Should Algorithms for Random SAT and Max-SAT be Different?

Sixue Liu
Microsoft Research
Redmond, WA, USA

Gerard de Melo
Rutgers University
New Brunswick, NJ, USA

Abstract

We analyze to what extent the random SAT and Max-SAT problems differ in their properties. Our findings suggest that for random k-CNF with ratio in a certain range, Max-SAT can be solved by any SAT algorithm with subexponential slowdown, while for formulae with ratios greater than some constant, algorithms under the random walk framework require substantially different heuristics. In light of these results, we propose a novel probabilistic approach for random Max-SAT called ProMS. Experimental results illustrate that ProMS outperforms many state-of-the-art local search solvers on random Max-SAT benchmarks.

1 Introduction

Given a number of Boolean variables and constraint clauses in conjunctive normal form (CNF), the classic SAT problem consists in determining whether an assignment exists that satisfies all constraints. Its optimization variant, Max-SAT, in contrast, aims to find an assignment that maximizes the number of satisfied clauses. Unfortunately, both problems are NP-hard, and for the Max-3-SAT variant, it is hard to approximate optimal solutions within a factor of more than $7/8$ (Håstad 2001). This can also be viewed from the perspective of a worst-case upper bound analysis: The time complexity for Max-k-SAT (i.e., Max-SAT with clauses of up to k variables) on n variables is $\tilde{O}(2^{(1-\mu)n})$. Although an asymptotic lower bound for μ has been proven, for $k > 2$, no specific constants are known (Williams 2005; Chen and Santhanam 2015).

In the context of random formulae, which have also been the subject of extensive study, the main focus is on the satisfiability threshold and algorithmic polynomial upper bounds for low ratios of clauses to variables (Achlioptas and Moore 2006; Coja-Oghlan and Frieze 2014). Recent theory shows that there exist thresholds for satisfying any large constant fraction of clauses (Achlioptas, Naor, and Peres 2007). However, to the best of our knowledge, there is no algorithm tailored to random Max-k-SAT.

Algorithms for SAT and Max-SAT are often categorized as either complete or incomplete. The former guarantee the optimality of the output but may fail to deliver any solution at all within a given time. Incomplete algorithms, in contrast, can deliver a solution within a given time but do not guarantee its optimality.

Within the paradigm of incomplete algorithms, a common strategy is to rely on local search. Given an objective function, local search algorithms begin with a candidate assignment and traverse the candidate space by iteratively moving to neighboring ones, while keeping track of the best solution encountered in the course of this traversal (Hoos and Stützle 2004). If the objective is to minimize the number of unsatisfied clauses, then the incremental nature of such a search implies that local search algorithms for Max-SAT can essentially be used to solve SAT, while SAT algorithms are empirically expected to deliver somewhat reasonable solutions when applied to Max-SAT problem instances.

A number of local search algorithms have been proposed. For SAT, the influential GSAT and WalkSAT algorithms pursue a local search in a greedy or probabilistic manner (Selman, Levesque, and Levesque 1992; Selman, Kautz, and Cohen 1994). More recently, configuration checking (CC) (Cai and Su 2012) and probSAT (Balint and Schöning 2012) have made important advances, scoring highest in the random tracks at the SAT competitions 2012 and 2013, respectively. Additionally, breakthrough on random k-SAT at the phase transition point has been made by polyLS (Liu and Papakonstantinou 2016). For Max-SAT specifically, Iterated Robust Tabu Search (Smyth, Hoos, and Stützle 2003) was ranked first in the Max-SAT Evaluation 2012. Later, a variant of CC named CCLS (Luo et al. 2015) was highest-ranked in the Unweighted Random track of the Max-SAT Evaluations 2013 and 2015, respectively, improving considerably over previous results.

Contributions. Our main contribution is to shed new light on the different nature of SAT and Max-SAT when faced with random formulae with different ratios. This theory also has practical implications bearing on the choice of heuristics employed in local search algorithms. We instantiated and empirically evaluated these in a new probabilistic algorithm, called ProMS (Probability Distribution-based Max-SAT Solving), which is shown to outperform the previous state-of-the-art on random instances. The significance of these results also stems from the observation that, despite the success of purely probabilistic strategies for SAT, to the best of our knowledge, no one has previously succeeded in adopting such a strategy

for Max-SAT, due to the somewhat different nature of the problems.

2 Preliminaries

Our input is a formula $F = c_1 \wedge \cdots \wedge c_m$ in conjunctive normal form (CNF), where the c_i are disjunctive clauses that consist of literals (Boolean variables or their negations) on a set of variables $V = \{x_1, x_2, \ldots, x_n\}$. A k-CNF formula is a CNF such that each clause contains at most k literals. Given the number of clauses m and number of variables n, a *uniform random k-CNF* instance consists of m clauses chosen randomly and independently from all $(2n)^k$ possible clauses, with replacement. We also define $r = \frac{m}{n}$ as the *ratio* of this formula. Throughout this paper, "random" shall always refer to "uniform random", and $F_k(n, r)$ shall denote a random k-CNF with n variables and $m = rn$ clauses. A complete assignment α is a candidate solution such that each variable has a truth value of 0 (*false*) or 1 (*true*). The Max-SAT problem consists in finding an α that minimizes the number of unsatisfied clauses, denoted as $MaxSAT(F)$. Similarly, $SAT(F)$ is the problem of finding an α satisfying all clauses in F. We use $\mathcal{O}(T(n))$ to represent $poly(n) \cdot T(n)$ where $poly(n)$ refers to polynomials of n.

3 Analyzing Random SAT and Max-SAT

We believe that for $F_k(n, r)$, when the ratio r is within a certain (low) range, there is no need to distinguish algorithms for $SAT(F)$ and $MaxSAT(F)$, i.e., an optimal SAT algorithm is also optimal for Max-SAT, while for high ratio r, one is well-advised to adopt different heuristics.

Low Ratio Random k-CNF

We first consider the low ratio case. We start with the following lemma, and then explain why the precondition within this lemma is very likely true.

Lemma 1. *If the lower bound of the time complexity of all algorithms on random k-SAT ($k > 2$) with n variables and ratio greater than some constant r_k^p is Δ^n ($1 < \Delta \leq 2$), then, given $F_k(n, r)$, if there exists an assignment violating $o(m/\log m)$ clauses, it follows that $MaxSAT(F)$ can be solved in $\mathcal{O}(\min(2, \Delta + \epsilon)^n)$ steps for any $\epsilon > 0$.*

Proof. If $\Delta = 2$, trivially Max-SAT can be solved in 2^n steps. Otherwise, we solve $MaxSAT(F)$ in $\mathcal{O}((\Delta + \epsilon)^n)$ steps by i) enumerating all the combinations of violated clauses, ii) deleting them from F (we randomly shuffle the order of enumerating deleted clauses to maintain the uniform randomness of F'), iii) checking the satisfiability of the remaining formulae F' using the SAT algorithm in Δ^n steps. The number of combinations of violating $o(m/\log m)$ clauses is bounded by the following:

$$\begin{aligned}\sum_{i=0}^{o(m/\log m)} \binom{m}{i} &< o(m/\log m)\binom{m}{o(m/\log m)} \\ &< o(m/\log m)m^{o(m/\log m)} \\ &= o(m/\log m)2^{o(m)} \\ &= 2^{o(m)} = 2^{o(n)}\end{aligned}$$

Thus the SAT algorithm is invoked at most $2^{o(n)}$ times. Since $SAT(F')$ can be solved in Δ^n steps, we obtain that $MaxSAT(F)$ can be solved in $\mathcal{O}((\Delta + \epsilon)^n)$ steps for any $\epsilon > 0$. □

The precondition in Lemma 1 is that no algorithm can solve random k-SAT ($k > 2$) with ratio greater than some constant in subexponential time. Four considerations suggest that this conjecture holds true.

1. Random walk solves 3-CNF with ratio smaller than 1.63 in polynomial time, with ratio greater than 1.63 still remaining unknown (Alekhnovich and Ben-Sasson 2007).
2. The best result for solving random k-SAT in polynomial time is on formula with $r \leq \rho \cdot 2^k/k$ for some constant ρ ($\rho = \frac{1}{25}$ has been proposed in the literature), while for greater r no such algorithm is known (Coja-Oghlan and Frieze 2014).
3. There are exponential lower bounds for DPLL algorithms on satisfiable random 3-CNF formulae (Achlioptas and Menchaca-Mendez 2012).
4. The exponential time hypothesis (ETH) for the NPC class is widely considered true (Impagliazzo, Paturi, and Zane 2001).

Before continuing, we first give the following lemma, which later contributes to our main results.

Lemma 2. *For any integer $k \geq 2$, there exists a $r_k^+ > 0$, such that given $F_k(n, r)$ with $r > r_k^+$, there exists an assignment violating $o(m/\log m)$ clauses with probability $2^{-\Omega(n)}$.*

Proof. Let X_s be the number of assignments that satisfy at least s clauses in F and let X_α be the number of clauses that a random assignment α satisfies in F. We have $\mathbb{E}[X_s] = 2^n Pr[X_\alpha \geq s]$. The event that each clause is satisfied by an α is independent, so assuming that $\tilde{X}_\alpha$ denotes α satisfying a random clause in F, we find that

$$\begin{aligned}\mathbb{E}[X_s] &= 2^n \sum_{i=s}^{m} \left(\binom{m}{i} Pr[\tilde{X}_\alpha]^i (1 - Pr[\tilde{X}_\alpha])^{m-i} \right) \\ &= 2^n \sum_{i=s}^{m} \left(\binom{m}{i} \left(\frac{2^k - 1}{2^k}\right)^i \left(\frac{1}{2^k}\right)^{m-i} \right). \quad (1)\end{aligned}$$

Since $i \geq o(m/\log m)$, this sum is dominated by the first term with $i = s$, thus

$$\begin{aligned}\mathbb{E}[X_s] &\simeq 2^n \binom{m}{s} \left(\frac{2^k - 1}{2^k}\right)^s \left(\frac{1}{2^k}\right)^{m-s} \\ &< 2^n \binom{m}{s} \left(\frac{2^k - 1}{2^k}\right)^s\end{aligned}$$

Let $s = m - o(m/\log m)$, this becomes:

$$\begin{aligned}\mathbb{E}[X_s] &< 2^n \binom{m}{o(m/\log m)} \left(\frac{2^k-1}{2^k}\right)^{m-o(m/\log m)} \\ &< 2^n \left(\frac{2^k m}{2^k-1}\right)^{o(m/\log m)} \left(\frac{2^k-1}{2^k}\right)^m \\ &< 2^n 2^{o(m)} \left(\frac{2^k-1}{2^k}\right)^m \\ &< 2^n \left(\frac{2^k-1}{2^k}(1+\delta)\right)^m.\end{aligned}$$

Here δ can be any positive constant. We choose $\delta = \frac{1}{2^k}$ to bound $\mathbb{E}[X_s]$ from above:

$$\mathbb{E}[X_s] < 2^n \left(1 - \frac{1}{4^k}\right)^m = \left(2(1-\frac{1}{4^k})^r\right)^n.$$

By selecting $r = 1/\log\frac{4^k}{4^k-1} + \eta$ for some positive constant η, we have $2(1-\frac{1}{4^k})^r < 1$, and $\mathbb{E}[X_s] = 2^{-\Omega(n)}$. Applying Markov's inequality, we obtain

$$Pr[X_s > 0] = Pr[X_s \geq 1] \leq \mathbb{E}[X_s] = 2^{-\Omega(n)}.$$

Recall that $s = m - o(m/\log m)$. The conclusion follows. □

As a counterpart to Lemma 2, we also give the following conjecture regarding low ratio instances, which is analogous to the Sharp Threshold Conjecture for SAT (Friedgut and Bourgain 1999).

Conjecture 1. *For any integer $k \geq 2$, there exists a $r_k^- > 0$, such that given $F_k(n,r)$ with $r < r_k^-$, there exists an assignment violating $o(m/\log m)$ clauses with probability $1 - 2^{-\Omega(n)}$.*

Conjecture 1 is compatible with the Sharp Threshold Conjecture for SAT, thus r_k^- and r_k^+ are also conjectured to converge to the same value when k goes to infinite.

Conclusion 1. *If Conjecture 1 holds, then for large enough random k-CNF ($k > 2$) F with ratio within a certain range, there exists an assignment violating $o(m/\log m)$ clauses with high probability, so by Lemma 1 we know that this implies an optimal algorithm for $MaxSAT(F)$.*[1]

High Ratio Random k-CNF

We now turn to our main results, providing intuitions about why algorithms for Max-SAT on high ratio formulae should be different from those for SAT (and Max-SAT) on low ratio formulae, with important ramifications for the design of local search heuristics. The following lemma is a simplified version of an upper bound analysis in previous work (Achlioptas, Naor, and Peres 2007).

Lemma 3. *For any integer $k \geq 2$, given a positive constant $\lambda < \frac{1}{2^k}$, there exists a $r_k^c > 0$, such that given $F_k(n,r)$ with $r > r_k^c$, the probability of there existing an assignment violating at most λm clauses is $2^{-\Omega(n)}$.*

[1]Note that we do not consider specific algorithms but only aim to characterize the existence theoretically.

Proof. Set $s = (1-\lambda)m$ in Eq. 1 (in the proof of Lemma 2). Since $\lambda < \frac{1}{2^k}$, this sum is still dominated by the first term, which is

$$\mathbb{E}[X_{(1-\lambda)m}] \simeq 2^n \binom{m}{\lambda m} \left(\frac{2^k-1}{2^k}\right)^{(1-\lambda)m} \left(\frac{1}{2^k}\right)^{\lambda m}.$$

Using the fact that $\binom{m}{\lambda m} \sim 2^{h(\lambda)m}$ (omit the polynomial), where $h(\lambda) = -\lambda\log\lambda - (1-\lambda)\log(1-\lambda)$ is the *binary entropy function* (Cover and Thomas 2012), we obtain:

$$\mathbb{E}[X_{(1-\lambda)m}] \simeq \left(2\left(2^{h(\lambda)}\frac{2^k-1}{2^k}\left(\frac{1}{2^k-1}\right)^\lambda\right)^r\right)^n.$$

If $r > -1/\left(h(\lambda) + \lambda\log\frac{1}{2^k-1} + \log\frac{2^k-1}{2^k}\right) + \delta$ for any positive constant δ, we have:

$$Pr[X_{(1-\lambda)m} > 0] \leq \mathbb{E}[X_{(1-\lambda)m}] = 2^{-\Omega(n)}.$$

This is the probability of there existing an assignment that violates at most λm clauses, thus concluding the proof. □

Lemma 3 immediately implies the following corollary.

Corollary 1. *For any integer $k \geq 2$, given $F_k(n,r)$ with $r > 2^k \ln 2$, at least $f_k^{-1}(r)m$ clauses are violated by any assignment with probability $1 - 2^{-\Omega(n)}$, where f_k^{-1} is the inverse function of $f_k(\lambda) = -1/(h(\lambda) + \lambda\log\frac{1}{2^k-1} + \log\frac{2^k-1}{2^k})$ and h is the binary entropy function.*

In other words, for large enough high ratio random k-CNF, a constant fraction of clauses must be violated. To understand how this constant fraction of violated clauses influences algorithms for Max-SAT, we first give necessary definitions regarding local search.

Definition 1. *Given a CNF F and a complete assignment α for it, the* make value *$m(v)$ of a variable v is the number of clauses in F that will transition from unsatisfied to satisfied after flipping v under α, while the* break value *$b(v)$ of such a v is the number of clauses in F that will transition from satisfied to unsatisfied after flipping v under α.*

Definition 2. *Given a CNF F and an initial complete assignment α_0 for it, a local search algorithm $\mathcal{A}$ on F starts with step 0 and α_0, and the assignment in step t is denoted as $\alpha(\mathcal{A},t)$. We further define $\mathcal{C}_{\mathrm{U}}(F,t) = \{c \mid$ clause c violated in F under $\alpha(\mathcal{A},t)\}$, and if the context is clear, $\mathcal{C}_{\mathrm{U}}$ denotes the set of all the currently violated clauses.*

Let the *Hamming Distance* d be the number of bits on which the current assignment α disagrees with the optimal assignment α^* (violating a minimal number of clauses). An optimal solution is reached if $d = 0$. Note that if there are multiple optimal solutions, this process could terminate earlier, but does not influence our analysis. We evaluate the influence of the make value on d probabilistically.

For any variable v with make value $m(v)$, flipping v will satisfy $m(v)$ clauses. For every such clause c, c is violated under α^* with probability λ (Corollary 1), and thus d increases by 1 after flipping v. Another case is more interesting: with probability $1-\lambda$, c is satisfied under α^*, so with probability

at least $\frac{1}{k}$, d decreases by 1 because we satisfy the right literal, and with probability at most $1 - \frac{1}{k}$, we set the wrong literal to true and increase d by 1. Simple calculation shows that:

$$Pr[d \to d+1] \leq 1 - \left(\frac{1-\lambda}{k}\right)^{m(v)} \tag{2}$$

$$Pr[d \to d-1] \geq 1 - \left(1 - \frac{1-\lambda}{k}\right)^{m(v)} \tag{3}$$

Let us now consider a random walk on a CNF formula, i.e., a Markov Chain with absorbing state $d = 0$ (Schöning 1999). Decreasing the expectation of steps of a random walk from the initial state to state $d = 0$ requires reducing the right side of (2) or enlarging the right side of (3). Unfortunately this cannot be done at the same time since both of the right sides increase when $m(v)$ increases. It has empirically been shown that considering only the break value is highly preferable in SAT algorithms (McAllester, Selman, and Kautz 1997). Further studies have observed that taking the make value into account hurts the performance of local search for SAT (Balint and Schöning 2012). We provide the following theoretical analysis.

Explanation. Random walk guarantees that $m(v) \geq 1$ by always choosing variables from a violated clause. If we take the make value into account, comparing to $m(v) = 1$, the upper bound of $Pr[d \to d+1]$ increases by $\mathcal{U}(\lambda) = \frac{1-\lambda}{k} - (\frac{1-\lambda}{k})^{m(v)}$, while the lower bound of $Pr[d \to d-1]$ increases by $\mathcal{L}(\lambda) = 1 - \frac{1-\lambda}{k} - (1 - \frac{1-\lambda}{k})^{m(v)}$. Define the cost function as $g(\lambda) = \mathcal{U}(\lambda)/\mathcal{L}(\lambda)$ $(m(v) > 1)$, which essentially means increasing the lower bound of $Pr[d \to d-1]$ by x will cost an $x \cdot g(\lambda)$ increment in the upper bound of $Pr[d \to d+1]$.

Here is our key observation: for $k \geq 2$ and $0 \leq \lambda < 1$, $g(\lambda)$ is a strictly decreasing function for $m(v) > 2$, and constant for $m(v) = 2$. Recall that SAT is the special case of $\lambda = 0$. We have that Max-SAT with positive λ has a smaller cost. In other words, for the same improvement on the lower bound of $Pr[d \to d-1]$, Max-SAT costs less with respect to increments in the upper bound of $Pr[d \to d+1]$. Hence, flipping variables with higher make values does not hurt $Pr[d \to d+1]$ as seriously as for SAT.

Conclusion 2. *For sufficiently large random k-CNF F ($k \geq 2$) with high ratio, local search algorithms for $MaxSAT(F)$ should more likely consider make values than algorithms for low ratio random k-CNF. Moreover, since higher r imply higher λ (Corollary 1) and thus smaller $g(\lambda)$, more weight should be given to variables with high make values.*

4 ProMS Algorithm

Our analysis leads us to believe that for Max-SAT, the make value, as well, is a crucial piece of information about a variable. In other words, if $f(v)$ is a scoring function for choosing variables v to flip, an ideal $f(v)$ should be of the form $f(v) = g(m(v), b(v))$, i.e., depend on both the make and break values of a variable v. Moreover, Conclusion 2 suggests that for higher ratio formulae, variables with high make values should be afforded a higher probability of being flipped, so $f(v)$ may take the form $f(v) = g(\zeta(m(v), r), b(v))$, where $\zeta(m(v), r)$ is an increasing function on r with $m(v)$ fixed.

Algorithm 1: ProMS

Input: CNF-formula F, max. steps M
Output: An assignment α^* of F

```
generate a random assignment α, α* ← α
for step ← 1 to M do
    c ← pickClause(C_U(F, α))        ▷ random violated clause
    τ ← Σ_{v∈c} f(v)
    if τ > δ then
        foreach v ∈ c do
            choose v and break the loop with probability f(v)/τ
    else
        v ← a variable in c chosen at random
    α ← α with v flipped
    if |C_U(F, α)| < |C_U(F, α*)| then
        α* ← α
return α*
```

Our ProMS algorithm (Algorithm 1) first randomly generates a complete assignment, and then repeatedly picks a variable and flips it, for up to a maximal number of steps M. In each step, once a clause has been selected, the incident variables are chosen with probability p according to a distribution function f. We then update the current assignment. If the number of unsatisfied clauses is now lower than for the previous best assignment α^*, we update α^* to be the current one. Ultimately, the best found assignment is returned.

Variable Selection Probabilities

Following our earlier analysis, we select variables v for flipping based on both the make values $m(v)$ and the break values $b(v)$, while $m(v)$ is not used in algorithms like WalkSAT and probSAT for the SAT problem. The polynomial form for $f(v)$ has been proved to be an appropriate choice both in theory and practice (Liu and Papakonstantinou 2016), so we define $f(v) = m(v)^{\zeta}(1+b(v))^{\eta}$, where $\zeta = \zeta(r)$ is an increasing function of ratio r. Our algorithm picks variables v with probability

$$p(v) = \begin{cases} \frac{f(v)}{\tau(c)} & \tau(c) \geq \delta \\ \frac{1}{|c|} & \text{otherwise,} \end{cases} \tag{4}$$

where $\tau(c) = \sum_{v \in c} f(v)$ denotes the score of a clause c and δ is a threshold parameter.

Explanation. When $\tau(c)$ is very low, which implies that all incident variables have high break values or low make values and thus there are no promising variables, every variable within the clause is chosen with equal probability. Such a purely random selection is also used for diversification in dynamic local search (Li and Huang 2005). Due to the influence of $m(v)$ on $f(v)$, our algorithm could fall into local optima much faster than with a break-only function. Thus the purely random mode serves to neutralize excessive greediness and prevent our algorithm from performing poorly. When $\tau(c)$ is above the threshold, every variable is allowed to flip with

a probability greater than 0 (note that $m(v)$ is always positive because we choose variables from an unsatisfied clause), while in WalkSAT, some flips are forbidden when 0-break variables exist. Specific choices for the three parameters η, ζ and δ are given in Section 5.

5 Experiments

We now describe our experiments to assess ProMS and compare it with the winners of recent Max-SAT Evaluations.

Parameters. Our approach has thee parameters: η, ζ and δ. In order to tune these, we use the benchmark data from the Max-SAT Evaluation 2012[2], because for all of these instances optimal solutions are available[3]. Based on a grid search, we set the parameters as $\eta = -2.5$, $\zeta = r + 17.5$, and $\delta = 0.4 \cdot r - 1.4$, where r is the ratio. Note that the ζ parameter means that for formula with larger ratio, more preference is given to variables according to their make values, which is in line with Conclusion 2.

Experimental Setup

Benchmarks. We use all of the random 3-CNF[4] instances from the Max-SAT Evaluation 2016 (244 in total), with particularly large ratios ranging from 7.5 to 21.5 to evaluate the robustness of our algorithm. Such high ratio benchmarks are well-suited for an empirical confirmation of our theory, because low ratio formulae do not aid in distinguishing the performance of different solvers in the Max-SAT Evaluation. The instances are evaluated 20 times each.

Baselines. We compared ProMS with four state-of-the-art SLS solvers, all using the optimal parameters suggested in the referenced literature below.

- **probSAT:** We use probSAT, downloaded from EDACC[5], which was the best-performing system in the SAT Competition 2013 "Sequential Random SAT" track and the SAT Competition 2014 "Parallel Random SAT" track. Parameters are tuned based on the Max-SAT Evaluation 2012, on the same tuning set as for ProMS.
- **MaxWalkSAT:** A version of WalkSAT for Max-SAT, obtained from its homepage[6].
- **iraNovelty++:** The second place in the Max-SAT Evaluation 2013 "Unweighted Random" track. We use the latest binary, provided by its author (Abramé and Habet 2012).
- **CCLS:** CCLS (Luo et al. 2015) placed first in the Incomplete Solvers track of the Max-SAT Evaluation 2015 "Unweighted Random" track. We use the binary submitted to the Max-SAT Evaluation 2015, since the solver for 2016 is not available for download.

[2] http://www.maxsat.udl.cat/12/benchmarks/index.html

[3] http://www.maxsat.udl.cat/12/detailed/ms-random-incomplete-table.html

[4] Random 2-CNF instances in the Max-SAT Evaluation 2016 are too trivial to report, cf. http://maxsat.ia.udl.cat/detailed/incomplete-ms-random-table.html

[5] http://satcompetition.org/edacc/sc14/experiment/24/solver-configurations/1559

[6] http://www.cs.rochester.edu/u/kautz/walksat/

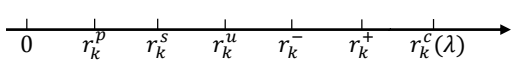

Figure 1: For random k-CNF, with high probability, ratios below r_k^p have a polynomial decidable algorithm; ratios below r_k^s are satisfiable; ratios below r_k^- have $o(m/\log m)$ violated clauses; ratios beyond r_k^u are unsatisfiable; ratios beyond r_k^+ have $\Omega(m/\log m)$ violated clauses; ratios beyond $r_k^c(\lambda)$ have λm violated clauses.

Configurations. ProMS is implemented in C, and compiled with gcc using the "-O3" option for optimization. The cutoff time is set to 300 seconds for all instances and all solvers. All experiments are carried out on a machine with Intel Core Xeon E5-2650 2.60GHz CPU and 32GB RAM under Linux.

Evaluation Methodology

We define the best solution for each instance as the minimal solution found by any of the solvers over all runs. Runs that output the best solution are regarded as successful. We also define the optimal solution for each solver found for each instance as the minimal one among all runs of the solver on that instance. We report the average time ("time") over successful runs, the average optimal solution ("opt."), and the average solution ("avg.") over all runs for each class. A hyphen in the "time" columns indicates that a solver failed to deliver a minimal solution in any run. Please refer to the Max-SAT Evaluation website for further details regarding the methodology.[7]

Results

The experimental results are given in Table 1. CCLS is based on Configuration Checking and a dynamic clause weighting scheme. Although the latter can be regarded as a form of exploiting the make value, CCLS fails to compete with ProMS, particularly on high ratio instances. MaxWalkSAT, in contrast, is based on WalkSAT and only considers the break value. Even with parameters tuned specifically for Max-SAT, it lags far behind CCLS and ProMS. This indicates that the make value plays a key role for Max-SAT.

Interestingly, probSAT, which pursues a similar strategy to ProMS, but neglects make values, turns out to be among the weakest of all approaches. Recall that probSAT and its variants represent the state-of-the-art for random SAT. This confirms our conjecture that algorithms for Max-SAT and SAT need to differ in heuristics that they consider.

6 Conclusions

Should algorithms for random SAT and Max-SAT be different? We have attempted to approach this question both theoretically and empirically. The relevant theory (including conjectures) can be summarized by Figure 1, where r_k^- and r_k^+ are the thresholds proposed in this work. This suggests that for formulae with ratio in the range $[r_k^p, r_k^-]$, there is no

[7] http://maxsat.ia.udl.cat/detailed/incomplete-ms-random-table.html

Instance Class	CCLS		iraNovelty++		MaxWalkSAT		probSAT		ProMS	
	opt. avg.	time	opt. avg.	time	opt. avg.	time	opt. avg.	time	opt. avg.	time
v70c700	22.8 23.0	2.3	23.0 23.4	16.8	23.0 23.0	20.8	23.2 23.4	4.0	**22.8** **22.8**	**1.2**
v70c800	30.2 30.4	2.1	30.4 31.0	35.9	30.6 31.0	38.8	31.8 32.4	22.1	**30.2** **30.4**	**1.8**
v70c900	39.2 39.4	3.9	39.4 40.0	10.1	40.0 41.0	103.5	40.2 41.4	88.3	**39.0** **39.4**	**2.0**
v70c1000	45.2 45.4	2.9	45.6 45.6	21.1	45.8 46.1	230.9	45.8 46.6	190.3	**44.8** **44.8**	**2.7**
v70c1100	**53.8** **53.8**	**1.9**	54.2 54.4	220.8	54.0 54.6	105.5	54.2 54.4	64.3	53.8 54.0	1.3
v70c1200	64.2 64.4	1.8	65.0 65.9	76.3	65.0 65.5	65.0	65.2 66.0	33.4	**64.0** **64.2**	**1.5**
v70c1300	71.4 71.6	1.5	71.6 72.0	10.2	72.0 72.8	143.7	72.2 72.4	98.3	**71.2** **71.4**	**1.5**
v70c1400	79.6 79.6	2.9	79.8 80.0	33.2	79.8 80.2	89.3	80.2 80.6	103.2	**79.4** **79.6**	**2.3**
v70c1500	90.0 90.2	1.8	91.2 91.8	45.4	90.8 91.4	-	91.2 92.0	-	**89.8** **90.0**	**3.7**
v80c600	13.4 13.5	2.9	13.5 13.5	8.5	13.5 13.8	99.3	13.6 13.7	105.4	**13.4** **13.5**	**1.9**
v80c700	18.8 18.8	3.0	19.2 19.8	15.3	19.52 19.8	319.3	19.4 19.7	404.6	**18.7** **18.8**	**2.0**
v80c800	27.3 27.4	8.3	27.5 27.8	26.3	27.4 27.8	155.9	27.5 27.7	64.2	**27.3** **27.3**	**1.5**
v80c900	34.2 34.2	1.9	34.4 34.8	45.9	34.4 34.6	30.5	34.5 34.6	24.4	**34.1** **34.2**	**2.0**
v80c1000	41.0 41.1	3.5	41.2 41.3	33.0	41.2 41.3	232.9	41.2 41.2	14.0	**40.9** **41.0**	**1.9**
v90c700	**16.9** **17.1**	**1.9**	17.0 17.5	33.6	17.1 17.8	260.0	17.1 17.4	34.2	16.9 17.1	2.1
v90c800	23.3 23.5	4.2	23.3 23.6	66.7	23.5 23.8	210.0	23.5 24.0	-	**23.1** **23.3**	**4.5**
v90c900	28.3 28.5	8.3	28.5 28.8	98.9	28.6 28.8	450.3	28.4 28.6	312.2	**28.2** **28.3**	**3.5**
v90c1000	37.9 37.9	3.5	38.4 38.8	119.3	38.5 40.0	-	38.4 40.1	-	**37.8** **37.8**	**2.9**
v90c1100	45.3 45.4	5.8	45.6 45.9	46.2	45.8 50.1	-	46.0 49.5	-	**45.1** **45.2**	**14.2**
v90c1200	53.6 53.7	6.4	53.9 54.4	33.1	54.0 54.3	-	54.1 54.4	-	**53.5** **53.7**	**8.3**
v90c1300	61.6 62.1	3.1	61.9 62.8	102.7	62.0 62.8	-	61.8 62.7	332.4	**61.4** **61.6**	**22.1**
v110c700	10.4 10.6	8.2	10.5 10.7	40.8	10.5 10.8	117.0	10.4 10.8	51.6	**10.3** **10.5**	**11.4**
v110c800	16.5 16.6	3.1	16.6 16.7	56.4	16.7 16.8	100.4	16.7 16.9	44.8	**16.5** **16.6**	**2.4**
v110c900	21.6 21.7	4.1	21.8 22.0	30.3	21.9 22.3	133.2	21.8 22.0	71.0	**21.5** **21.7**	**9.4**
v110c1000	44.9 45.1	1.2	45.0 45.3	38.5	45.5 45.8	130.4	45.6 45.9	-	**44.8** **45.0**	**1.2**
v110c1100	37.0 37.4	1.9	37.8 38.0	313.9	38.0 38.3	-	38.1 38.5	-	**36.9** **37.0**	**22.5**

Table 1: Each line represents a class of instances, with the number of variables and clauses designated in its name, containing many instances of the same size. The best performance on each class is in **bold**, defined as the solver with minimal "opt.", breaking ties with smaller "avg." and then "time". MaxWalkSAT and probSAT are unable to find the best solution for all instances, while iraNovelty++ finds solutions with better quality but also substantially longer average running time than others. ProMS gives the solutions with best quality on all classes except *v70c1100*, and on *v90c700* it outputs the same solution with 0.2s slowdown in running time comparing to CCLS.

need to distinguish between SAT and Max-SAT algorithms, while for ratios higher than r_k^+, the nature of the two problems is different. More specifically, our work suggests that under the random walk framework, not only break values but also make values are to be considered when choosing variables to flip, and more preference should be given to the latter when the ratio goes up.

To back up our findings and as an additional contribution, we have presented a novel algorithm for Max-SAT called ProMS. Unlike most previous Max-SAT approaches, ProMS eschews a greedy strategy in favor of a more probabilistic one. Contrasting between algorithms with and without make value, as well between ProMS and probSAT, our empirical findings confirm the value of exploiting make values and corroborate our theoretical conclusion that algorithms for SAT and Max-SAT should differ for high ratio formulae.

Regarding future work, note that the gap between r_k^s and r_k^u never closes for constant k, so exploring a rigorous lower bound for r_k^- and its gap to r_k^+ for small k will be an intriguing direction. Further, new theory based on a deeper understanding of the geometry of the solution space may also aid in analyzing and improving the design of heuristics. From a practical perspective, having found that ProMS outperforms the winners in Max-SAT Evaluations leads us to consider self-adaptive parameters and clause weighting schemes to apply our ideas to further kinds of Max-SAT instances. Another interesting point is the strong randomness in the clause and variable selection, which brings considerable variance. Thus engineering a Max-SAT solver incorporating the ProMS algorithm with multiple threads would achieve much better results in practice.

7 Acknowledgments

This research was done when the first author was in IIIS, Tsinghua University and in Microsoft Research, Redmond.

References

Abramé, A., and Habet, D. 2012. Inference rules in local search for max-sat. In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012*, 207–214.

Achlioptas, D., and Menchaca-Mendez, R. 2012. Exponential lower bounds for DPLL algorithms on satisfiable random 3-cnf formulas. In *Theory and Applications of Satisfiability Testing - SAT 2012*, 327–340.

Achlioptas, D., and Moore, C. 2006. Random k-sat: Two moments suffice to cross a sharp threshold. *SIAM J. Comput.* 36(3):740–762.

Achlioptas, D.; Naor, A.; and Peres, Y. 2007. On the maximum satisfiability of random formulas. *J. ACM* 54(2).

Alekhnovich, M., and Ben-Sasson, E. 2007. Linear upper bounds for random walk on small density random 3-cnfs. *SIAM J. Comput.* 36(5):1248–1263.

Balint, A., and Schöning, U. 2012. Choosing probability distributions for stochastic local search and the role of make versus break. In *Theory and Applications of Satisfiability Testing*, SAT 2012. 16–29.

Cai, S., and Su, K. 2012. Configuration checking with aspiration in local search for SAT. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI 2012, 334–340.

Chen, R., and Santhanam, R. 2015. Improved algorithms for sparse MAX-SAT and max-k-csp. In *Theory and Applications of Satisfiability Testing*, SAT 2015, 33–45.

Coja-Oghlan, A., and Frieze, A. M. 2014. Analyzing walksat on random formulas. *SIAM J. Comput.* 43(4):1456–1485.

Cover, T. M., and Thomas, J. A. 2012. *Elements of information theory*.

Friedgut, E., and Bourgain, J. 1999. Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American mathematical Society* 12(4):1017–1054.

Håstad, J. 2001. Some optimal inapproximability results. *Journal of the ACM (JACM)* 48(4):798–859.

Hoos, H. H., and Stützle, T. 2004. *Stochastic Local Search: Foundations & Applications*.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* 63(4):512–530.

Li, C. M., and Huang, W. Q. 2005. Diversification and determinism in local search for satisfiability. In *Theory and Applications of Satisfiability Testing*, SAT 2005, 158–172.

Liu, S., and Papakonstantinou, P. A. 2016. Local search for hard sat formulas: the strength of the polynomial law. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, AAAI 2016, 732–738.

Luo, C.; Cai, S.; Wu, W.; Jie, Z.; and Su, K. 2015. CCLS: an efficient local search algorithm for weighted maximum satisfiability. *IEEE Trans. Computers* 64(7):1830–1843.

McAllester, D. A.; Selman, B.; and Kautz, H. A. 1997. Evidence for invariants in local search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI 97, 321–326.

Schöning, U. 1999. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science*, FOCS 1999, 410–414.

Selman, B.; Kautz, H.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI 1994, 337–343.

Selman, B.; Levesque, H.; and Levesque, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI 1992, 440–446.

Smyth, K.; Hoos, H. H.; and Stützle, T. 2003. Iterated robust tabu search for MAX-SAT. In *Advances in Artificial Intelligence*. 129–144.

Williams, R. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348(2-3):357–365.