1^*(j)\})$$

where $h_1(j)$ is the extra number of clauses removed from the flow into $C_1(j)$ when the chosen literal is not a unit clause and $1 - E\{w_1^*(j)\}$ is the probability (to within $O(\frac{1}{n})$) that the chosen literal is not a unit clause. Therefore,

$$E\{w_1^*(j)\} = \frac{E\{w_1(j)\} - h_1(j)}{1 - h_1(j)}.$$

```
**Procedure** UCL($\phi$)
**Input**: a CNF expression $\phi$.
**Output**: either “unsatisfiable” or a model for $\phi$.

**begin**
    $M := \emptyset$;
    $j := 0$;
    $L := \{x, \bar{x} : \exists C \in \phi$ such that either $x \in C$ or $\bar{x} \in C\}$;
    repeat the following {
        if $C_1(j) \neq \emptyset$ then randomly choose literal $x \in C_1(j)$;
        else {
            choose literal $y$ randomly from $L$;
            if # occurrences of $\bar{y}$ in $C_3(j) >$ # occurrences of $y$ in $C_3(j)$ then
                $x := \bar{y}$;
            else
                $x := y$;
        }
        $L := L - \{x, \bar{x}\}$;
        remove from $\phi$ all clauses containing $x$;
        remove from $\phi$ all occurrences of $\bar{x}$;
        if $x$ is an uncomplemented literal then $M := M \cup \{x\}$.
        $j := j + 1$;
    } until $\phi = \emptyset$ or there are two complementary unit clauses in $\phi$;
    if $\phi = \emptyset$ then return $M$;
    else return(“cannot determine whether a model exists”);
**end**;
```

Figure 9: A heuristic which chooses variables randomly but assigns values based on the difference in the number of occurrences of complemented and uncomplemented literals.

Thus $E\{w_1^*(j)\} < 1$ is equivalent to $E\{w_1(j)\} < 1$ and no benefit is gained by considering the number of occurrences of the chosen literal in $C_2(j)$.

There is another improved heuristic suggested by flow analysis. If "pumping" clauses at the bottom level by means of the unit clause rule is effective, putting "pumps" at all levels should be more effective. This amounts to adopting the following strategy for literal selection which is a generalization of the unit clause rule called the smallest clause rule: choose a literal from a clause of smallest size. Thus, if there is at least one unit clause, choose from one of them; otherwise, if there is at least one 2-literal clause, choose a literal from a 2-literal clauses; otherwise, if there is at least one 3-literal clause, choose a literal from a 3-literal clause, and so on. This is Algorithm SCA, Figure 5 with $s = k$. We call this Algorithm GUC.

The effectiveness of "pumping" at all levels is revealed by a flow analysis applied to Algorithm **GUC**. The results, taken from [21], are

Theorem 15 *Algorithm* **GUC** *determines that a random (n, m, k)-CNF expression has a model with probability bounded from below by a constant when*

$$\frac{m}{n} < \frac{3.09 * 2^{k-2}}{k+1}\left(\frac{k-1}{k-2}\right)^{k-2} \quad \text{and} \quad 4 \le k \le 40.$$

□

Theorem 16 *Algorithm* **GUC** *determines a random (n, m, k)-CNF expression has a model with probability tending to 1 when*

$$\frac{m}{n} < \frac{1.845 * 2^{k-2}}{k+1}\left(\frac{k-1}{k-2}\right)^{k-2} - 1 \quad \text{and} \quad 4 \le k \le 40.$$

□

But it is not necessary to "pump" at all levels to get this kind of result. Consider Algorithm SCA with $s = 2$ and call it Algorithm SC. The following analysis sketch of Algorithm SC is based on results by Chvátal and Reed [23]. Let p_j denote the probability that a fixed input clause shrinks to two literals after exactly j iterations of Algorithm SC. Then $p_j m$ is the average flow into $C_2(j)$. The following simple expression for p_j can be obtained straightforwardly.

$$p_j = \frac{\binom{j-1}{k-3}\binom{n-j}{2}}{\binom{n}{k}} \frac{1}{2^{k-2}}$$

This can be bounded from above by setting j to the value that maximizes p_j (the maximum occurs when the ratio p_{j+1}/p_j crosses the value 1). We are interested in conditions that imply the bound for p_j is less than $1/m$ since that gives an average flow into $C_2(j)$ that is less than 1. Straightforward calculation reveals, for all j,

$$p_j < \frac{1-\epsilon}{m} \text{ for any fixed } \epsilon > 0 \text{ such that } \frac{m}{n} < \left(\frac{1-\epsilon}{1+\epsilon}\right)\frac{2^k}{8k}\left(\frac{k-1}{k-3}\right)^{k-2}\frac{k-1}{k-2}.$$

This yields the following

Theorem 17 *Algorithm* **SC** *determines a random (n, m, k)-CNF expression, $k \geq 3$, has a model with probability tending to 1 when*

$$\frac{m}{n} < \frac{2^k}{8k}\left(\frac{k-1}{k-3}\right)^{k-3}\frac{k-1}{k-2}.$$

□

The difference between the result of Theorem 17 and that of Theorem 16 is due to improved analysis facilitated by working with an easier algorithm.

By adding a limited amount of backtracking to GUC, Frieze and Suen produced an algorithm, called GUCB, for 3-CNF expressions that finds a model, with probability tending to 1, when $m/n < 3.003$ [48]. Probabilistic analysis of a backtracking algorithm given random (n, m, k)-CNF expressions ϕ can be exceedingly difficult because statistical dependences can easily show up when returning to subexpressions after a failure to locate a model. However, Frieze and Suen showed it is possible to carefully manage a limited amount of backtracking so that this does not happen. The following explains how the backtracking in GUCB is accomplished. The initial operation of GUCB is the same as that of GUC. Suppose GUCB has successfully completed t iterations and has chosen the sequence of literals $\{x_{\pi_1}, x_{\pi_2}, \ldots x_{\pi_t}\}$ and set them to 1. Suppose $|C_1(t')| = 0$, $t' < t$ and $|C_1(j)| > 0$ for all $t' < j \leq t$ so the last iteration that saw no unit clauses was iteration t'. Suppose further that choosing and setting literal $x_{\pi_{t+1}}$ to 1 results in the existence of complementary unit clauses in $C_1(t+1)$. Then GUCB backtracks by setting $x_{\pi_{t'}} = x_{\pi_{t'+1}} = \ldots = x_{\pi_t} = x_{\pi_{t+1}} = 0$. Corresponding adjustments are made to the clauses and literals of ϕ. That is, clauses now satisfied are removed, removed clauses now *not* satisfied are reinstated, literals now falsified are removed, and removed literals now *not* falsified and in non-satisfied clauses are reinstated. After backtracking, GUCB continues choosing and setting literals to 1 as before. The algorithm succeeds if all clauses are eliminated. The algorithm fails in two ways: 1) the resetting of literals from 1 to 0 results in a null clause; 2) a complementary pair of unit clauses is encountered before $|C_1|$ has become 0 after a backtrack. The analysis of GUCB is possible because the effect on the distribution of $C_i(j)$ is slight and because, with probability tending to 1, GUCB backtracks at most $\ln^5(n)$ times when $m/n < 3.003$.

Frieze and Suen also apply limited backtracking to Algorithm SC and call the resulting algorithm SCB. The result is the following:

Theorem 18 *For $k \geq 4$,* SCB *succeeds, with probability tending to 1, when $m/n < \eta_k 2^k/k$ where $\eta_4 \approx 1.3836$, $\eta_5 \approx 1.504$, and $\lim_{k\to\infty} \eta_k \approx 1.817$.* $\square$

This may be compared to the above result for GUC which, at $k = 40$, has a performance bound of $m/n < (1.2376)2^{40}/40$ (probability tending to 1), and the above result for SC which has a performance bound of $\lim_{k\to\infty} m/n < (0.9236)2^k/k$ (probability tending to 1).

There is a limit to the probabilistic performance of straight line, myopic algorithms. That is, there is an optimal policy for choosing a variable and value on any iteration, depending on the iteration. For $k = 3$, the optimal policy is given by the following:

Optimal Myopic Literal Selection Policy

If $m_1(j) = 0$ choose 2-literal clause $\{v, w\}$ or $\{v, \bar{w}\}$, or $\{\bar{v}, w\}$ or $\{\bar{v}, \bar{w}\}$ at random. Select v, $\bar{v}$, w or $\bar{w}$ at random and temporarily assign the value to the chosen variable which satisfies its clause. Then fix the assignment according to the following. Let $M_3(j)$ be the number of 3-literal clauses satisfied minus the number of literals falsified in 3-literal clauses. Let $M_2(j)$ be the number of 2-literal clauses satisfied minus the number of literals falsified in 2-literal clauses. Let $d_3(j) = m_3(j)/(n-j)$. Let $d_2(j) = m_2(j)/(n-j)$. Define $\Theta(j) = (1.5 \cdot d_3(j) - 2 \cdot d_2(j) + \gamma)/(1 - d_2(j))$ where γ is some constant. Reverse the assignment if $\Theta(j) > M_3(j)/M_2(j)$.

Theorem 19 ([3]) *Algorithm* SCA *with $s = 1$ and using the optimal myopic literal selection policy instead of choosing a literal randomly from L_2 succeeds in finding a model given random $(n, m, 3)$-CNF expressions as input with probability bounded from below by a constant when $m/n < 3.22$. This is the best possible literal selection policy for* SCA *on random $(n, m, 3)$-CNF expressions.* $\square$

According to the optimal myopic literal selection policy if there is a literal that will maximize both the number of 3-literal clauses and 2-literal clauses satisfied, one will be selected. Otherwise, a literal is selected as a compromize between reducing the number of 2-literal clauses immediately and increasing the chance of reducing the number of 2-literal clauses at some point in the future.

A better result is due to a literal selection policy which may actually look at 2 literals on an iteration instead of just 1. The result is:

Theorem 20 *There exists a myopic straight line algorithm which succeeds in finding a model given random $(n, m, 3)$-CNF expressions as input with probability bounded from below by a constant which $m/n < 3.26$. This is the best performance possible by any myopic straight line algorithm.* $\square$

6.2 Non-myopic algorithms for satisfiable expressions

In light of experimental evidence suggesting r_3 is approximately 4.25 and the results of Theorem 20 there appears to be an unfortunate gap between what is achievable by myopic algorithms and what we would hope is possible from straight line algorithms. However, the tools of Section 3.3 may still be applied to the analysis of non-myopic straight line algorithms, under some restrictions. For example, flow analysis is still possible if algorithmic operations include [65]:

1. Select uniformly at random a pure literal, assign it the value 1 and remove all satisfied clauses.
2. Select uniformly at random a literal occuring exactly once in the expression and its occurrence is in a 3-literal clause, assign it the value 0, remove it from the clause it appears, and remove all clauses containing its complementary literal (these are satisfied).
3. Select uniformly at random a literal occuring exactly once in the expression and its occurrence is in a 2-literal clause, assign it the value 0, remove it from the clause it appears, and remove all clauses containing its complementary literal (these are satisfied). Then apply the unit clause unit to exhaustion (until no unit clauses remain).

An example, which we call GPL, from [63] is shown in Figure 10.

Theorem 21 *Algorithm* GPL *succeeds in finding a model given random* $(n, m, 3)$*-CNF expressions as input with probability bounded from below by a constant when* $m/n < 3.42$. □

Similar algorithms have been reported to have good performance out to $m/n < 3.52$ [55, 64].

Other non-myopic straight line algorithms have shown even better performance on random (n, m, k)-CNF inputs. For example, the algorithm of Figure 11 seems to do well for $m/n < 3.6$ on random $(n, m, 3)$-CNF inputs. The algorithm is designed to iteratively select a variable and value which maximizes the expected number of models possessed by what is left of the given expression after satisfied clauses and falsified literals are removed, assuming somehow the clauses of the new expression are statistically independent. Of course, that is not the case. But it is conceivable that the fundamental idea of maximizing expected number of models or maximizing the probability a model exists can be enhanced to provide algorithms of greater performance. An analysis of this and other algorithms like it has not yet been reported.

```
**Procedure** GPL($\phi$)
**Input**: a CNF expression $\phi$.
**Output**: either "gives up" or returns a model for $\phi$.

**begin**
    $M := \emptyset$;
    repeat the following {
        if $\phi = \emptyset$ then return $M$;
        if $\emptyset \in \phi$ then return("cannot determine whether model exists");
        let $l$ be a literal in $\phi$ occurring at least as frequently as any other;
        if $l$ is an uncomplemented literal then $M := M \cup \{l\}$;
        $\phi := \{C - \{\bar{l}\} : C \in \phi, l \notin C\}$
        repeat the following until no unit clauses exist in $\phi$ {
            let $\{l\}$ be a unit clause in $\phi$;
            if $l$ is an uncomplemented literal then $M := M \cup \{l\}$;
            $\phi := \{C - \{\bar{l}\} : C \in \phi, l \notin C\}$;
        }
    }
**end**;
```

Figure 10: A non-myopic straight line algorithm.

6.3 Lower bounds on the satisfiability threshold

The results of the previous two sections may be used to obtain lower bounds for the random (n, m, k)-CNF threshold. Thus, from algorithmic analysis we know $r_3 > 3.42$. Unfortunately, the best that can be said for the general threshold, based on algorithmic analysis, is $r_k > c \cdot 2^k/k$, c a constant. But recently the second moment method has been applied ([5]) to show that

$$r_k \geq 2^k \ln(2) - (k+1)\ln 2/2 - 1 - \delta_k$$

where $\delta_k \to 0$ for increasing k. These results improve and are built upon those of [4]. We sketch the motivation leading to these results in this section.

The crucial question is which random variable X can the second moment method be applied to. As stated in Section 3.2, if X is the number of models, the bound provided by the second moment method is inadequate. This is because for many random (n, m, k)-CNF expressions the number of models is far from the mean number causing the variance of X to be too high. More specifically, $Pr(z|w)$ (see Lemma 1, Page 7) is too high. This can be seen as follows. Let z and w be assignments to variables in random (n, m, k)-CNF expression ϕ agreeing in αn variables. Then

$$Pr(z \text{ is a model for } \phi | w \text{ is a model for } \phi) = \left(1 - \frac{1-\alpha^k}{2^k - 1}\right)^m.$$

```
Procedure MPS($\phi$)
Input: a CNF expression $\phi$.
Output: either "gives up" or returns a model for $\phi$.

begin
    $M := \emptyset$;
    repeat the following {
        if $\phi = \emptyset$ then return $M$;
        if $\emptyset \in \phi$ then return("cannot determine whether model exists");
        repeat the following until no unit clauses exist in $\phi$ {
            let $\{l\}$ be a unit clause in $\phi$;
            if $l$ is an uncomplemented literal then $M := M \cup \{l\}$;
            $\phi := \{C - \{\bar{l}\} : C \in \phi, l \notin C\}$;
        }
        for all unassigned literals $l$ do the following {
            $\phi_l := \{C - \{\bar{l}\} : C \in \phi, l \notin C\}$;
            define $w(l)$, the weight of literal $l$, to be $\prod_{C \in \phi_l}(1 - 2^{-|C|})$;
        }
        choose $l$ so that $w(l)$ is maximum over all unassigned literals;
        if $l$ is an uncomplemented literal then $M := M \cup \{l\}$
        $\phi := \{C - \{\bar{l}\} : C \in \phi, l \notin C\}$;
    }
end;
```

Figure 11: Another non-myopic straight line algorithm.

Therefore, to apply the second moment method, we will need to control

$$\sum_{0 \le \alpha \le 1} \binom{n}{\alpha n} \left(1 - \frac{1 - \alpha^k}{2^k - 1}\right)^m.$$

Observe the maximum of this sum does not occur at $\alpha = 1/2$ and is not $o(\mu)$ as needed (recall $\mu = 2^n(1 - 2^{-k})^m$).

The fix is to use a bound for a closely related problem for which the maximum of the sum occurs at $\alpha = 1/2$. The problem is called Not-All-Equal k-CNF (which we denote by NAE$_k$). Given random (n, m, k)-CNF expression ϕ, NAE$_k$ is the problem of finding a model for ϕ such that every clause has at least one literal falsified as well as at least one satisfied. Since a NAE$_k$ model for ϕ is also a model for ϕ

$$Pr(\exists \text{ model for } \phi) > Pr(\exists \text{ NAE}_k \text{ model for } \phi).$$

But

$$Pr(z \text{ is a NAE}_k\ model\ for\ \phi | w \text{ is a NAE}_k\ model\ for\ \phi) =$$

$$\left(1 - \frac{1-\alpha^k - (1-\alpha)^k}{2^{k-1}-1}\right)^m$$

and

$$\sum_{0\le\alpha\le 1} \binom{n}{\alpha n} \left(1 - \frac{1-\alpha^k - (1-\alpha)^k}{2^{k-1}-1}\right)^m \approx \frac{2^n(1-2^{-k+1})^m}{\sqrt{n}} = o(\mu)$$

because all the significant contributions to the sum occur near the maximum of both terms in the summand which is at $\alpha = 1/2$ and μ for NAE$_k$ on random (n, m, k)-CNF inputs is $2^n(1 - 2^{-k+1})^m$.

6.4 Algorithms for verifying unsatisfiability: resolution

The algorithms discussed above are useless for verifying the unsatisfiability of an expression where it is required to prove that no truth assignment is a satisfying one. Resolution is one popular method capable of doing this and can be extremely effective when certain clause patterns are present, with high probability. Recall from Page 26 that the resolvent of two clauses c_1 and c_2 such that there is exactly one variable v that occurs as a complemented literal in one clause and as an uncomplemented literal in the other, denoted by $\mathcal{R}^{c_1}_{c_2}$, is a clause which contains all the literals of c_1 and all the literals of c_2 except for v and $\bar{v}$. That is, $\mathcal{R}^{c_1}_{c_2} = \{l : l \in c_1 \cup c_2 \setminus \{v, \bar{v}\}\}$. If resolvents are generated until one is the empty set, then the original expression is unsatisfiable.

There are many resolution-based algorithms: that is, algorithms using the generation of resolvents to determine whether a given expression is satisfiable. They differ in the restrictions that are applied to help improve the total number of resolvents generated. The following restricted resolution algorithm, called RW for Restricted Width, has polynomial time complexity because resolvents are restricted to be no greater than k in width:

```
Procedure RW($\phi$)
Input: a CNF expression $\phi$.
Output: either "unsatisfiable" or "gives up".

begin
    repeat the following {
        if there are clauses $C_1, C_2 \in \phi$ s.t. $\mathcal{R}_{C_2}^{C_1} \notin \phi$ and $|\mathcal{R}_{C_2}^{C_1}| \le k$ then
            $\phi := \phi \cup \{\mathcal{R}_{C_2}^{C_1}\}$;
        else
            break out of the loop;
    }
    if $\emptyset \in \phi$ then return ("unsatisfiable");
    else return ("cannot determine whether $\phi$ is unsatisfiable");
end;
```

Clearly, Algorithm RW cannot be mistaken if it outputs unsatisfiable.

It can be shown that Algorithm RW is effective on random (n, m, k)-CNF expressions up to a point. Particular patterns of clauses that RW can exploit to produce a null resolvent are developed as follows:

Let $\mathcal{S}_{i,j}$ denote the family of clause sets defined as follows over a set of variables $X = \{x_{11}, x_{21}, \ldots, x_{12}, x_{22}, \ldots\}$:

$$\mathcal{S}_{i,j} = \begin{cases} \{\{x_{i1}\}\{\bar{x}_{i1}\}\} & if\ j = 1; \\ \{P \cup \{x_{ij}\} : P \in S_{2i-1,j-1}\} \cup \{P \cup \{\bar{x}_{ij}\} : P \in S_{2i,j-1}\} & if\ j > 1. \end{cases}$$

Thus,

$$\begin{aligned} \mathcal{S}_{1,1} &= \{\{x_{11}\}, \{\bar{x}_{11}\}\} \\ \mathcal{S}_{1,2} &= \{\{x_{12}, x_{11}\}, \{x_{12}, \bar{x}_{11}\}, \{\bar{x}_{12}, x_{21}\}, \{\bar{x}_{12}, \bar{x}_{21}\}\} \\ \mathcal{S}_{1,3} &= \{\{x_{13}, x_{12}, x_{11}\}, \{x_{13}, x_{12}, \bar{x}_{11}\}, \{x_{13}, \bar{x}_{12}, x_{21}\}, \{x_{13}, \bar{x}_{12}, \bar{x}_{21}\}, \\ &\quad \{\bar{x}_{13}, x_{22}, x_{31}\}, \{\bar{x}_{13}, x_{22}, \bar{x}_{31}\}, \{\bar{x}_{13}, \bar{x}_{22}, x_{41}\}, \{\bar{x}_{13}, \bar{x}_{22}, \bar{x}_{41}\}\} \end{aligned}$$

Over variable set $X \cup \{y_0, y_1, y_2, \ldots\}$ define

$$\mathcal{R}_{k,r} = \left\{ \begin{array}{l} \{P \cup \{y_0, \bar{y}_1\} : P \in \mathcal{S}_{1,k-2}\} \cup \\ \left\{\cup_{i=1}^{r-2}\{P \cup \{y_i, \bar{y}_{i+1}\} : P \in \mathcal{S}_{i+1,k-2}\}\right\} \cup \\ \{P \cup \{y_{r-1}, y_0\} : P \in \mathcal{S}_{r,k-2}\} \cup \\ \{P \cup \{\bar{y}_0, \bar{y}_r\} : P \in \mathcal{S}_{r+1,k-2}\} \cup \\ \left\{\cup_{i=r+1}^{2r-2}\{P \cup \{y_{i-1}, \bar{y}_i\} : P \in \mathcal{S}_{i+1,k-2}\}\right\} \cup \\ \{P \cup \{y_{2r-2}, \bar{y}_0\} : P \in \mathcal{S}_{2r,k-2}\} \end{array} \right\}$$

For example,

$$\mathcal{R}_{3,3} = \left\{ \begin{array}{l} \{y_0, \bar{y}_1, x_{11}\}, \{y_0, \bar{y}_1, \bar{x}_{11}\}, \{y_1, \bar{y}_2, x_{21}\}, \{y_1, \bar{y}_2, \bar{x}_{21}\}, \\ \{y_2, y_0, x_{31}\}, \{y_2, y_0, \bar{x}_{31}\}, \{\bar{y}_0, \bar{y}_3, x_{41}\}, \{\bar{y}_0, \bar{y}_3, \bar{x}_{41}\}, \\ \{y_3, \bar{y}_4, x_{51}\}, \{y_3, \bar{y}_4, \bar{x}_{51}\}, \{y_4, \bar{y}_0, x_{61}\}, \{y_4, \bar{y}_0, \bar{x}_{61}\} \end{array} \right\}$$

Algorithm RW, applied to any k-CNF expression that contains a subset of clauses which can be mapped to $\mathcal{R}_{k,r}$ $(r \geq 2)$ after renaming of variables, always results in a certificate of unsatisfiability for that expression. Therefore, if random (n, m, k)-CNF expression ϕ has such a subset with high probability, then RW will conclude unsatisfiability with high probability.

It is straightforward to obtain the probability that there is a subset of clauses that maps to $\mathcal{R}_{k,r}$ for some r. Expression $\mathcal{R}_{k,r}$ contains $r2^{k-1}$ clauses and $r2^{k-1} - 1$ distinct variables, and is *minimally unsatisfiable*: that is, it is unsatisfiable but removal of any clause produces a satisfiable set. This property is important to the analysis since a minimally unsatisfiable expression can be padded with additional clauses to get other unsatisfiable clause patterns with a much higher ratio of variables to clauses, but the probability that such a padded pattern exists is not greater than the probability that one of its base minimally unsatisfiable sets exist. The existence probability crosses from tending to 0 to tending to 1 at a particular ratio of m/n. This can be estimated by finding conditions that set the average number of minimally unsatisfiable sets to ∞ in the limit. The probability that a particular subset of $r2^{k-1}$ clauses matches the pattern $\mathcal{R}_{k,r}$ for a particular choice of variables is $(r2^{k-1})!/\left(2^k\binom{n}{k}\right)^{r2^{k-1}}$. There are $\binom{n}{r2^{k-1}-1}$ ways to choose the variables and each of $2^{r2^{k-1}-1}$ ways to complement the chosen variables presents a pattern that RW can handle. Futhermore, any of the $(r2^{k-1} - 1)!$ permutations of those variables also presents a solvable pattern. Finally, there are $\binom{m}{r2^{k-1}}$ ways to choose clauses. Therefore, the average number of $\mathcal{R}_{k,r}$ patterns in ϕ is the product of the above terms which is about $(k!m/(2^{k-1}n^{(k-1)+1/r2^{k-1}}))^{r2^{k-1}}$. This tends to ∞ when $m/n^{(k-1)} > (n\omega(n))^{1/r2^{k-1}}$ where $\omega(n)$ is any slowly growing function of n. Setting $r = \lfloor \ln^{1+\epsilon}(n) \rfloor$, where ϵ is a small constant is sufficient to give the following result (in this case patterns are small enough for a second moment analysis to carry through).

Theorem 22 *Algorithm* RW *succeeds, with probability tending to 1, on random* (n, m, k)*-CNF expressions if* $m/n > n^{k-2}2^{k-1}/k!$. $\Box$

This analysis shows the importance of the ratio of the number of variables to the number of clauses in a minimally unsatisfiable expression. A higher ratio means a lower cutoff for m/n^{k-1}. We remark that the first results of the above nature appearing in print, as far as we know, are due to Xudong Fu [49]. Fu used a minimally unsatisfiable pattern which he called a *Flower* to achieve the above result. But Flowers have $2^{k-1}(r+k-1)$ clauses and $2^{k-1}r+k-1$ variables. Since the difference between clauses and variables in $R_{k,r}$ is 1, whereas the difference is $(2^{k-1}-1)(k-1)$ in the case of Flowers, we chose to present $R_{k,r}$ instead.

Improving on the above result is hard. An obvious initial attempt consists of finding minimally unsatisfiable clause patterns where the ratio of variables to clauses is higher than 1 and hopefully close to k (recall, a ratio of 1 is roughly what is achieved above). But, the following result, which can be proved by induction on the number of variables in a minimally unsatisfiable set and first appeared in [9] (see also [11] and [68]), is discouraging:

Theorem 23 *If ϕ is a minimally unsatisfiable CNF expression with m clauses, then ϕ has less than m variables.* $\square$

Notice that Theorem 23 does not say anything about the number of literals in a clause: it could be fixed at any k or different for each clause and the result still holds.

Theorems 22 and 23 do not say that resolution fails to prove unsatisfiability in polynomial time with high probability when $m/n^{(k-1)} \to 0$. These theorems suggest only that looking for clause patterns which cause an algorithm such as RW to succeed often for such values of m/n is likely not to bear fruit. However, a deeper and illuminating argument shows that resolution can't be successful, with high probability, for a wide range of m/n values.

A number of papers consider the probabilistic performance of resolution for random unsatisfiable (n, m, k)-CNF expressions (for example, [11, 12, 24, 49]). The following two results are noteworthy.

Theorem 24 *For $k > 3$, an unsatisfiable random (n, m, k)-CNF expression has only exponential size resolution proofs, with probability tending to 1, if $m/n^{(k+2)/4-\epsilon} < 1$ for any $\epsilon > 0$. A random $(n, m, 3)$-CNF expression has only exponential size resolution proofs, with probability tending to 1, if $m/n^{6/5-\epsilon} < 1$, for any $\epsilon > 0$.* $\square$

Theorem 25 *There exists a DPLL variant which verifies the unsatisfiability of a random (n, m, k)-CNF expression in polynomial time with probability tending to 1 when $m/n > (n/\log(n))^{k-2}$.* $\square$

The fact that no resolution result better than that of Theorem 25 has been found is surprising: it seems hard to imagine that RW is so close to what is best possible for resolution. But a result of [12] shows that at least some natural DPLL variants cannot do much better. These pessimistic results motivate the results of the next section.

6.5 Algorithms for unsatisfiability: a spectral analysis

The pessimistic results obtained for verifying unsatisfiability via resolution algorithms has motivated the search for alternatives. In this section a simple one is presented and an analysis outlined. A decision is made based on counting variables which must be set to 1 or 0 if a model exists.

The rough idea is as follows. Given a random (n, m, k)-CNF expression ϕ, throw out all clauses except those containing all uncomplemented literals and those containing all complemented literals. Let n_+ be the minimum number of variables that must be set to 1 to satisfy the uncomplemented clauses and let n_- be the number of varibles that must be set to 0 to satisfy the complemented clauses. If $n_+ + n_- > n$ then some variable must be set to both 1 and 0 if ϕ has a model. But this is impossible so ϕ cannot have a model. Hence, this simple counting argument provides a means to verify unsatisfiability. The only problem is that finding n_+ and n_- is NP-complete. However, it may be sufficient merely to approximate n_+ and n_- closely enough. The following explains how based on work reported in [52] (warning: some simplifications have been made). We start with this theorem which is obvious from the above discussion:

Theorem 26 *If k-CNF expression ϕ has a model then there is a subset V' of $n/2$ variables such that either ϕ has no all uncomplemented clause or no all complemented clause taken strictly from V'.* $\square$

For purposes of discussion fix $k = 4$. Construct two graphs G_+ and G_-, corresponding to the uncomplemented and complemented clause sets of ϕ, respectively. Both G_+ and G_- have $\binom{n}{2}$ vertices and each vertex of G_+ and G_- is uniquely labeled by a pair of variables. An edge between two vertices of G_+ exists if and only if the uncomplemented clause set of ϕ contains a clause consisting of all variables labeling both its endpoints. An edge between two vertices of G_- exists if and only if the all complemented clause set of ϕ contains a clause consisting of all variables labeling both its endpoints. It follows from Theorem 26 that

Theorem 27 *If ϕ has a model, either G_+ or G_- has an independent set of size greater than $\binom{n/2}{2} \approx n^2/8$.* $\square$

The following theorem connects independent sets with eigenvalues.

Theorem 28 ([69]) *Let $G(V, E)$ be an undirected graph with vertex set $V = \{1, ..., n'\}$. Let $0 \le p \le 1$. Define $n' \times n'$ matrix $A_{G,p}$ such that $A_{G,p}(i, j) = 1$ if edge $\langle i, j\rangle \notin E$ and $A_{G,p}(i, j) = 1 - 1/p$ otherwise ($A_{G,p}(i, i) = 1$). Let $\lambda_1(A_{G,p})$ be the largest eigenvalue of $A_{G,p}$. Let $\alpha(G)$ be the size of the largest independent set of $G(V, E)$. For any possible p, $\lambda_1(A_{G,p}) \ge \alpha(G)$.* $\square$

The problem of determining the eigenvalues of $A_{G,p}$ can be handled by the following

Theorem 29 (for example, [79]) *The eigenvalues of $A_{G,p}$ can be computed with relative error less than 2^{-b} in time $O((n')^3 + (n' \log^2(n')) \log(b))$.* $\square$

The G of matrix $A_{G,p}$ will be G_+ once and G_- once. The p parameter is set to represent the probability that a pair of vertices of $A_{G,p}$ is connected by an edge. That is,[11] the number of clauses represented in G is about $m' = m/16$ and $p = m'\binom{n'}{2} = (m/16)\binom{n^2}{2}$. A theorem such as the following can then be proved.

Theorem 30 (adapted from [52]) *Let $m' = (\ln^7(n')/2)n'$ and $p = m'\binom{n'}{2} = \ln^7(n')/(n'-1)$. With high probability,*

$$\lambda_1(A_{G,p}) \leq 2n'(1/(\ln(n'))^{7/2})(1 + o(1)).$$

$\square$

Theorems 28 and 30 give an adequate bound on the size of the independent sets of G_+ and G_-. These results are used as follows. Suppose $m = 16(\ln(n^2))^7 n^2 = 16(2^7)(\ln^7(n))n^2$. Since $n' = n^2$ and $m' = m/16$, $A_{G,p}$ has about $2^7(\ln^7(\sqrt{n'}))n' = (\ln^7(n'))n'$ edges chosen with probability $p = \ln^7(n')/(n'-1)$. From Theorems 28 and 30 $\alpha(G_+) \leq n^2/16$ and $\alpha(G_-) \leq n^2/16$ with high probability. Hence, with high probability the maximum independent set of G_+ and of G_- is less than that required in Theorem 27 and finding the eigenvalues is enough to prove it in polynomial time. The general result is

Theorem 31 *Random (n, m, k)-CNF expressions can be certified as unsatisfiable in polynomial time with high probability if $m > n^{k/2+o(1)}$, $k \geq 4$, and if $m > n^{3/2+\epsilon}$, ϵ an arbitrarily small constant, $k = 3$.* $\square$

6.6 Polynomial time solvable classes.

Expressions that are members of certain polynomial time solvable classes, such as those defined in Section 4, are not generated frequently enough, for interesting densities m/n, to assist in determining whether random (n, m, k)-CNF expressions have models. This is unlike the case for random (n, m, p)-CNF expressions (Definition 11, Page 19 and following). We illustrate with a few examples. Whenever we use ϕ below it is to be understood that ϕ is a random (n, m, k)-CNF expression.

First consider the frequency with which random (n, m, k)-CNF expressions are Horn or hidden Horn. The probability that a clause is Horn is $(k + 1)/2^k$. Therefore, the probability that ϕ is Horn is $((k + 1)/2^k)^m$ which tends to 0 for any fixed k. For a hidden Horn expression, regardless of switch set, there are

[11]The factor of 1/16 is due to G_+ and G_- representing, with high probability, 1/16 of the m input clauses in the case that $k = 4$.

only $k+1$ out of 2^k ways (negation patterns) that a random clause can become Horn. Therefore, the expected number of successful switch sets is $2^n((k+1)/2^k)^m$ which tends to 0 if $m/n > 1/(k-\log_2(k+1))$. Thus, ϕ is not hidden Horn, with probability tending to 1, if $m/n > 1/(k-\log_2(k+1))$. Even when $k = 3$, this is $m/n > 1$. But this bound can be improved considerably by finding complex structures that imply an expression cannot be hidden Horn. Such a structure is presented next.

The following result for q-Horn expressions is taken from [46]. For $p = \lfloor \ln(n) \rfloor \geq 4$, call a set of p clauses a *c-cycle* if all but two literals can be removed from each of $p-2$ clauses, all but three literals can be removed from two clauses, the variables can be renamed, and the clauses can be reordered in the following sequence

$$\{v_1, \bar{v}_2\}, \{v_2, \bar{v}_3\} \ldots \{v_i, \bar{v}_{i+1}, v_{p+1}\} \ldots \{v_j, \bar{v}_{j+1}, \bar{v}_{p+1}\} \ldots \{v_p, \bar{v}_1\} \tag{8}$$

where $v_i \neq v_j$ if $i \neq j$. We use the term "cycle" to signify the existence of cyclic paths through clauses which share a variable: that is, by jumping from one clause to another clause only if the two clauses share a variable, one may eventually return to the starting clause. Given a c-cycle $\mathcal{C} \subset \phi$, if no two literals removed from $\mathcal{C}$ are the same or complementary, then $\mathcal{C}$ is called a *q-blocked c-cycle*.

If ϕ has a q-blocked c-cycle then it is not q-Horn. Let a q-blocked c-cycle in ϕ be represented as above. Develop inequalities (4) and (5) for ϕ. After rearranging terms in each, a subset of these inequalities is as follows

$$\begin{aligned}
&\alpha_1 \leq Z - 1 + \alpha_2 - \ldots && (9)\\
&\ldots\\
&\alpha_i \leq Z - 1 + \alpha_{i+1} - \alpha_{p+1} - \ldots\\
&\ldots\\
&\alpha_j \leq Z - 1 + \alpha_{j+1} - (1 - \alpha_{p+1}) - \ldots\\
&\ldots\\
&\alpha_p \leq Z - 1 + \alpha_1 - \ldots. && (10)
\end{aligned}$$

From inequalities (9) to (10) we deduce

$$\alpha_1 \leq pZ - p + \alpha_1 - (1 - \alpha_{p+1} + \alpha_{p+1}) - \ldots$$

or

$$0 \leq pZ - p - 1 - \ldots$$

where all the terms in $\ldots$ are non-negative. Thus, all solutions to (9) through (10) require $Z > (p+1)/p = 1 + 1/p = 1 + 1/\lfloor \ln^2 n \rfloor > 1 + 1/n^\beta$ for any fixed $\beta < 1$. This violates the requirement that $Z \leq 1$ in order for ϕ to be q-Horn.

The expected number of q-blocked c-cycles can be found and the second moment method applied to give the following result.

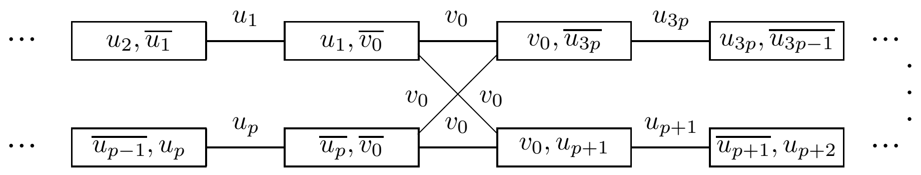

Figure 12: A "criss-cross loop" of $t = 3p + 2$ clause nodes. Each box or clause node represents a clause. An edge exists between two clause nodes if and only if the two corresponding clauses have a variable in common and the edge is labeled by that variable. Only "cycle literals" are shown in the nodes; "padding literals," required for $k \geq 3$, are present but are not shown. Padding literals are different from cycle variables of a criss-cross loop.

Theorem 32 *A random (n, m, k)-CNF expression is not q-Horn, with probability tending to 1, if $m/n > 4/(k^2 - k)$.* □

For $k = 3$ this is $m/n > 2/3$.

A similar analysis yields the same results for hidden Horn, SLUR, CC-balanced, or extended Horn expressions. The critical substructure which causes ϕ not to be SLUR is called a criss-cross loop. An example is shown is Figure 12. Looking at Figure 12 and Expression (8) we see that the SLUR and q-Horn classes are vulnerable to certain types of "cyclic" structures. Most other polynomial time solvable classes are similarly vulnerable to cyclic structures of various kinds. But the generator of random expressions is relatively blind to such cyclic structures: as m/n is increased, at some point cycles begin to appear in ϕ in abundance and when this happens, "killer" cycles also show up. Hence, cycles appear in abundance when $m/n > 1/O(k^2)$ and this is where "killer" cycles appear too.

But the Matched class (see Definition 10, Page 19) is not affected by cycles and consequently probabilistic analysis tells us that there are many "more" Matched expressions than SLUR or q-Horn expressions. Let $Q \subset \phi$ be any subset of clauses of ϕ. Define the *neighborhood* of Q, denoted $\mathbf{V}(Q)$, to be the set of variables that occur in Q. Recall the bipartite graph G^ϕ contains two vertex sets C^ϕ and V^ϕ, where the elements of C^ϕ are the clauses of ϕ, the elements of V^ϕ are the variables of ϕ, and the edges of G^ϕ connect $v \in V^\phi$ to $c \in C^\phi$ just when variable v appears (complemented or uncomplemented) in clause c. Thus $\mathbf{V}(Q)$ is also the set of vertices in V^ϕ adjacent to the vertices corresponding to Q.

Definition 33 *Define the* deficiency *of Q, denoted $\delta(Q)$, as $\delta(Q) = |Q| - |\mathbf{V}(Q)|$, that is, the excess of clauses over distinct variables in those clauses. A subset $Q \subseteq C^\phi$ is said to be* deficient *if $\delta(Q) > 0$.* □

The following theorem is well known.

Theorem 34 (Hall's Theorem [57])
Given a bipartite graph with vertex sets V^ϕ and C^ϕ, a matching that includes every vertex of C^ϕ exists if and only if no subset of C^ϕ is deficient. $\square$

Theorem 35 *Random (n, m, k)-CNF expressions are Matched expressions with probability tending to 1 if $m/n < r(k)$ where $r(k)$ is given by the following table [46].*

k	$r(k)$
3	*.64*
4	*.84*
5	*.92*
6	*.96*
7	*.98*
8	*.990*
9	*.995*
10	*.997*

Theorem 35 may be proved by finding a lower bound on the probability that G^ϕ has a matching that includes every vertex of C^ϕ. By Theorem 34 it is sufficient to prove an upper bound on the probability that there exists a deficient subset of C^ϕ, then show that the bound tends to 0 for $m/n < r(k)$ as given in the theorem. Using the first moment method, a sufficient upper bound is given by the expected number of deficient subsets.

These results are interesting for at least two reasons. First, Theorem 35 says that random expressions are Matched expressions with high probability if $m/n < r(k)$ and $r(k)$ is roughly 1. But, by Theorem 32 and similar results, random expressions are almost never one of the well-studied classes mentioned earlier unless $m/n < 4/(k^2 - k)$. This is somewhat disappointing because all the other classes were proposed for rather profound reasons, usually reflecting cases when corresponding instances of integer programming present polytopes with some special properties. And in spite of all the theory that helped establish these classes, the Matched class, ignored in the literature because it is so trivial in nature, turns out to be, in some probabilistic sense, much bigger than all the others.

Second, the results provide insight into the nature of larger classes of polynomial time solvable expressions. Classes vulnerable to "killer" cycles appear to be handicapped relative to classes that are not. In fact, the Matched class may be generalized considerably to larger polynomial time solvable classes such as Linear Autarkies [71, 74].

6.7 Randomized algorithms: upper bounds

Exploitable properties have been used to find $2^{n(1-\epsilon)}$ upper bounds on the number of steps needed to solve a given k-CNF expression, ϕ. The first non-trivial upper bound is found in the classic paper of Monien and Speckenmeyer [78] and is based on the notion of autark assignments. An assignment to a set of variables is said to be *autark* if all clauses that contain at least one of those variables are satsfied by the assignment. If an autark assignment is found during search, it is sufficient to fix the autark assignment below that point in the search: that is, backtracking to consider other values for the variables of the autark assignment is unnecessary at the point the autark assignment is discovered. An example is an assignment causing a pure literal to have value 1: clearly, it is unnecessary to expand the search space with a pure literal set to 0. The search algorithm analyzed in [78] is a DPLL variant which branches on all assignments to the variables of a shortest width clause except the one falsifying the clause. Such assignments either produce a clause of shorter width or are autark. This provides an upper bound of $O(\alpha_k^n)$, where α_k is the largest real root of the equation

$$\alpha_k^k - 2\alpha_k^{k-1} + 1 = 0.$$

Thus, if $k = 3$, the upper bound is $O(1.618^n)$ steps. The bound increases with increasing k and approaches $O(2^n)$. A number of more complex variants have been studied and the best bounds obtained in this way, so far, appear to be $O(1.505^n)$ [70] and $O(1.497^n)$ [88] for 3-SAT.

Better bounds have been obtained for probabilistic algorithms. A family of such algorithms exploit a relationship between the structure of a k-CNF expression and structure of the space of models expressed in terms of isolated models and critical clauses [80]. A *j-isolated model* is a model with exactly $n-j$ assignment neighbors differing in one variable that are also models. If j is close to n, the set of such models, called nearly isolated, for a given k-CNF expression has a relatively "short" description. Thus, a nearly isolated model, if one exists, is relatively easy to find by searching the space of relatively "short" descriptions. The important observation is that any satisfiable k-CNF expression either has a nearly isolated model or has very many models. In either case, a model can be found relatively quickly.

Short descriptions depend on the notion of critical clauses. If M is a model but reversing variable v_i in M is not a model, then there must exist a clause with exactly one literal that has value 1 under M and that literal is either v_i if $v_i \in M$ or $\bar{v}_i$ if $v_i \notin M$. This clause is said to be *critical* for variable i at model M. A critical clause cannot be critical for two different variables at the same model. Therefore, a j-isolated model has j critical clauses.

A description of a model can be encoded in fewer than n bits as follows. Let π be a permutation of $\{1, 2, ..., n\}$. Let M be a model for a k-CNF expression ϕ. Define a string $x^M \in \{0, 1\}^n$ such that $x_i^M = 1$ if and only if $v_i \in M$ (that is, v_i is assigned value 1). An encoding of x^M with respect to π, written $I_\pi(x^M)$, is a permuation of its bits according to π but with the ith bit deleted for any i such that 1) there is a critical clause for the variable v_{π_i} at M; and 2) v_{π_i} or $\bar{v}_{\pi_i}$ is the last literal in the clause if literals are ordered by π.

An encoding can be efficiently decoded. That is, a model M can be recovered efficiently from its encoding as described above. This is done by starting with the original expression ϕ, assigning values to variables, one at a time, in order, according to π, and reducing ϕ accordingly. Bits of the encoding are used to decide values of assignment M except in the case that the reduced ϕ has a unit clause consisting of the current variable, in order, or its complement. In that case, the current variable is set to satisfy that clause. This leads to the following lemma.

Lemma 36 - **The Satisfiability Coding Lemma** (*from [80]*) - *If M is a j-isolated model of a k-CNF expression ϕ, then its average (over all permutations π) description length under the encoding $I_\pi(x^M)$ is at most $n - j/k$.* $\square$

This has inspired the algorithm shown in Figure 13. Notice the algorithm essentially randomly chooses a permutation π and then simultaneously 1) attempts a decoding according to π; and 2) checks a potential satisfying assignment according to π. Performance of the algorithm is given by the following.

Theorem 37 (*from [80]*) *The algorithm of Figure 13 runs in $O(n^2|\phi|2^{n-n/k})$ time and finds a model for a given satisfiable k-CNF expression with probability tending to 1.* $\square$

The complexity is easy to see. The following outlines the reason the probability of finding a model tends to 1. Suppose ϕ has a j-isolated model x^M. Fix j critical clauses as above. By Lemma 36, the average number (over all π) of critical variables that appear last among the variables in their critical clauses is at least j/k. Since the maximum number of critical variables is n, for at least $1/n$ fraction of the permutations the number of such variables is at least j/k. Therefore, the probability that there are at least j/k critical clauses where the critical variables occur last for π (the variable choices for a single round of the algorithm) is at least $1/n$. Since x^M is j-isolated, if the algorithm is to output x^M it makes at most $n - j/k$ random assignments. So the probability that the algorithm outputs x^M on a round is at least $2^{-n+j/k}/n$. A straightforward summation over all models for ϕ determines a bound on the probability that the algorithm gives up.

An improved randomized algorithm which uses RandomSatSolver of Figure 13 is given in [81]. It finds a model, when one exists, for a 3-CNF expression with probability tending to 1 in $O(1.362^n)$ steps. The algorithm is peculiar because its performance actually deteriorates as the number of models for a given expression increases [60]. This is likely due to the emphasis of the algorithm on finding nearly isolated models and the lack of intelligence in looking for non-isolated models.

```
**Procedure** RandomSatSolver ($\phi$)
**Input**: a $k$-CNF expression $\phi$ with $n$ variables.
**Output**: either "give up" or a model for $\phi$.

**begin**
    repeat the following $n^2 2^{n-n/k}$ times {
        $M := \emptyset$;
        $V := \{v : \exists C \in \phi$ such that $v \in C$ or $\bar{v} \in C\}$;
        $\phi' := \phi$;
        while $V \neq \emptyset$ do the following {
            if $\emptyset \in \phi'$ break; // No assignment extension will be a model
            if $\phi' = \emptyset$ return $M$;
            randomly choose $v \in V$;
            if $\{v\} \in \phi'$ then
                $M := M \cup \{v\}$;
                $\phi' := \{C - \{\bar{v}\} : C \in \phi', v \notin C\}$;
            else if $\{\bar{v}\} \in \phi'$ then
                $\phi' := \{C - \{v\} : C \in \phi', \bar{v} \notin C\}$;
            else with probability 1/2 do the following
                $M := M \cup \{v\}$;
                $\phi' := \{C - \{\bar{v}\} : C \in \phi', v \notin C\}$;
            else do the following
                $\phi' := \{C - \{v\} : C \in \phi', \bar{v} \notin C\}$;
            $V := V \setminus \{v\}$;
        }
    }
    return "give up";
**end**;
```

Figure 13: A randomized algorithm for finding a model for a satisfiable k-CNF expression.

```
Procedure SLS ($\phi$)
Input: a $k$-CNF expression $\phi$ with $n$ variables.
Output: either "give up" or a model for $\phi$.

begin
    $V := \{v : \exists C \in \phi$ such that $v \in C$ or $\bar{v} \in C\}$;
    repeat the following $n(2(1-1/k))^n$ times {
        $M := \emptyset$;
        for each $v \in V$ set $M := M \cup \{v\}$ with probability 1/2;
        repeat the following $3n$ times {
            Let $\phi' := \{C \in \phi : \forall v \in C, v \notin M$ and $\forall \bar{v} \in C, v \in M\}$;
            if $\phi' = \emptyset$ return $M$;
            randomly choose $C \in \phi'$;
            randomly choose literal $l \in C$;
            if $l$ is a positive literal $v$ then $M := M \cup \{v\}$;
            otherwise, if $l$ is a negative literal $\bar{v}$ then $M := M \setminus \{v\}$;
        }
    }
    return "give up";
end;
```

Figure 14: A local-search randomized algorithm for finding a model for a satisfiable k-CNF expression. The factor n in the `repeat` loop can be improved but is good enough for the purposes of this exposition.

A better randomized algorithm for 3-CNF expressions, which is based on local search, is presented in [90] and shown in Figure 14. Each round of this algorithm begins with an assignment that is hoped to be fairly close to a model. Then, for $3n$ iterations, single variable assignment flips are made, attempting to eliminate all falsified clauses. After $3n$ iterations, if a model is not found, it is unlikely that the assignment chosen at the beginning of the round is close to a model, so the round terminates and another assignment is picked on the next round and the process repeats.

Theorem 38 (*adapted from [90]*) *The algorithm of Figure 14 has complexity $O(n|\phi|(2(1-1/k))^n)$ and finds a model for a given satisfiable k-CNF expression with probability tending to 1.* □

For 3-CNF expressions, this algorithm has $O(1.333^n)$ complexity. Contrary to the randomized algorithm of Figure 13, the performance of this algorithm improves as the number of models of an input increases. This observation is exploited in an algorithm presented in [60] where a round begins with a random assignment τ and then the $3n$ local search steps of SLS and the $O(n)$ Davis-Putnam steps of RandomSatSolver are run on τ. The resulting algorithm has complexity $O(1.324^n)$ for 3-CNF expressions.

6.8 Results inspired by statistical mechanics

New CNF algorithms, startlingly successful, have been developed by observing similarities in CNF structure and models of matter. We describe one here for the purpose of illustrating the impact non-rigorous methods may have on future SAT algorithm development.

We are concerned with the CNF structure that arises from the bipartite graph G^ϕ introduced on Page 50. Recall, given CNF expression ϕ, G^ϕ has vertex set V^ϕ corresponding to variables, vertex set C^ϕ corresponding to clauses, and edge set E such that for $v \in V^\phi$ and $c \in C^\phi$, $\langle v, c\rangle \in E$ if and only if clause c has either literal v or $\bar{v}$. It has been observed that if G^ϕ has few "short" cycles and ϕ has no model, then ϕ is a hard problem for resolution. For random (n, m, k)-CNF expressions, not only are cycles "long" but the minimum length cycle grows logarithmically with n for fixed m/n. Even when m/n is such that a random expression has a model with high probability, it is difficult for a variety of proposed algorithms to find such a model when m/n is greater than a certain threshold (about 4 in the case of random $(n, m, 3)$-CNF expressions).

At the core of the statistical physics of disordered systems is the spin glass problem. A model of a spin glass contains binary variables, $\sigma_1, \sigma_2, \sigma_3, ...$, called spins taking values +1 or -1, and energy relationships $E_1, E_2, E_3, ...$ among small groups of neighboring spins. This model can be represented graphically, as depicted in Figure 15, where circled vertices are spins, boxed vertices are energy functions, and edges show the dependence of each energy function on spins. Such a graph is bipartite: there is no edge between a pair of circle vertices and no edge between a pair of box vertices. It is a general rule in physics that systems tend to migrate toward their minimum energy (that is maximum entropy) state. Thus, the spin glass problem is to determine the minimum energy of a given spin glass model.

The spin glass problem is analogous to the problem of determining a model for a given CNF expression: spins take the role of Boolean variables, energy functions take the role of clauses, and energy corresponds to clauses satisfied. Thus, if the minimum energy of a spin glass is 0, the corresponding CNF expression has a model and if the minimum energy is greater than 0, the expression has no model. Energy functions consistent with the so called *Ising model* and expressing this behavior for k literal clauses composed of literals taken from variables $v_{i_1}, v_{i_2}, ..., v_{i_k}$ corresponding to spins $\sigma_{i_1}, \sigma_{i_2}, ..., \sigma_{i_k}$ are

$$E_i = 2\prod_{r=1}^{k} \frac{1 + J_i^r \sigma_{i_r}}{2}$$

where J_i^r is +1 if the rth literal of the ith clause is complemented and -1 if it is

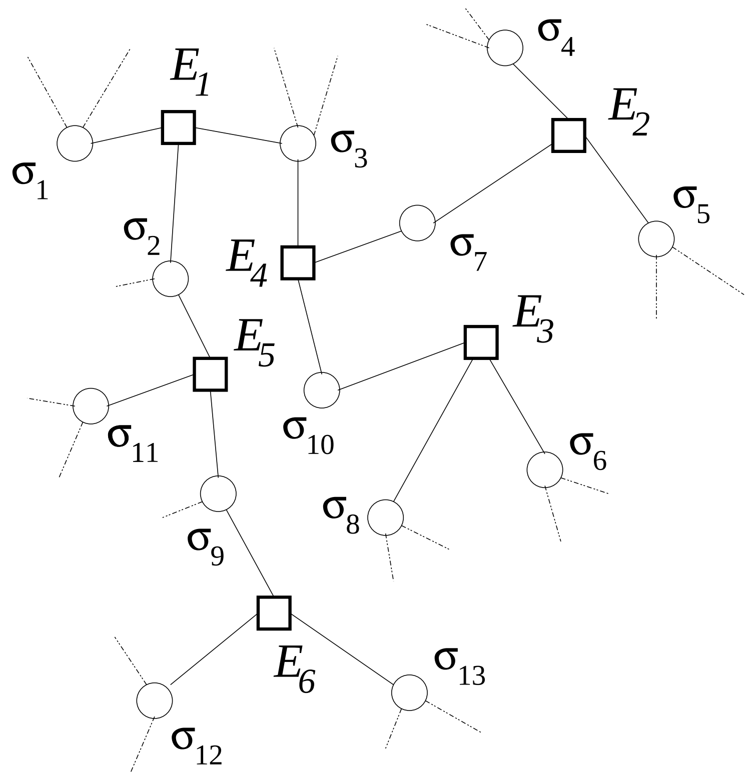

Figure 15: Model of a spin glass. Variables $\sigma_1, \sigma_2, ...$, called spins, are binary taking values +1 or -1. Functions $E_1, E_2, ...$ are energy functions, each of which depends on a small number of neighboring spins. The spin glass problem is to find spin values which minimize the total energy of a given collection of energy functions.

uncomplemented[12]. Thus, the energy of clause $\{v_1, \bar{v}_2, \bar{v}_3\}$ is $(1-\sigma_1)(1+\sigma_2)(1+\sigma_3)/4$. Observe this is 0 if $\sigma_1 = +1$, $\sigma_2 = -1$, or $\sigma_3 = -1$ and is 2 otherwise. The energy of an entire system of m energy functions is

$$E = \sum_{i=1}^{m} E_i.$$

All clauses are satisfied if and only if the energy of the analogous spin glass is 0.

Where can physicists help? They can make assumptions analogous to those that apply to the physical world and thereby aim for a level of understanding that computer scientists would not have thought of. It is natural in physics to consider the probability distribution of spins:

$$Pr(\sigma_1, \sigma_2, ..., \sigma_n) = \frac{1}{Z} e^{-\frac{1}{T}E}$$

where T is temperature and Z is a normalizing constant. This distribution follows from the observation that a system in equilibrium tends to seek its highest entropy, or equivalently, lowest energy state. At $T = 0$, the lowest energy states are the only significant terms contributing to this distribution (non-rigorously) and, given a system of energy functions as defined above, one is interested in the distribution at the 0, or lowest possible, energy state. It can be calculated in a fairly straightforward manner for each spin separately due to the assumption of a thermodynamic limit: that is, $\lim_{n\to\infty} E/n$ is bounded. By this assumption, what is happening at any particular spin is independent of what is happening at nearly all other spins. Thus, probability distributions are assumed to be decomposed into products of probability distributions of influential variables.

Let's see how this may apply to k-CNF analogs. For simplicity of notation, and without loss of generality, let E_i be a function of spins $\sigma_1, ..., \sigma_k$. Let $h_{j\to i}\cdot\sigma_j$ be the energy contributed to E_i by the spin σ_j assuming the effect of E_i on σ_j is disregarded. The h terms are called magnetic fields in physics. Let $u_{i\to j}$ be the contribution to the magnetic field of spin σ_j from E_i. Consider the marginal distribution for σ_1 of E_i. If the Ising model is assumed, one writes

$$\sum_{\sigma_2,...,\sigma_k} e^{-\frac{1}{T}(E_i - h_{2\to i}\cdot\sigma_2 - ... - h_{k\to i}\cdot\sigma_k)} = e^{\frac{1}{T}(w_{i\to 1} + u_{i\to 1}\cdot\sigma_1)}.$$

Since one is interested in the case $T = 0$, this simplifies to

$$\min_{\sigma_2,...,\sigma_k} \{E_i - h_{2\to i}\cdot\sigma_2 - ... - h_{k\to i}\cdot\sigma_k\} = -w_{i\to 1} - u_{i\to 1}\cdot\sigma_1.$$

For the k-CNF problem, E_i has been given on Page 56 and

$$\begin{aligned} w_{i\to 1} &= |h_{2\to i}| + ... + |h_{k\to i}| - \theta(J_i^2\cdot h_{2\to i})\cdot ...\cdot\theta(J_i^k\cdot h_{k\to i}) \\ u_{i\to 1} &= -J_i^1\cdot\theta(J_i^2\cdot h_{2\to i})\cdot ...\cdot\theta(J_i^k\cdot h_{k\to i}) \end{aligned}$$

[12]The factor of 2 is explained below.

where $\theta(x) = 1$ if $x > 0$ and $\theta(x) = 0$ if $x \leq 0$. Interpreting $h_{j\to i} > 0$ as evidence that σ_j should be +1 and $h_{j\to i} < 0$ as evidence that σ_j should be -1 to minimize the ith energy function (satisfy the ith clause), $w_{i\to 1} + u_{i\to 1} \cdot \sigma_1 = |h_{2\to i}| + ... + |h_{k\to i}|$ if for any $\sigma_2 ... \sigma_k$ the evidence *supports* the current value of the corresponding spin, or if no support by these h variables is given and σ_1 has a value which minimizes the ith energy function. But $w_{i\to 1} + u_{i\to 1} \cdot \sigma_1 = |h_{2\to i}| + ... + |h_{k\to i}| - 2$ if σ_1 is not set to minimize E_i and no support is given for any of $\sigma_2, ..., \sigma_k$ (the ith clause is falsified). This explains the factor of 2 used in defining clausal energy on Page 56. The minimum energy of the entire system given spin, say σ_j, has a particular value is

$$E|_{\sigma_j} = C - \sum_{i:\sigma_j \in E_i} w_{i\to j} - \sigma_j \sum_{i:\sigma_j \in E_i} u_{i\to j}$$

where C is some constant. Write

$$h_j = \sum_{i:\sigma_j \in E_i} u_{i\to j}. \tag{11}$$

Let $Q_{i\to j}(u)$ be the probability distribution of $u_{i\to j}$. Let $P_{j\to i}(h)$ be the probability distribution of the contribution of h_j to function E_i. Suppose energy functions $E_{\pi_1}, ..., E_{\pi_p}$ influence spin σ_j. By independence of distributions, which follows from the assumption of thermodynamic limit,

$$P_{j\to i}(h) = C_{j\to i} \sum_{u_1,...,u_p:\sum_{x=1}^{p} u_x = h} Q_{\pi_1\to j}(u_1) \cdot ... \cdot Q_{\pi_p\to j}(u_p) e^{y|h|}.$$

For each spin σ_j of E_i, let $\sigma_{\pi'_1}, ..., \sigma_{\pi'_{p'}}$ be the remaining spins of E_i.
Let $\theta(J_i^{\pi'_1} \cdot h_1) \cdot ... \cdot \theta(J_i^{\pi'_{p'}} \cdot h_{p'})$ be denoted by $w_{i,1...p'}$. Write

$$Q_{i\to j}(u) = C_{i\to j} \sum_{\substack{h_1, ..., h_{p'} : \\ u = -J_i^j \cdot w_{i,1...p'}}} P_{\pi'_1\to i}(h_1) \cdot ... \cdot P_{\pi'_{p'}\to i}(h_{p'}) e^{-y \cdot w_{i,1...p'}}.$$

The terms $C_{i\to j}$ and $C_{j\to i}$ are normalizing constants and y is a parameter which expresses the rate of change of complexity with respect to E/n, and complexity is a measure of the number of different energy states possible. Since this information is not known generally, y must be guessed.

The values for $Q_{i\to j}(u)$ and $P_{j\to i}(h)$ may be computed by the algorithm of Figure 16. It is then a simple matter to determine $P_j(h)$, the distribution for the field acting on spin σ_j. But the value of $P_j(h)$ suggests a setting for spin σ_j which minimizes energy: namely, +1 if $\sum_{h>0} P_j(h) > \sum_{h<0} P_j(h)$ and -1 if $\sum_{h>0} P_j(h) < \sum_{h<0} P_j(h)$. If $\sum_{h>0} P_j(h) = \sum_{h<0} P_j(h)$ or $P_h(0) = 1$ no bias is detected. This suggests the algorithm of Figure 17 for solving CNF expressions: assign values to "biased" variables first, recomputing $P_j(h)$ after each variable is assigned, then apply a standard SAT solver to complete the assignment.

Considerable success has been reported with this approach [75] on random (n, m, k)-CNF expressions and also some benchmarks which have been considered hard for a variety of SAT solvers. What is different about the method discussed here? First, it provides a way to choose initial variables and values, based on apparent probability distributions, that is intelligent enough to know when to stop. Traditional methods for choosing the first so many variables to branch on will choose as many values and variables that the user would like. Mistakes higher up the search process are very serious and, if made, can result in very long searches at the lower end. The method of Mézard and Zecchina seems to make few mistakes on many expressions. By maximizing entropy, variable/value choices tend to result in reduced expressions such that an additional variable choice will yield essentially the same reduced subexpressions regardless of its value. Second, more interdependence of CNF components is taken into account. We illustrate with the well-known Johnson heuristic [61] which chooses a variable and assigns a value so as to maximize the probability that a 0 energy state (a model) exists assuming clauses are statistically independent (an unlikely situation). Although successful in several cases, this heuristic cannot really see very far ahead of the current state. But the method of Mézard and Zecchina is designed to, in some sense, explore all possible energy states for a given expression or subexpression, particularly the lowest energy states, and present statistics on those states and their causes.

```
Procedure SP ($G^\phi$)
Input: bipartite graph $G^\phi$ relating energy functions to spins.
Output: set of distributions of all spin fields.

begin
    for all energy function $E_i$ to spin $\sigma_j$ edges in $G^\phi$ define:
        $Q_{i\to j}(u) = \begin{cases} c_{i\to j} & \text{if } u = 0 \\ 1 - c_{i\to j} & \text{if } u = J_i^j \\ 0 & \text{otherwise} \end{cases}$
        and initialize $c_{i\to j}$ randomly;
    repeat the following until convergence {
        select an energy function $E_i$;
        for each spin $\sigma_j$ input to $E_i$ compute $P_{j\to i}(h) =$

            $C_{j\to i} \sum_{u_1,...,u_p : \sum_{x=1}^{p} u_x = h} Q_{\pi_1\to j}(u_1) \cdot ... \cdot Q_{\pi_p\to j}(u_p) e^{y|h|};$

        where $E_{\pi_1}, ..., E_{\pi_p}$ are functions depending on spin $\sigma_j$ except $E_i$
        and $C_{j\to i}$ is a normalizing constant;
        for each spin $\sigma_j$ input to $E_i$ let $\theta(J_i^{\pi_1} \cdot h_1) \cdot ... \cdot \theta(J_i^{\pi_{p'}} \cdot h_{p'})$
        be denoted by $w_{i,1...p'}$ (that is, from Page 58, $w_{i,1...p'}$
        denotes $w_{i\to j} - |h_{\pi'_1}| - ... - |h_{\pi'_{p'}}|$ where $\sigma_{\pi'_1}...\sigma_{\pi'_{p'}}$ are spins of
        $E_i$) and compute $Q_{i\to j}(u) =$

            $C_{i\to j} \sum_{h_1,...,h_{p'} : u = -J_i^j \cdot w_{i,1...p'}} P_{\pi'_1\to i}(h_1) \cdot ... \cdot P_{\pi'_{p'}\to i}(h_{p'}) e^{-y \cdot w_{i,1...p'}}$

    }
    Compute $P_j(h) =$

        $C_j \sum_{u_1,...,u_{p+1} : \sum_{x=1}^{p+1} u_x = h} Q_{\pi_1\to j}(u_1) \cdot ... \cdot Q_{\pi_{p+1}\to j}(u_{p+1}) e^{y|h|}.$

        where $E_{\pi_1}...E_{\pi_{p+1}}$ are functions depending on spin $\sigma_j$;
    return $P_j(h)$ for all $j$;
end;
```

Figure 16: *An algorithm for computing the distributions $P_j(h)$.*

```
Procedure SID ($\phi$)
Input: a CNF expression $\phi$.
Output: either "unsatisfiable" or a model for $\phi$.

begin
    repeat the following indefinitely {
        $M := \emptyset$;
        establish $G^\phi$ and variables $u_{i\to j}$;
        randomly choose values for $u_{i\to j}$;
        repeat the following until $\emptyset \in \phi$ {
            run SP on $G^\phi$ to evaluate $P_j(h)$;
            if $P_j(h) = 0$ except at $h = 0$ for all $j$ then {
                apply a SAT solver to $\phi$;
                if it finds a model then return that model;
            }
            for all $\sigma_j$, let $w_j^- = \sum_{h<0} P_j(h)$ and $w_j^+ = \sum_{h>0} P_j(h)$;
            choose $\sigma_j$ such that $|w_j^+ - w_j^-|$ is a maximum;
            if $w_j^+ > w_j^-$ then $\sigma_j := +1$; else $\sigma_j := -1$;
            if $\sigma_j = +1$ then {
                $M := M \cup \{v_j\}$;
                $\phi := \{C \setminus \{\bar{v}_j\} : C \in \phi, v_j \notin C\}$;
            } else
                $\phi := \{C \setminus \{v_j\} : C \in \phi, \bar{v}_j \notin C\}$;
            while there exists a unit clause $\{l\} \in \phi$ do the following {
                if $l$ is an uncomplemented literal then $M := M \cup \{l\}$;
                $\phi := \{C \setminus \{\bar{l}\} : C \in \phi, l \notin C\}$;
            }
            if $\phi = \emptyset$ then return $M$;
        }
    }
end;
```

Figure 17: *An algorithm for solving a CNF expression based on maximizing entropy.*

References

[1] D. Achlioptas, Lower bounds for random 3-SAT via differential equations, *Theoretical Computer Science* 265 (2001) pp. 159–185.

[2] D. Achlioptas, Setting 2 variables at a time yields a new lower bound for random 3-sat, in *32nd ACM Symposium on Theory of Computing*, Association for Computing Machinery, New York, 2000, pp. 28–37.

[3] D. Achlioptas and G. Sorkin, Optimal myopic algorithms for random 3-SAT, in *41st Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 2000, pp. 590–600.

[4] D. Achlioptas and C. Moore, The asymptotic order of the random k-SAT thresholds, in *43rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 2002, pp. 779–788.

[5] D. Achlioptas and Y. Peres, The threshold for random k-SAT is $2^k \log 2 - O(k)$. *Journal of the American Mathematical Society* 17 (2004) pp. 947–973.

[6] D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc, M. Molloy and Y. Stamatiou, Random constraint satisfaction: a more accurate picture, in *3rd Conference on the Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science 1330, Springer, Berlin, 1997, pp. 107–120.

[7] D. Achlioptas, L.M. Kirousis, E. Kranakis and D. Krizanc, Rigorous results for random $(2+p)$-SAT, *Theoretical Computer Science* 265 (2001) pp. 109–129.

[8] D. Achlioptas and M. Molloy, The analysis of a list-coloring algorithm on a random graph, in *38th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 204–212.

[9] R. Aharoni and N. Linial, Minimal non-two colorable hypergraphs and minimal unsatisfiable formulas, *Journal of Combinatorial Theory, Series A* 43 (1986) pp. 196–204.

[10] N. Alon and J. Spencer, *The Probabilistic Method* (2nd Edition), Wiley, New York, 2000.

[11] P. Beame and T. Pitassi, Simplified and improved resolution lower bounds, in *37th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 274–282.

[12] P. Beame, R.M. Karp, T. Pitassi and M. Saks, On the complexity of unsatisfiability proofs for random k-CNF formulas, in *30th Annual Symposium on the Theory of Computing*, Association for Computing Machinery, New York, 1998, pp. 561–571.

[13] E. Ben-Sasson and A. Wigderson, Short proofs are narrow - resolution made simple, *Journal of the Association for Computing Machinery* 48 (2001) pp. 149–169.

[14] B. Bollobás, C. Borgs, J. Chayes, J.H. Kim and D.B. Wilson, The scaling window of the 2-SAT transition, *Random Structures and Algorithms* 18 (2001) pp. 201–256.

[15] E. Boros, P.L. Hammer and X. Sun, Recognition of q-Horn formulae in linear time, *Discrete Applied Mathematics* 55 (1994) pp. 1–13.

[16] E. Boros, Y. Crama, P.L. Hammer and M. Saks, A complexity index for satisfiability problems, *SIAM Journal on Computing* 23 (1994) pp. 45–49.

[17] A.Z. Broder, A.M. Frieze and E. Upfal, On the satisfiability and maximum satisfiability of random 3-CNF formulas, in *4th Annual ACM-SIAM Symposium on Discrete Algorithms*, Association for Computing Machinery, New York, 1993, pp. 322–330.

[18] V. Chandru and J.N. Hooker, Extended Horn sets in propositional logic. *Journal of the Association for Computing Machinery* 38 (1991) pp. 205–221.

[19] M.-T. Chao, Probabilistic Analysis and Performance Measurement of Algorithms for the Satisfiability Problem, Ph.D. Thesis, 41–45, Department of Computer Engineering and Science, Case Western Reserve University, 1985.

[20] M.-T. Chao and J. Franco, Probabilistic analysis of two heuristics for the 3-Satisfiability problem, *SIAM Journal on Computing* 15 (1986) pp. 1106–1118.

[21] M.-T. Chao and J. Franco, Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem, *Information Sciences* 51 (1990) pp. 289–314.

[22] P. Cheeseman, B. Kanefsky and W.M. Taylor, Where the really hard problems are, in *12th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1991, pp. 331–340.

[23] V. Chvátal and Bruce Reed, Mick gets some (the odds are on his side), in *33th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 620–627.

[24] V. Chvátal and E. Szemerédi, Many hard examples for resolution, *Journal of the Association for Computing Machinery* 35 (1988) pp. 759–768.

[25] M. Conforti, G. Cornuéjols, A. Kapoor, K. Vušković and M.R. Rao, Balanced Matrices, in: J.R. Birge and K.G. Murty, eds., *Mathematical Programming: State of the Art*, Braun-Brumfield, United States. Produced in

association with the 15th International Symposium on Mathematical Programming, University of Michigan, 1994.

[26] S.A. Cook, The complexity of theorem-proving procedures, in *3rd Annual ACM Symposium on Theory of Computing*, Association for Computing Machinery, New York, 1971, pp. 151–158.

[27] S.A. Cook and R.A. Reckhow, Corrections for "On the lengths of proofs in the propositional calculus preliminary version," *SIGACT News (ACM Special Interest Group on Automata and Computability Theory)* 6 (1974) pp. 15–22.

[28] N. Creignou and H. Daudé, Generalized satisfiability problems: minimal elements and phase transitions, *Theoretical Computer Science* 302 (2003) pp. 417–430.

[29] N. Creignou and H. Daudé, Smooth and sharp thresholds for random k-XOR-CNF satisfiability, *Theoretical Informatics and Applications* 37 (2003) pp. 127–148.

[30] N. Creignou and H. Daudé, Combinatorial sharpness criterion and phase transition classification for random CSPs, *Information and Computation* 190 (2004) pp. 220–238.

[31] N. Creignou, H. Daudé and J. Franco, A sharp threshold for q-Horn, *Discrete Applied Mathematics* 153 (2005) pp. 48–57.

[32] M. Davis, G. Logemann and D. Loveland, A machine program for theorem proving, *Communications of the ACM* 5 (1962) pp. 394–397.

[33] W.F. Dowling and J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *Journal of Logic Programming* 1 (1984) pp. 267–284.

[34] O. Dubois and Y. Boufkhad, A general upper bound for the satisfiability threshold of random r-SAT formulae, *Journal of Algorithms* 24 (1997) pp. 395–420.

[35] O. Dubois. Upper bounds on the satisfiability threshold. *Theoretical Computer Science* 265 (2001) pp. 187–197.

[36] O. Dubois, Y. Boufkhad and J. Mandler, Typical random 3-SAT formulae and the satisfiability threshold, in *11th ACM-SIAM Symposium on Discrete Algorithms*, Association for Computing Machinery, New York, 2000, pp. 124–126.

[37] A. El Maftouhi and W. Fernandez de la Vega, On random 3-SAT, *Combinatorics, Probability, and Computing* 4 (1995) pp. 189–195.

[38] S. Even, A. Itai and A. Shamir, On the complexity of timetable and multicommodity flow problems, *SIAM Journal on Computing* 5 (1976) pp. 691–703.

[39] W. Fernandez de la Vega, On random 2-sat, manuscript, 1992.

[40] J. Franco, Results related to threshold phenomena research in Satisfiability: lower bounds, *Theoretical Computer Science* 265 (2001) pp. 147–157.

[41] J. Franco, On the probabilistic performance of algorithms for the Satisfiability problem, *Information Processing Letters* 23 (1986) pp. 103–106.

[42] J. Franco, On the occurrence of null clauses in random instances of Satisfiability, *Discrete Applied Mathematics* 41 (1993) pp. 203–209.

[43] J. Franco, Elimination of infrequent variables improves average case performance of Satisfiability algorithms, *SIAM Journal on Computing* 20 (1991) pp. 1119–1127.

[44] J. Franco and Y.C. Ho, Probabilistic performance of heuristic for the Satisfiability problem. *Discrete Applied Mathematics* 22 (1988/89) pp. 35–51.

[45] J. Franco and M. Paull, Probabilistic analysis of the Davis-Putnam procedure for solving the Satisfiability problem, *Discrete Applied Mathematics* 5 (1983) pp. 77–87.

[46] J. Franco and A. Van Gelder, A perspective on certain polynomial time solvable classes of Satisfiability, *Discrete Applied Mathematics* 125 (2003) pp. 177–214.

[47] E. Friedgut, and an appendix by J. Bourgain, Sharp thresholds of graph properties, and the k-SAT problem, *Journal of the American Mathematical Society* 12 (1999) pp. 1017–1054.

[48] A.M. Frieze and S. Suen, Analysis of two simple heuristics on a random instance of k-SAT, *Journal of Algorithms* 20 (1996) pp. 312–355.

[49] X. Fu, On the complexity of proof systems, Ph.D. Thesis, University of Toronto, Canada, 1995.

[50] Z. Galil, On resolution with clauses of bounded size, *SIAM Journal on Computing* 6 (1977) pp. 444–459.

[51] A. Goerdt, A threshold for unsatisfiability, *Journal of Computer System Science* 53 (1996) pp. 469–486.

[52] A. Goerdt and M. Krivelevich, Efficient recognition of random unsatisfiable k-SAT instances by spectral methods, in *Lecture Notes in Computer Science* Vol. 2010, Springer, Berlin, 2001, pp. 294–304.

[53] A. Goldberg, On the complexity of the satisfiability problem, in *4th Workshop on Automated Deduction*, Academic Press, 1979, pp. 1–6.

[54] A. Goldberg, P.W. Purdom and C. Brown, Average time analysis of simplified Davis-Putnam procedures, *Information Processing Letters* 15 (1982) pp. 72–75.

[55] M.T. Hajiaghayi and G.B. Sorkin, The Satisfiaility threshold of random 3-SAT is at least 3.52, available from http://arxiv.org/pdf/math.CO/0310193.

[56] A. Haken, The intractability of resolution, *Theoretical Computer Science* 39 (1985) pp. 297–308.

[57] P. Hall, On representatives of subsets, *Journal of London Mathematical Society* 10 (1935) pp. 26–30.

[58] A. Itai and J. Makowsky, On the complexity of Herbrand's theorem, Working paper 243, Department of Computer Science, Israel Institute of Technology, Haifa, 1982.

[59] K. Iwama, CNF Satisfiability test by counting and polynomial average time, *SIAM Journal on Computing* 18 (1989) pp. 385–391.

[60] K. Iwama, Improved upper bounds for 3-SAT, *Electronic Colloquium on Computational Complexity* Report No. 53, 2003, avaliable from `http://www.eccc.uni-trier.de/eccc-reports/2003/TR03-053/Paper.pdf.`

[61] D.S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and Systems Sciences* 9 (1974) pp. 256–278.

[62] A. Kamath, R. Motwani, K. Palem and P. Spirakis, Tail bounds for occupancy and the satisfiability conjecture, *Random Structures and Algorithms* 7 (1995) pp. 59–80.

[63] A.C. Kaporis, L.M. Kirousis and E.G. Lalas, The probabilistic analysis of a greedy satisfiability algorithm, in *Lecture Notes in Computer Science* Vol. 2461, Springer, Berlin, 2002, pp. 574–586.

[64] A.C. Kaporis, L.M. Kirousis and E.G. Lalas, Selecting complementary pairs of literals, *Electronic Notes in Discrete Mathematics* Vol. 16, Elsevier, the Netherlands, 2003, pp. 47-70.

[65] A.C. Kaporis, L.M. Kirousis and Y.C. Stamatiou, How to prove conditional randomness using the principle of deferred decisions, 2002, available from http://www.ceid.upatras.gr/faculty/kirousis/kks-pdd02.ps.

[66] R.M. Karp and M. Sipser, Maximum matchings in sparse random graphs, in *22nd Annual Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1981, pp. 364–375.

[67] L.M. Kirousis, E. Kranakis, D. Krizanc and Y.C. Stamatiou, Approximating the unsatisfiability threshold of random formulae, *Random Structures and Algorithms* 12 (1998) pp. 253–269.

[68] H. Kleine Büning, On the minimal unsatisfiability problem for some subclasses of CNF, in *Abstracts of 16th International Symposium on Mathematical Programming*, Mathematical Programming Society, 1997.

[69] M. Krivelevich and V.H. Vu, Approximating the independence number and the chromatic number in expected polynomial time, in *Lecture Notes in Computer Science* Vol. 1853, Springer, Berlin, 2000, pp. 13-24.

[70] O. Kullmann, New methods for 3-SAT decision and worst-case analysis, *Theoretical Computer Science* 223 (1999) pp. 1–72.

[71] O. Kullmann, Investigations on autark assignments, *Discrete Applied Mathematics* 107 (2000) pp. 99-137.

[72] T.G. Kurtz, Solutions of ordinary differential equations as limits of pure jump Markov processes, *Journal of Applied Probability* 7 (1970) pp. 49–58.

[73] H.R. Lewis, Renaming a set of clauses as a Horn set, *Journal of the Association for Computing Machinery* 25 (1978) pp. 134–135.

[74] H. van Maaren, A short note on linear autarkies, q-Horn formulas, and the complexity index, Technical Report 99-26, Dimacs, Rutgers University, New Brunswick, New Jersey, 1999.

[75] M. Mézard and Riccardo Zecchina, The random k-satisfiability problem: from an analytic solution to an efficient algorithm, *Physical Review E* Vol. 66, American Physical Society, 2002, pp. 056126–056126-27.

[76] M. Mitzenmacher, Tight thresholds for the pure literal rule, DEC/SRC Technical Note 1997-011, June 1997.

[77] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman and L. Troyansky, Determining computational complexity from characteristic "phase transitions," *Nature* 400 (1999) pp. 133–137.

[78] B. Monien and E. Speckenmeyer, Solving Satisfiability in less than 2^n steps, *Discrete Applied Mathematics* 10 (1985) pp. 287–295.

[79] V.Y. Pan and Z.Q. Chen, The complexity of the matrix eigenproblem, in *31st ACM Symposium on Theory of Computing*, Association for Computing Machinery, New York, 1999, pp. 506–516.

[80] R. Paturi, P. Pudlák and F. Zane, Satisfiability coding lemma, in *38th Annual Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 566–574.

[81] R. Paturi, P. Pudlák, M.E. Saks and F. Zane, An improved exponential-time algorithm for k-SAT, in *39th Annual Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 628–637.

[82] J.W. Plotkin, J.W. Rosenthal and J. Franco, Correction to probabilistic analysis of the Davis-Putnam procedure for solving the satisfiability problem, *Discrete Applied Mathematics* 17 (1987) pp. 295–299.

[83] P.W. Purdom and G.N. Haven, Probe order backtracking, *SIAM Journal on Computing* 26 (1997) pp. 456–483.

[84] P.W. Purdom and C.A. Brown, The pure literal rule and polynomial average time, *SIAM Journal on Computing* 14 (1985) pp. 943–953.

[85] P.W. Purdom, Search rearrangement backtracking and polynomial average time. *Artificial Intelligence* 21 (1983) pp. 117–133.

[86] J.A. Robinson, A machine-oriented logic based on the resolution principle, *Journal of the ACM* 12 (1965) pp. 23–41.

[87] J.W. Rosenthal, J.W. Plotkin and J. Franco, The probability of pure literals, *Journal of Logic and Computation* 9 (1999) pp. 501–513.

[88] I. Schiermeyer, Pure literal look ahead: an $O(1.497^n)$ 3-Satisfiability algorithm, in *1st Workshop on Satisfiability*, Certosa di Pontignano, Università degli Studi di Siena, Italy, 1996, available from:
`http://www.ececs.uc.edu/ franco/Sat-workshop/ps/Schiersat.ps`.

[89] J.S. Schlipf, F. Annexstein, J. Franco and R. Swaminathan, On finding solutions for extended Horn formulas, *Information Processing Letters* 54 (1995) pp. 133–137.

[90] U. Schöning, A probabilistic algorithm for k-SAT and constraint satisfaction problems, in *40th Annual Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 1999, pp. 410–414.

[91] M.G. Scutella, A note on Dowling and Gallier's top-down algorithm for propositional Horn Satisfiability, *Journal of Logic Programming* 8 (1990) pp. 265–273.

[92] R.P. Swaminathan and D.K. Wagner, The arborescence–realization problem, *Discrete Applied Mathematics* 59 (1995) pp. 267–283.

[93] C.A. Tovey, A simplified NP-complete satisfiability problem, *Discrete Applied Mathematics* 8 (1984) pp. 85–89.

[94] K. Truemper, Monotone Decomposition of Matrices, Technical Report UTDCS-1-94, University of Texas at Dallas, 1994.

[95] K. Truemper, Effective Logic Computation. Wiley, New York, 1998.

[96] A. Urquhart, Hard examples for resolution, *Journal of the Association for Computing Machinery* 34 (1987) pp. 209–219.

[97] N.C. Wormald, Differential equations for random processes and random graphs, *Annals of Applied Probability* 5 (1995) pp. 1217–1235.