Research Reports on Mathematical and Computing Sciences

Random Instance Generation for MAX 3SAT

Mitsuo Motoki

August 2001, C–153

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Abstract

For Maximum 3-Satisfiability Problem (MAX 3SAT), we discuss about random test instance generation.

First, we analyze the expected number of unsatisfiable clauses on the optimal assignment. We prove two lower bounds theoretically. One is $\Omega(m/\ln m)$ for $m \geq 5.19n$ and the other is $m/8 - O(\sqrt{mn})$ for $m \geq 9.7n$ where n is the number of variables and m is the number of clauses. We also compare these theoretical lower bounds with experimental results.

Next, we propose a simple random test instance generator that takes the number of variables n and the minimum number of unsatisfiable clauses u as inputs and outputs an appropriate size test instance and its optimal solution with high probability. For this algorithm, we analyze the number of clauses that is sufficient to output its optimal solution with high probability. Theoretically, we prove the upper bound $8u + \Omega(n \ln n) + \Omega(n\sqrt{u \ln n})$.

1 Introduction

MAX SAT is one of famous combinatorial optimization problems stated as follows: given a multiset of clauses, find an assignment that maximizes the number of satisfied clauses (that is equivalent to finding an assignment that minimizes the number of unsatisfied clauses). MAX 3SAT is restricted version of MAX SAT, that is, its input is restricted to a multiset of 3-clauses, i.e., each clause in multiset contains exactly 3 literals whose underlying variables are distinct each other. Since these problems are not only NP-hard problem, but MAX SNP-complete problem even if the size of each clause is restricted to at most 2, there is no polynomial time approximation algorithm whose approximation ratio is close to 1 unless P = NP. Furthermore, Håstad showed that it is NP-hard to approximate MAX 3SAT within $8/7 - \varepsilon$ for any $\varepsilon > 0$ [8]. In spite of these negative results, many polynomial time approximation algorithms with proven approximation ratio for MAX SAT have been proposed [6, 7, 11]. On the other hand, some researchers are also interested in exact algorithms [3, 10, 13].

To evaluate these approximation and exact algorithms experimentally, test instances are necessary. While there are some benchmark problems for MAX SAT, we still do not have enough number of problems, and we would like to have sure way generating nontrivial test instance systematically. In particular, it would be nice if we have an efficient test instance generators for MAX SAT.

Test instance generator have been studied for NP problems (see, e.g. [1, 2, 9, 12]). On the other hand, for optimization problems like MAX SAT, no generator, as far as the author knows, has been proposed in the literature.

When one evaluates an approximation algorithm, test instances should have their optimal solution otherwise we can not determine how good approximated solution are obtained. In spite of necessity of optimal solution, it seems hard to check whether given a (candidate of) optimal solution is really optimal. In fact, in the case of MAX SAT, it is co-NP complete to check the optimality of a given solution.

Since this problem, i.e., the hardness of generating a test instance with its optimal solution, seems unavoidable, we consider two alternative approaches. The first one is to estimate (analytically and experimentally) the expected number $E[s(F)]$ of optimally satisfied clauses of F, when MAX SAT instance F is generated under a certain distribution. Suppose that we can estimate $E[s(F)]$ with reasonable precision. Then even if we can not supply optimal solutions, we could still test the average performance of a given MAX SAT algorithm. That is to say, we just run it on generated instances and compare the average number of clauses satisfied by obtained solution with $E[s(F)]$. The second approach is to design a test instance generator that provides both an instance and its candidate of optimal solution that is really optimal with high probability.

In this paper, we obtain some basic results for these two approaches. For the first one, we analyze the minimum number $u(F)$ of unsatisfiable clauses of given formula F. We obtain two lower bounds on its expectation $E[u(F)]$, $\Omega(m/\ln m)$ and $m/8-O(\sqrt{mn})$, for random 3CNF formula with m clauses over n variables. We also analyze the expectation experimentally. For the second approach, on the other hand, we propose a simple random instance generator that, for given u, generates an instance and an assignment (i.e., a candidate for the optimal solution) that leaves just u clauses unsatisfied. Furthermore, we prove that the generated solution is indeed optimal with high probability if the number of clauses is $\Omega(n \ln n) + \Omega(n\sqrt{u \ln n})$.

Preliminaries Hereafter, we describe basic notations. Let $v_1, \ldots, v_n$ be propositional variables that take one of two values, *true* or *false*. We represent true and false by 1 and 0 respectively. An assignment t is a mapping from variables to Boolean values, $v_1, \ldots, v_n \to \{0,1\}^n$. We denote by T_d a set of assignments whose Hamming distance from 1^n is d.

A *literal* is a propositional variable v (*positive* literal) or its *negation* $\overline{v}$ (*negative* literal). We use a complement-free set of literals to denote a *clause*, a disjunction of literals such as $(v_i \vee \overline{v_j} \vee v_k)$. We denote by $\perp$ an empty clause $\{\}$ which is always unsatisfied. We also use $\top$ to denote a special clause which is always satisfied.

Let F be a multiset of clauses, and we assume that $|F| = m$. We say F is *3CNF* if each clause consists from exactly 3 literals. For any F and t, we denote the number of unsatisfied clauses in F at an assignment t by $u(F,t)$. Furthermore, $u(F)$ is the minimum number of unsatisfied clauses in F for all assignments, that is, $u(F) = \min_{t \in \{0,1\}^n} u(F,t)$. We say $u(F)$ is the *optimal value* of F. We use the number of unsatisfied clauses instead of the number of satisfied clauses which is commonly used.

We say F is a *random 3CNF formula* with m clauses over n variables if each clause in F is chosen from possible $8\binom{n}{3}$ clauses over n variables uniformly at random.

2 The Average Optimal Value for Random 3CNF Formula

We consider the expected number $E[u(F)]$ of unsatisfiable clauses on the optimal solution. On the expected maximum number of satisfiable clauses, Broder et al. proved, by using Martingale technique, the probability that the difference from the expected maximum number of satisfiable clause is larger than $\sqrt{2m \log m}$ is less than $1/m$ where m is the total number of clauses [4]. Furthermore, there exists an obvious upper bound $m/8$ since it is the upper bound of $u(F)$. Here we prove two lower bounds of $E[u(F)]$.

First, we prove the lower bound in a naive way.

Theorem 1. *For any $m \geq n \ln 2/\ln(8/7) \approx 5.19n$, the expected number of unsatisfiable clauses on optimal assignment satisfies the following bound,*

$$E[u(F)] \geq \frac{m \ln 8/7 - n \ln 2 + \ln(m/7 - 1)}{\ln m/7}(1 - 7/m).$$

Proof. For any assignment t, the probability $\Pr[u(F,t) = v]$ that the number of unsatisfiable clauses in F on t is exactly v is estimated at $\binom{m}{v}\left\{\frac{1}{8}\right\}^v\left\{\frac{7}{8}\right\}^{m-v}$ (through out this section, the probability is taken under the random choice of F). Thus we can bound the probability that the number of unsatisfiable clauses in F on its optimal assignment is larger than v as follows.

$$\begin{aligned}
\Pr[u(F) \geq v] &\geq 1 - \sum_{i=0}^{v-1} \Pr\left[\bigvee_t u(F,t) = i\right] \\
&\geq 1 - 2^n(7/8)^m \sum_{i=0}^{v-1} \binom{m}{i} 7^{-i} \\
&\geq 1 - 2^n(7/8)^m \frac{(m/7)^v}{m/7 - 1}.
\end{aligned}$$

Because of $m \geq \frac{n \ln 2}{\ln 8/7}$, we have $2^n(7/8)^m \leq 1$, that is, the bound of $\Pr[u(F) \geq v]$ is larger than 0 when $v \leq \frac{m \ln 8/7 - n \ln 2 + \ln(m/7-1)}{\ln m/7}$. By using Markov's inequality $\Pr[X \geq a] \leq \mathrm{E}[X]/a$, we obtain

$$\mathrm{E}[u(F)] \geq v\left\{1 - 2^n(7/8)^m \frac{(m/7)^v}{m/7 - 1}\right\}.$$

We set v to $\frac{m \ln 8/7 - n \ln 2 + \ln(m/7-1)}{\ln m/7} - 1$, which complete the proof. □

In the Theorem 1, rounding $\binom{m}{v}$ by m^v causes the denominator of upper bound, $\ln m/7$. Thus we need to estimate the probability that $u(F)$ is larger than v more tightly. The probability $\Pr[u(F) \geq v]$ includes partial sum of binomial distribution. For sufficiently large m, we can approximate binomial distribution to normal distribution by using central limit theorem. This observation leads us to the next result.

Theorem 2. *For any* $m \geq 7n\ln 2 + \sqrt{7n\ln 2 - 16}\sqrt{7n\ln 2} - 8 \approx 9.7n - 8$*, the expected number* $E[u(F)]$ *of optimally unsatisfiable clauses satisfies following bound,*

$$E[u(F)] \geq \frac{1}{8}\left(m - \sqrt{14mn\ln 2} + 8\right)\left(1 - \frac{4}{\sqrt{7\pi nm\ln 2}}\right).$$

Proof. We already have $\Pr[u(F) \geq v] \geq 1 - 2^n \sum_{i=0}^{v-1} \binom{m}{i}(1/8)^i(7/8)^{m-i}$ in the proof of Theorem 1. This probability includes partial sum of binomial distribution. For sufficiently large m, we can approximate binomial distribution to normal distribution by using central limit theorem. Thus

$$\begin{aligned}
\Pr[u(F) \geq v] &\geq 1 - 2^n \sum_{i=0}^{v-1} \binom{m}{i} \left\{\frac{1}{8}\right\}^i \left\{\frac{7}{8}\right\}^{m-i} \\
&\approx 1 - 2^n \sum_{i=0}^{v-1} \frac{8\exp\left(-\frac{(i-m/8)^2}{14m/64}\right)}{\sqrt{2\pi}\sqrt{7m}} \\
&\geq 1 - \frac{2^{n+3}}{\sqrt{7m}}\Phi\left(-\frac{m - 8v + 8}{\sqrt{7m}}\right),
\end{aligned}$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\exp(-t^2/2)dt$. We set v to $m/8 - \sqrt{14mn\ln 2}/8 + 1$. Since m is larger than $7n\ln 2 + \sqrt{7n\ln 2}\sqrt{7n\ln 2 - 16} - 8$, this v is larger than 0, and $(m - 8v + 8)/\sqrt{7m}$ is also larger than 0. Thus applying $\Phi(-x) < \exp(-x^2/2)/\sqrt{2\pi}x$ if $x > 0$ [5], we can bound

$$\Pr[u(F) \geq v] \geq 1 - \frac{2^{n+3}\exp\left(-\frac{(m-8u+8)^2}{14m}\right)}{\sqrt{2\pi}(m - 8v + 8)}.$$

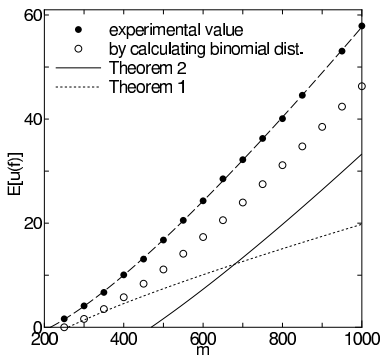
Since $v = m/8 - \sqrt{14mn\ln 2}/8 + 1$, we have $2^n \exp\left(-\frac{(m-8u+8)^2}{14m}\right) = 1$. Thus, $\Pr[u(F) \geq v] \geq 0$.

Combine all, by using Markov's inequality, we complete the proof. □

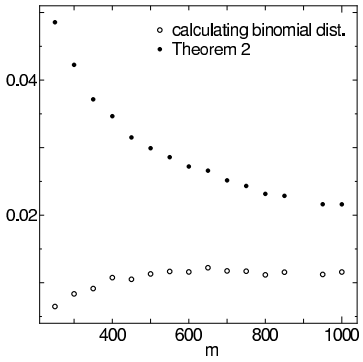
In theoretical analyses, we show two lower bounds, $\Omega(m/\ln m)$ and $m/8 - \mathcal{O}(\sqrt{mn})$, for the expected minimum number $E[u(F)]$ of unsatisfiable clauses for random 3CNF formula. We compare these theoretical lower bounds with the average number of unsatisfiable clauses on optimal assignment that we obtained through some experiments.

Now we describe the setup used in the experiments. We use an exact MAX SAT solver introduced in Appendix. We fixed the number of variables to 50, and we selected the numbers of clauses m from 250 to 1000 at intervals of 50 (we skipped 900). For each m, we checked the average optimal value to 100 (50, for over 600 clauses) random 3CNF.

We show results in Fig. 1(a) , along the horizontal axis we give the number of clauses m, and along the vertical axis we give the average number of unsatisfiable clauses on optimal. Big black dots are results of experiments, and dotted



(a) the average of $u(F)$



(b) the ratio to m of difference between the experimental and the theoretical value

Figure 1: Experimental results

line and solid line are lower bounds of Theorem 1 and Theorem 2 respectively. Furthermore, white circle are values obtained by calculating the partial sum of binomial distribution, $\arg\max_v \left(v \left(1 - \sum_{j=0}^{v} \binom{m}{v} (1/8)^v (7/8)^{m-v} \right) \right)$. With this graph, it seems that behavior of experimental results is similar to that of lower bound of Theorem 2.

In Fig. 1(b), we show the ratio of difference between the experimental and the theoretical value to m. Black dots are the ratio of the difference between the experimental value and the lower bound of Theorem 2. White circles are the ratio of the difference between the experimental value and the value that obtained by calculating the binomial distribution. We can see that the difference is as much as $0.02m$.

Thus we assume that the relation of $E[u(F)]$ with m forms $(1/8 - c_1)m - \sqrt{c_2 m} + c_3$, and get $(1/8 - 0.00620)m - \sqrt{7.13m} + 10.9$ (dashed line in Fig. 1(a)) by using the least squares.

3 Random Instance Generation for MAX 3SAT

Our final goal is to generate MAX SAT instance with its optimal solution at random. When evaluating MAX SAT solver experimentally, the optimal solution is necessary since it is hard to determine whether obtained solutions are correct. On the contrary, it is also hard to check a (candidate of) optimal solution generated with an instance. Hence, we relax the condition of instance generator such that the candidate of optimal solution is correct with high probability, that is, for given n and u, our generator generates appropriate size MAX SAT instance over n variables and an assignment (for candidate of optimal solution) that unsatisfies exactly u clauses.

To generate MAX SAT instances whose optimal value is u, we can use unique

solution SAT instances (see [2, 12] for generating unique solution SAT instances), that is, satisfiable CNF formulae with exactly one satisfying assignment. Just generate u many unique solution SAT instances such that all instances have same satisfying assignment $\hat{t}$, then combine these SAT instances and add u clauses that are not satisfied by $\hat{t}$ randomly. In this way, even if we can generate unique solution SAT instances with n clauses over n variables (we remark that it is impossible), the number of clauses of obtained MAX SAT instance is un. This is not good way, since the optimal value u of obtained instance is too small comparing with the expected optimal value for un clauses random 3CNF that is larger than $un/8 - \mathcal{O}(\sqrt{u}n)$. Thus, we have to consider other ways.

Our basic idea is as follows. Let F and $\hat{t}$ be the output instance and assignment of our MAX SAT instance generator for given n and u. Without loss of generality, we assume that 1^n is chosen as $\hat{t}$. Of course, F shall have exactly u clauses that are not satisfied by $\hat{t}$. In the case of MAX 3SAT, these clauses shall be the form of $\{\overline{v_i}, \overline{v_j}, \overline{v_k}\}$. There exist some assignments in T_1 that satisfy some of these u clauses while the number of unsatisfied clauses by 1^n is exactly u. For example, if $\{\overline{v_i}, \overline{v_j}, \overline{v_k}\}$ is contained in these u clauses, three assignments such that just one of v_i, v_j and v_k is 0 satisfy the clause $\{\overline{v_i}, \overline{v_j}, \overline{v_k}\}$. Therefore, for each of these u clauses, we add three more clauses forms of $\{v_i, \overline{v_{i_1}}, \overline{v_{i_2}}\}$, $\{v_j, \overline{v_{j_1}}, \overline{v_{j_2}}\}$ and $\{v_k, \overline{v_{k_1}}, \overline{v_{k_2}}\}$, respectively. Now each assignment in T_1 has exactly u unsatisfied clauses. Then we add appropriate number of clauses that satisfied by 1^n. Intuitively, 1^n becomes one of optimal solutions of F with high probability when enough number of clauses are added.

Now we introduce our instance generation algorithm formally. First, we define some sets of clauses C_{9-}, C_{9+}, C_5^i, C_{3-} and C_{3+} as follows:

$$
\begin{aligned}
C_{9-} &= \{\{\overline{v_{i1}}, \overline{v_{i2}}, \overline{v_{i3}}\}, \{v_{i1}, \overline{v_{i4}}, \overline{v_{i5}}\}, \{v_{i2}, \overline{v_{i6}}, \overline{v_{i7}}\}, \{v_{i1}, \overline{v_{i8}}, \overline{v_{i9}}\}\},\\
C_{9+} &= \{\{v_{i1}, v_{i2}, v_{i3}\}, \{\overline{v_{i1}}, v_{i4}, v_{i5}\}, \{\overline{v_{i2}}, v_{i6}, v_{i7}\}, \{\overline{v_{i1}}, v_{i8}, v_{i9}\}\},\\
C_5^i &= \{\{v_i, \overline{v_{i1}}, \overline{v_{i2}}\}, \{\overline{v_i}, v_{i3}, v_{i4}\}\},\\
C_{3-} &= \{v_{i1}, \overline{v_{i2}}, \overline{v_{i3}}\}, \text{and } C_{3+} = \{\overline{v_{i1}}, v_{i2}, v_{i3}\}.
\end{aligned}
$$

Hereby we show a pseudo code of our algorithm in Figure 2. In this algorithm, u many sets in the form of C_{9-} (C_{9+}, resp.) guarantee that an assignment 1^n (0^n, resp.) does not satisfy u clauses exactly and each assignment in T_1 (T_{n-1}, resp.) does not satisfy at least u clauses. On the other hand, there will exist some assignments that does not satisfy less than u clauses. In such case, $\hat{t}$ is no longer the optimal solution of F. To avoid these assignments, we add r clauses satisfied by both of 1^n and 0^n (C_{3-} and C_{3+}). We also add b many sets of $\cup_{i=1}^n C_5^i$ to balance the number of occurrence of each literal. Intuitively, the larger r makes the higher probability that $\hat{t}$ is the optimal solution of F. Hence, we analyze the condition of r such that $\hat{t}$ is the optimal solution of generated instance.

Let random variable X_t be the number of clause sets forms of C_{9-}, C_{9+}, C_5^i, C_{3-}, and C_{3+} that contain at least one clause not satisfied t. It is clear that $u(F,t)$ is at least u for any t in T_0 ($= \{1^n\}$), T_1, T_{n-1} and T_n ($= \{0^n\}$).

input: the number of variables n,
the desirable optimal value u,
the number of clause sets $\cup_{i=1}^{n} C_5^i$, b,
the number of random clauses r.
begin
let $F = \emptyset$;
repeat u times **do**
  choose 9 distinct variables, $v_{r1}, \ldots, v_{r9}$ at random;
  add $C_{9-}(v_{r1}, \ldots, v_{r9})$ to F;
  choose 9 distinct variables, $v_{r1}, \ldots, v_{r9}$ at random;
  add $C_{9+}(v_{r1}, \ldots, v_{r9})$ to F;
repeat b times **do**
  for $i = 1$ to n **do**
    choose 4 distinct variables $v_{r1}, \ldots, v_{r4}$ from $X \setminus \{v_i\}$ at random;
    add $C_5^i(v_{r1}, \ldots, v_{r4})$ to F;
repeat r times **do**
  choose 3 distinct variables v_{r1}, v_{r2}, v_{r3} at random;
  add $C_{3-}(v_{r1}, v_{r2}, v_{r3})$ to F;
  choose 3 distinct variables v_{r1}, v_{r2}, v_{r3} at random;
  add $C_{3+}(v_{r1}, v_{r2}, v_{r3})$ to F;
choose a truth assignment $\hat{t} \in \{0,1\}^n$ at random;
flip literals whose under lying variables are assigned 0 on $\hat{t}$;
output F and $\hat{t}$;
end.

Figure 2: Algorithm. We remark that $\bar{\hat{t}}$ ($= 0^n$ by assumption) is also optimal solution if $\hat{t}$ is optimal.

Thus our goal is to show the condition that X_t is at least u for any t in T_d ($2 \leq d \leq n-2$) with high probability (we remark that X_t is less than $u(F,t)$).

First we consider μ_d the expectation of X_t on arbitrary assignments t in T_d ($0 \leq d \leq n$). Clearly, μ_1 and μ_{n-1} are larger than u and $\mu_0 = \mu_n = u$.

We denote by $uC_{9-}(i,u)$ ($uC_{9+}(i,u)$) the number of assignments over 9 variables used in C_{9-} (resp., C_{9+}) such that Hamming distance from 1^9 is i and exactly u clauses in C_{9-} (C_{9+}, resp.) are unsatisfied. We also use $uC_{9-}(n,d,u)$ ($uC_{9+}(n,d,u)$) to denote the number of assignments over n variables whose Hamming distance from 1^n is d and that unsatisfy exactly u clauses in C_{9-} (C_{9+}, resp.).

Lemma 1. *For any d ($0 \leq d \leq n$), for any assignment $t \in T_d$, let μ_d be the expected number of clause sets C_{9-}, C_{9+}, C_5, C_{3-} and C_{3+} in F such that each set contains at least one clause not satisfied by t (the expectation of X_t). Then*

$$\mu_d \geq u - u\frac{d(n-d)}{n(n-1)} + b\frac{d(n-d)}{n-1} + r\frac{d(n-d)}{n(n-1)}.$$

	i									
u	0	1	2	3	4	5	6	7	8	9
0	0	0	6	27	60	78	65	33	9	1
1	1	9	27	50	57	45	19	3	0	0
2	0	0	3	6	9	3	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0

(a) $uC_{9-}(i,u)$

	i									
	0	1	2	3	4	5	6	7	8	9
$uC_{9-}(i,0)+uC_{9+}(i,0)$	1	9	39	92	138	138	92	39	9	1
$\binom{9}{i}+\binom{7}{i-1}$	1	10	43	105	161	161	105	43	10	1

(b) $uC_{9-}(i,0)+uC_{9+}(i,0)$

Table 1: $uC_{9-}(i,u)$ and $uC_{9-}(i,0)+uC_{9+}(i,0)$

Proof. First we count $uC_{9-}(i,u)$, the number of assignments for 9 variables used in C_{9-} that unsatisfy just u clauses in a C_{9-} where Hamming distance from 1^9 is exactly i. Of course we can count $uC_{9-}(i,u)$ by a brute force, and results are in Table 1(a). We also define $uC_{9+}(i,u)$ as well as $uC_{9-}(i,u)$ and remark that $uC_{9+}(i,u) = uC_{9-}(9-i,u)$. Furthermore, we define $uC_{9-}(n,d,u)$ as the number of assignments for n variables that unsatisfy just u clauses in C_{9-} and whose Hamming distance from 1^n is d. We have

$$uC_{9-}(n,d,u) = \sum_{i=0}^{i=9} \binom{n-9}{d-i} uC_{9-}(i,u),$$

and also $uC_{9+}(n,d,u) = uC_{9-}(n,n-d,u)$.

By using above notations, we represent the probability that $t \in T_d$ does not satisfy at least one clause in C_{9-} as following:

$$\Pr[t \text{ does not satisfy at least one clause in } C_{9-}] = 1 - \frac{uC_{9-}(n,d,0)}{\binom{n}{d}}.$$

Clearly, the probabilities that $t \in T_d$ does not satisfy C_{3-} and C_{3+} are $\frac{d(n-d)(n-d-1)}{n(n-1)(n-2)}$ and $\frac{d(d-1)(n-d)}{n(n-1)(n-2)}$ respectively. The probability that $t \in T_d$ does not satisfy at least one clause in C_5^i is as follows:

$$\Pr[t \text{ does not satisfy a clause in } C_5^i | t(x_i) = 0] = \frac{(n-d)(n-d-1)}{(n-1)(n-2)},$$
$$\Pr[t \text{ does not satisfy a clause in } C_5^i | t(x_i) = 1] = \frac{d(d-1)}{(n-1)(n-2)}.$$

Combine all together, we get

$$\mu_d = 2u - u\frac{uC_{9-}(n,d,0) + uC_{9+}(n,d,0)}{\binom{n}{d}} + b\frac{d(n-d)}{n-1} + r\frac{d(n-d)}{n(n-1)}.$$

It is easy to check that $uC_{9-}(i,u) + uC_{9+}(i,u) \leq \binom{9}{i} + \binom{7}{i-1}$ (see Table 1(b)), and thus we have

$$\begin{aligned} uC_{9-}(n,d,0) + uC_{9+}(n,d,0) &= \sum_{i=0}^{9} \binom{n-9}{d-i} (uC_{9-}(i,u) + uC_{9+}(i,u)) \\ &\leq \binom{n}{d} + \binom{n-2}{d-1}. \end{aligned}$$

This completes the proof. □

By using this lemma, we can use Chernoff bound to bound the probability that X_t is less than u and prove following theorem.

Theorem 3. *For any positive $\varepsilon < 1$, the probability that $\hat{t}$ is optimal solution of F is at least $1 - \varepsilon$ if*

$$r \geq u - bn + 4(n-1)\ln 2n - 2(n-1)\ln \varepsilon + (n-1)\sqrt{2\ln 2n - \ln \varepsilon}\sqrt{2u}.$$

Proof. We suppose that there is an assignment t that does not satisfy less than u clauses. Of course X_t, the number of clause sets forms of C_{9-}, C_{9+}, C_5^i, C_{3-}, and C_{3+} that contain at least one clause not satisfied by t, must be less than u. Thus, the probability that $\hat{t}$ is not optimal solution of F is bounded above as

$$\begin{aligned} \Pr[\hat{t} \text{ is not optimal solution of } F] &\leq \Pr\left[\bigvee_t (X_t < u)\right] \\ &\leq \sum_t \Pr[X_t < u]. \end{aligned}$$

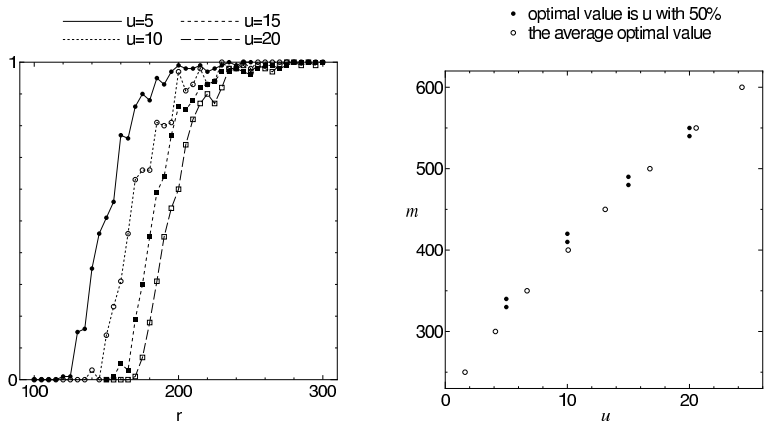
Let μ'_d be the lower bound of μ_d proven in Lemma 1. Hence each clause set is chosen independently, we can use the Chernoff bound to bound the probability that X_t is less than u, that is, for any $\sigma > 0$,

$$\begin{aligned} \Pr[X_t < (1-\delta)\mu'_d] &\leq \Pr[X_t < (1-\delta)\mu_d] \\ &< \exp\left(-\frac{\mu_d \sigma^2}{2}\right) \leq \exp\left(-\frac{\mu'_d \sigma^2}{2}\right). \end{aligned}$$

Thus by setting $\sigma = (1 - u/\mu'_d)$, we have $\Pr[X_t < u] < \exp\left(-\frac{(\mu'_d - u)^2}{2\mu'_d}\right)$.

Since $r \geq u - bn + 4(n-1)\ln 2n - 2(n-1)\ln \varepsilon + (n-1)\sqrt{2\ln 2n - \ln \varepsilon}\sqrt{2u}$, we have

$$\begin{aligned} r \geq & u - bn + 4(n-1)\ln 2n - 2(n-1)\ln \varepsilon + 2(n-1)\sqrt{\frac{\ln 2n}{d} - \frac{\ln \varepsilon}{d^2}}\sqrt{2u} \\ \geq & u - bn + \frac{2n(n-1)}{n-d}\ln 2n - \frac{2n(n-1)}{d(n-d)}\ln \varepsilon + \frac{n(n-1)}{d(n-d)}\sqrt{d\ln 2n - \ln \varepsilon}\sqrt{2u} \\ \geq & u - bn + \frac{n(n-1)}{(n-d)}\ln 2n - \frac{n(n-1)}{d(n-d)}\ln \varepsilon \\ & + \frac{n(n-1)}{d(n-d)}\sqrt{d\ln 2n - \ln \varepsilon}\sqrt{2u + d\ln 2n - \ln \varepsilon} \end{aligned}$$



(a) the ratio of instances changing optimal value u

(b) the number of clauses and optimal value u (ratio is 0.5)

Figure 3: Changing the value to be optimal u ($n = 50$, $b = 0$)

for $2 \leq d \leq n/2$. By using this r, $(\mu'_d - u)^2/2\mu'_d \geq d \ln 2n - \ln \varepsilon$. Combine all together,

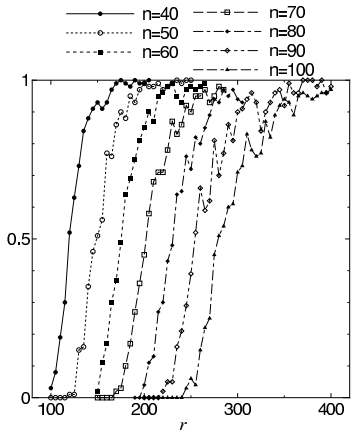
$$\begin{aligned}
\Pr[\hat{t} \text{ is not optimal solution of } F] &\leq \sum_{d=2}^{n-2} \binom{n}{d} \exp\left(-\frac{(\mu'_d - u)^2}{2\mu'_d}\right) \\
&\leq 2 \sum_{d=2}^{n/2} n^d \exp\left(-\frac{(\mu'_d - u)^2}{2\mu'_d}\right) \\
&\leq 2 \sum_{d=2}^{n/2} n^d \exp(\ln \varepsilon - d \ln 2n) \leq \varepsilon.
\end{aligned}$$

This completes the proof. □

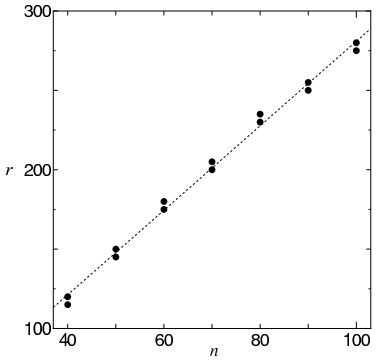
In theoretical analysis, we prove that our instance generator outputs instance with optimal value u with high probability if r is $u + \mathcal{O}(n \ln n) + \mathcal{O}(n\sqrt{u \ln n})$. But is such a large number really required? In fact, this upper bound becomes over 1300 even if $u = 5$, $n = 50$ and $\varepsilon = 1$. Thus we compare this theoretical bound with r obtained through experiments.

Since we want to determine the ratio of instance with optimal value u as exact as possible, we use exact MAX SAT solver instead of approximation algorithms. Throughout these experiments, we again use our MAX SAT solver introduced in Appendix. We remark that we can set u to the initial candidate of optimal value. Each experiment, we calculate the ratio of instances whose optimal value is u for 100 instances.

First, we estimate whether our algorithm can cover the average optimal value. We fix the number of variables n to 50 and b to 0. We select u between



(a) the ratio of instances whose optimal value is 5



(b) r for each n where 50% instances have optimal value 5

Figure 4: Changing the number of variables n ($u = 5$, $b = 0$)

5 to 20 at intervals of 5. Results are in Figure 3(a). The intervals of results are smaller as opposed to u. For each u, we pick up two values of $m = 8u + 2r$, the number of total clauses, such that one is the largest m where the ratio is less than 50% and the other is the smallest m where the ratio is larger than 50%. We compare this result with the average optimal value obtained experimentally in Section 2 (Figure 3(b)). From the figure, we can see that the number of clauses that the ratio of instance with optimal value u is just 50% is near the the number of clauses whose average optimal value is u. Thus, we consider that our algorithm generates an instance whose optimal solution is given with high probability if the optimal value is quite smaller than the average optimal value.

Second, we want to estimate whether the main term of upper bound on r is $n \ln n$ or not. We choose n from 40 to 100 at intervals of 10. And we fix u to 5 and b to 0. Results are in Figure 4(a). Then we also pickup two values of r where the ratio of instances with given optimal solution is just smaller and larger than 0.5 for each n. In Figure 4(b), the vertical axis is value of r where the ratio of instances with given optimal value is just stepping over 0.5, the horizontal axis is the number of variables n. From this result, the relation of r and n seems linear or the constant of $n \ln n$ is very small. Hence, we try to fit by linear function, and get $r = 2.6n + 14.8$ (dotted line).

Finally, we change b, the number of clause sets $\cup_{i=1}^{n} C_5^i$. In the theoretical upper bound, there exists the term $-bn$. This means that adding b sets of $\cup_{i=1}^{n} C_5^i$ is equivalent to adding bn random clauses of the each form C_{3-} and C_{3+} ($2bn$ clauses). We fix the number of variables n to 50 and the optimal value u to 5 again. We choose b from 1, 2 and 3. Results are in Figure 5, the vertical axis is the ratio of instances whose optimal value is 5 and the horizontal axis is the total number of clauses m. We can easily see that b diminishes the

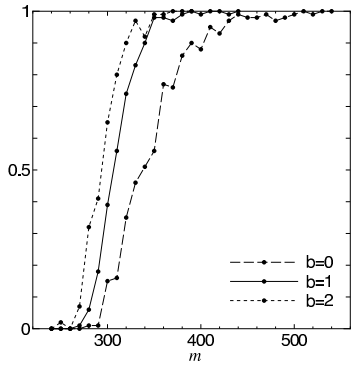


Figure 5: Changing b. ($n = 50$, $u = 5$)

total number of clauses. Furthermore, it contribute converging the ratio to 1 much faster.

4 Conclusion

By relaxing to binomial distribution, we showed two lower bounds of expected number of minimum unsatisfiable clauses for random 3CNF. Of course, these bounds are applicable to kCNF, i.e., 2CNF. On the other hand, there exist gap between these lower bounds and experimental value. Is the term $-\mathcal{O}(\sqrt{mn})$ really appropriate (for example, $-\mathcal{O}(\ln m)$)? Furthermore, it is important to check how the gap changes when changing the number of variables.

We also proposed a probabilistic MAX 3SAT instance generator. Although we showed the upper bound of number of clauses, $\mathcal{O}(n \ln n) + \mathcal{O}(n\sqrt{u \ln n})$, we believe that the term $\mathcal{O}(n \ln n)$ should be $\mathcal{O}(n)$. It is important not only to improve the probability that output assignment is the optimal solution but to modify the algorithm by rejecting such an instance.

References

[1] D. Achlioptas, C. Gomes, H. Kautz, and B. Selman. Generating satisfiable problem instnaces. *Proc. of AAAI-2000/IAAI-2000*, 2000.

[2] Y. Asahiro, K. Iwama, and E. Miyano. Random generation of test instances with controlled attributes. Johnson and Trick, eds., *Cliques, Coloring, and Satisfiability*, DIMACS vol. 26, pp. 377–393. AMS, 1996.

[3] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *J. Comb. Optim.*, 2:299–306, 1999.

[4] A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. *Proc. of SODA'93*, pp. 322–330. 1993.

[5] W. Feller. *An Introduction to Probability Theory and Applications*, volume 1. 1968.

[6] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7:656–666, 1994.

[7] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[8] J. Håstad. Some optimal inapproximability results. *Proc. of STOC'97*, pp. 1–10, 1997.

[9] S. Horie and O. Watanabe. Hard instance generation for SAT (extended abstract). *Proc. of ISAAC'97*, LNCS 1350, pp. 22–31, 1997.

[10] S. Joy, J. Mitchell, and B. Borchers. A branch and cut algorithm for MAX-SAT and weighted MAX-SAT. Du, Gu, and Pardalos, eds., *Satisfiability Problem: Theory and Applications*, DIMACS vol. 35, pp. 519–536. AMS, 1997.

[11] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? *Proc. of FOCS'97*, pp. 406–415, 1997.

[12] M. Motoki and R. Uehara. Unique solution instance generation for the 3-Satisfiability (3SAT) problem. Gent, van Maaren, and Walsh, eds., *SAT2000 Highlights of Satisfiability Research in the Year 2000*, pp. 293–307. IOS Press, 2000.

[13] R. Niedermeier and P. Rossmanith. New upper bounds for MaxSat. *Proc. of ICALP'99*, LNCS 1644, pp. 576–584. Springer, 1999.

[14] M. Yannakakis. On the approximation of maximum satisfiability. *J. Algorithms*, 17(3):475–502, 1994.

A Appendix

In this section, we mention our exact MAX SAT solver used in experimental analysis. Our algorithm is based on splitting depth first search method like Davis-Putnam procedure (see also EDPL [3]), that is, we assign Boolean value to a variable that occurs most frequently in a current formula without 1-clauses, for each node of binary search tree. We first assign 0 to v if the number of clauses containing $\overline{v}$ is larger than that of v, otherwise we assign 1 first. If the number of unsatisfied clauses exceeds the current best value, this algorithm takes backtrack. To prune the size of search tree, we apply some variable elimination rules and clause subset replacing rules for each node. In the following, $a, b, c, d, x, y, z, l_1, \ldots, l_i$ are literals whose underlying variables are all distinct and can appear any other clauses of the formula.

1. *Resolution rule*: Replace a subset of clauses according to the following:
   - $\{\{x, y, z\}, \{\overline{x}, y, z\}\}$ to $\{\top, \{y, z\}\}$,
   - $\{\{x, y\}, \{\overline{x}, y\}\}$ to $\{\top, \{y\}\}$, and
   - $\{\{x\}, \{\overline{x}\}\}$ to $\{\top, \bot\}$,
2. *Extracting satisfied clauses rule*: Replace a subset of clauses according to the following:
   - $\{\{x, \overline{y}\}, \{x, y, z\}, \{\overline{x}, \overline{y}, \overline{z}\}\}$ to $\{\top, \{x, z\}, \{\overline{y}, \overline{z}\}\}$,
   - $\{\{x, \overline{y}\}, \{y, \overline{z}\}, \{x, y, z\}\}$ to $\{\top, \{x\}, \{\overline{x}, y, \overline{z}\}\}$, and
   - $\{\{x, \overline{y}\}, \{x, \overline{z}\}, \{x, y, z\}\}$ to $\{\top, \{x\}, \{x, \overline{y}, \overline{z}\}\}$.
3. *Pure literal rule* [13], *Dominating 1-clause rule* [13] and *Extended Unit Propagation*:
   - If current formula contains a pure literal l, assign 1 to l.
   - If the number of unit clause $\{l\}$ in the current formula is larger than the total number of occurrence of literal $\overline{l}$, assign 1 to l.
   - If the sum of the number of empty clauses and the number of unit clause $\{l\}$ in the current formula is not less than current best value u_*, assign 1 to l.
4. *Complementary loop rule*: Replace a subset of clauses according to the following:
   - $\{\{x, y\}, \{x, z\}, \{\overline{y}, \overline{z}\}, \{\overline{x}, l_1\}, \{\overline{l_1}, l_2\}, \ldots, \{\overline{l_{i-1}}, l_i\}, \{\overline{l_i}\}\}$ with $\{\bot, \{x, y, z\}, \{\overline{x}, \overline{y}, \overline{z}\}, \{x, \overline{l_1}\}, \{l_1, \overline{l_2}\}, \ldots, \{l_{i-1}, \overline{l_i}\}, \top\}$,
   - $\{\{a, b\}, \{a, c\}, \{\overline{b}, d\}, \{\overline{c}, \overline{d}\}, \{\overline{a}, l_1\}, \{\overline{l_1}, l_2\}, \ldots, \{\overline{l_{i-1}}, l_i\}, \{\overline{l_i}\}\}$ with $\{\bot, \{a, b, \overline{d}\}, \{\overline{a}, \overline{b}, d\}, \{a, c, d\}, \{\overline{a}, \overline{c}, \overline{d}\}, \{a, \overline{l_1}\}, \{l_1, \overline{l_2}\}, \ldots, \{l_{i-1}, \overline{l_i}\}\}$ or $\{\bot, \{a, b, c\}, \{\overline{a}, \overline{b}, \overline{c}\}, \{\overline{b}, c, d\}, \{b, \overline{c}, \overline{d}\}, \{a, \overline{l_1}\}, \{l_1, \overline{l_2}\}, \ldots, \{l_{i-1}, \overline{l_i}\}\}$,
   - $\{\{x, y\}, \{x, z\}, \{\overline{y}, \overline{z}\}, \{\overline{x}, l_1\}, \{\overline{l_1}, l_2\}, \ldots, \{\overline{l_{i-1}}, l_i\}, \{\overline{l_i}, \overline{a}\}, \{a, b\}, \{a, c\}, \{\overline{b}, \overline{c}\}\}$ with $\{\top, \{x, y, z\}, \{\overline{x}, \overline{y}, \overline{z}\}, \{x, \overline{l_1}\}, \{l_1, \overline{l_2}\}, \ldots, \{l_{i-1}, \overline{l_i}\}, \{l_i, a\}, \{a, b, c\}, \{\overline{a}, \overline{b}, \overline{c}\}, \bot\}$,

- $\{\{x,y\},\{x,z\},\{\overline{y},\overline{z}\},\{\overline{x},l_1\},\{\overline{l_1},l_2\},\ldots,\{\overline{l_{i-1}},l_i\},\{\overline{l_i},\overline{a}\},\{a,b\},$ $\{a,c\},\{\overline{b},d\},\{\overline{c},\overline{d}\}\}$ with $\{\bot,\{x,y,z\},\{\overline{x},\overline{y},\overline{z}\},\{x,\overline{l_1}\},\{l_1,\overline{l_2}\},\ldots,\{l_{i-1},\overline{l_i}\},\{l_i,a\},\{a,b,\overline{d}\},$ $\{\overline{a},\overline{b},d\},\{a,c,d\},\{\overline{a},\overline{c},\overline{d}\}\}$ or $\{\bot,\{x,y,z\},\{\overline{x},\overline{y},\overline{z}\},\{x,\overline{l_1}\},\{l_1,\overline{l_2}\},\ldots,\{l_{i-1},\overline{l_i}\},\{l_i,a\},\{a,b,c\},$ $\{\overline{a},\overline{b},\overline{c}\},\{\overline{b},c,d\},\{b,\overline{c},\overline{d}\}\}$

5. *Symmetric flow* [14]: Replace the subset of clauses $\{\{l_1\},\{\overline{l_1},l_2\},\ldots,\{\overline{l_{i-1}},l_i\},\{\overline{l_i}\}\}$ with $\{\top,\{l_1,\overline{l_2}\},\ldots,\{l_{i-1},\overline{l_i}\},\bot\}$.
6. *3-star rule*: Replace the subset of clauses $\{\{x,y,z\},\{\overline{x}\},\{\overline{y}\},\{\overline{z},l_1\},\{\overline{l_1},l_2\},\ldots,\{\overline{l_{i-1}},l_i\},\{\overline{l_i}\}\}$ with $\{\{\overline{x},\overline{y}\},\{\overline{x},\overline{z}\},\{z,\overline{l_1}\},\{l_1,\overline{l_2}\},\ldots,\{l_{i-1},\overline{l_i}\},\bot\}$.

If we apply at least once for some rule, then we seek applicable rule from the first rule again.