An Empirical Study of MAX-2-SAT Phase Transitions ⋆

Haiou Shen [a] Hantao Zhang [a]

[a] *Computer Science Department*
The University of Iowa
Iowa City, IA 52242
{hshen, hzhang}@cs.uiowa.edu

Abstract

The decision version of the maximum satisfiability problem (MAX-SAT) is stated as follows: Given a set S of propositional clauses and an integer g, decide if there exists a truth assignment that falsifies at most g clauses in S, where g is called the *allowance* for false clauses. We conduct an extensive experiment on over a million of random instances of 2-SAT and identify statistically the relationship between g, n (number of variables) and m (number of clauses). In our experiment, we apply an efficient decision procedure based on the branch-and-bound method. The statistical data of the experiment confirm not only the "scaling window" of MAX-2-SAT discovered by Chayes, Kim and Borgs, but also the recent results of Coppersmith et al. While there is no easy-hard-easy pattern for the complexity of 2-SAT at the phase transition, we show that there is such a pattern for the decision problem of MAX-2-SAT associated with the phase transition. We also identify that the hardest problems are among those with high allowance for false clauses but low number of clauses.

1 Introduction

Given a propositional formula F in conjunctive normal form (CNF), the maximum satisfiability problem (MAX-SAT) is to find a truth assignment that satisfies the maximum number of clauses. The decision version of MAX-SAT is to ask if the number of false clauses in F under any assignment is less than than or equal to a given number, say g, the *allowance* for false clauses. It is well known that the decision version of MAX-SAT is NP-complete, even if each clause has

⋆ Partially supported by the National Science Foundation under Grant CCR-0098093.

at most two literals (so called the MAX-2-SAT problem). In recent years, there has been considerable interest in MAX-SAT [4,17,1,2,16,14,15,19,5,12,10,9]. Empirical studies can help to verify theoretical results, provide insights to theoretical analysis, and apply the techniques to other domains. However, very few of the reported research in this area involve empirical studies. This is in sharp contrast with the study of satisfiability problem (SAT), where great progress has been made in the recent years in developing and applying powerful SAT solvers.

When the allowance $g = 0$, the decision version of MAX-SAT becomes the satisfiability problem of F. It is well known that there is sharp transition in satisfiability for random 2-SAT at $m/n = 1$ [8,11], where m is the number of binary clauses and n is the number of variables. For $g > 0$, similar results are obtained very recently for MAX-2-SAT by Coppersmith et al. [9]. In this paper, we describe a detailed experimental investigation of the phase transition for random MAX-2-SAT in terms of g, m and n. We verify existing theoretical results and observe a remarkable consistency of features in the phase transition.

Because 2-SAT is solvable in linear time, there is no change of problem hardness associated with the phase transition of 2-SAT. However, since MAX-2-SAT is NP-hard, our experiment shows that there is a common easy-hard-easy pattern in the median difficulty of the MAX-2-SAT decision problem, with the hardest problems being associated with the phase transition, when the allowance is high but the number of clauses is low.

2 Preliminary

We will adopt the notations from [9]. Let F be a formula in 2–CNF with n variables $V = \{x_1, ..., x_n\}$. An *assignment* is a mapping from V to $\{0, 1\}$ and may be represented by a vector $\vec{X} \in \{0, 1\}^n$, where 0 means false and 1 means true. Let $F(\vec{X})$ be the number of clauses satisfied by $\vec{X}$. The MAX-2-SAT problem asks for $\max F = max_{\vec{X}} F(\vec{X})$, i.e., the maximum, over all assignments $\vec{X}$, of the size (number of clauses) of a maximum satisfiable subformula of F.

The decision version of MAX-2-SAT can be stated as follows:

Instance: A formula F in 2–CNF with m clauses and a nonnegative integer g.
Question: Is there an assignment $\vec{X}$ such that $m - F(\vec{X}) \leq g$?

Obviously, the phase transition of the above decision problem happens when g goes from $m - \max F - 1$ to $m - \max F$ for unsatisfiable F. That is, the

Table 1
Experimental results on Borchers and Furman's examples. Problems p100-p500 have 50 variables and problems p2200-p2400 have 100 variables. "BF" stands for Borchers and Furman's two-phase program for maxsat. "New" is the new algorithm presented in [20]. Times (in seconds) are collected on a Pentium 4 linux machine with 256M memory. "–" indicates an incomplete run after running for two hours.

Problem Name	allowance	BF	New
p100	4	0.035	0.02
p150	8	0.091	0.02
p200	16	6.425	0.12
p250	22	37	0.05
p300	32	530	0.85
p350	41	3866	1.45
p400	45	3467	0.54
p450	63	–	4.68
p500	66	–	1.78
p2200	5	0.191	0.53
p2300	15	763	7.67
p2400	29	–	172

decision procedure will return false for $g < m - \max F$ and return true when $g \geq m - \max F$.

Let $F(n, m)$ denote a random formula in 2–CNF with n variables and m clauses, where each clause is proper (consisting of two distinct literals and not a tautology); this is equivalent to choosing m clauses uniformly at random from the $t(n) = 2n(n-1)$ possible binary clauses with n variables. Typically, m is expressed as a linear function of n, $m = \lfloor cn \rfloor$ (or simply $m = cn$), where c is a constant.

Throughout this paper, we let $K(n, m) = m - \max F(n, m)$ if one instance is given; if multiple instances in $F(n, m)$ are given, $K(n, m)$ represents the arithmetic mean of individual $K(n, m)$'s.

The following results have been proved in [9].

Theorem 1 ([9]) (1) *For* $c < 1$, $K(n, cn) = \Theta(1/n)$.

(2) *For c large,* $(0.25c - 0.343859\sqrt{c} + O(1))n \succsim K(n, cn) \succsim (0.25c - 0.509833\sqrt{c})n$.

(3) *For any fixed* $\epsilon > 0$, $\frac{1}{3}\epsilon^3 n \succsim K(n, (1+\epsilon)n)$.

Table 2
Number of random 2-CNF instances used in the experiment.

n	c	min-instances/c	total instances
20	{ 0.8, 0.9, ..., 2.4, 3, 4, ..., 36 }	2000	238000
25	{ 0.8, 0.9, ..., 2.4, 3, 4, ..., 46 }	1000	214000
30	{ 0.8, 0.9, ..., 2.4, 3, 4, ..., 56 }	800	213200
35	{ 0.8, 0.9, ..., 2.4, 3, 4, ..., 50 }	400	189200
40	{ 0.8, 0.9, ..., 2.4, 3, 4, ..., 44 }	300	182600
50	{ 0.8, 0.9, ..., 2.4 }	10000	170000
70	{ 0.8, 0.9, ..., 2.4 }	10000	170000
90	{ 0.8, 0.9, ..., 2.4 }	10000	170000

In the above theorem, $\succsim$ is a standard asymptotic notation: $f(n) \succsim g(n)$ means that f is greater than or equal to g *asymptotically* — $f(n)/g(n) \geq 1$ when n goes to infinity — though it may be that $f(n) < g(n)$ even for arbitrarily large values of n [9].

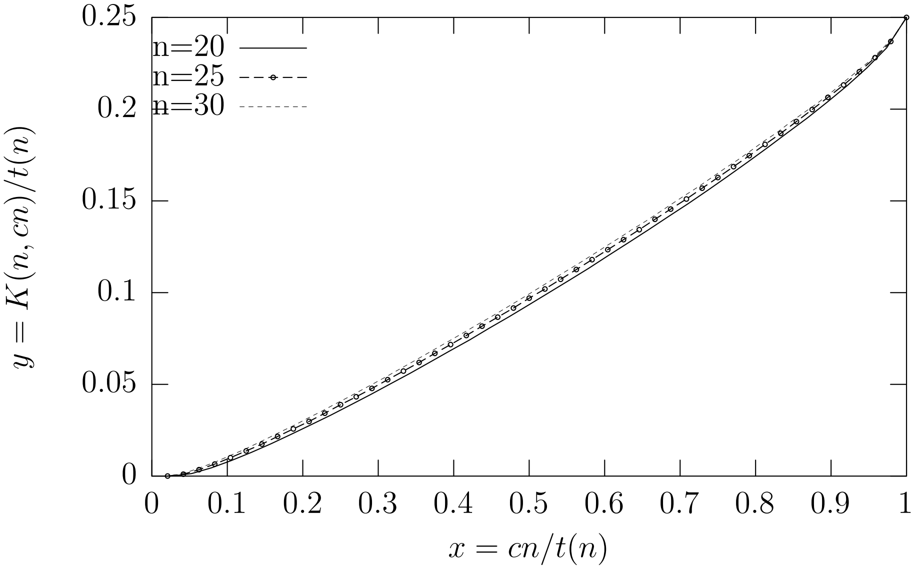
3 Experiments

Recently, we developed a new decision algorithm for MAX-2-SAT which takes a formula F in 2–CNF and an integer g as input and returns true if and only $m - F(\vec{X}) \leq g$ [20]. We proved that the algorithm takes time $O(n2^n)$ in the worst case, settling an open problem posed by Alber, Gramm and Niedermeier [18,2].

The new algorithm uses an efficient data structure for binary clauses and works as a typical branch-and-bound algorithm. The new algorithm runs substantially faster than Borchers and Furman's algorithm [6]. Table 1 shows some results of Borchers and Furman's maxsat program [6] and the new algorithm on the problems of 50 variables and some 100 variables problems distributed by Borchers and Furman. It is clear that the algorithm runs much faster than Borchers and Furman's program on all the examples except one.

The high performance algorithm allows us to run a large number of MAX-2-SAT instances. For each instance, we consider three parameters: the number n of the variables occurring in it, the number $m = cn$ of clauses for some constant c, and K, the minimal allowance for false clause under any assignment. For this study, we generated random MAX-2-SAT instances for $n = 20, 25, 30, 35, 40, 50, 70, 90$. The range of c for each n is given in Table 2, along with the minimum number of instances for each n and c.

Fig. 1. Relationship between y and x for given n.



When $m \leq n$, it is not easy to pick m random binary clauses which contain all the n variables. To create such an instance randomly, we employ the following procedure:

(1) Pick randomly m clauses and add them into S.
(2) While S does not contain all the n variables, pick randomly a new clause and add it into S.
(3) While $|S| > m$, randomly pick a clause c in S; if every variable of c appears in $S - \{c\}$, let $S = S - \{c\}$.

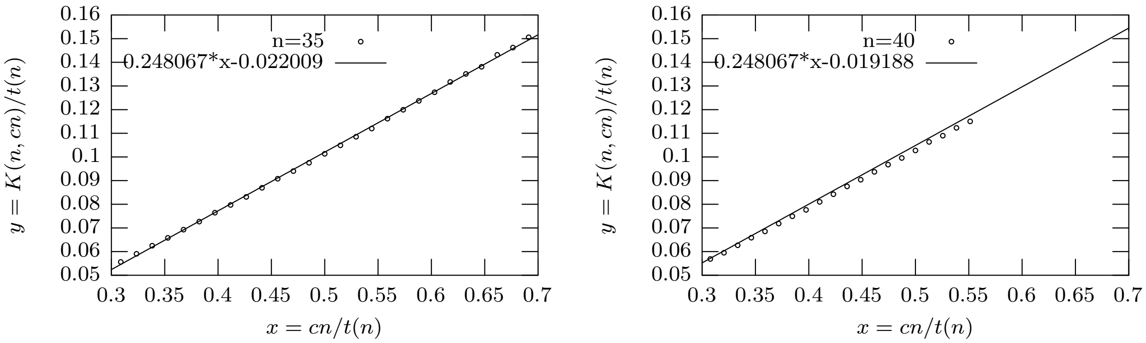
We were able to run all the instances only for $n = 20, 25$, and 30. For $n = 35$ and 40, we can only run instances for smaller c because our algorithm cannot finish the runs in a reasonable amount of time.

In order to study the relationship between $K(n, m)$ and m for a given n, we compute the arithmetic mean of $K(n, m)$ for each m. To better observe the relationship between K and m, we try to plot the data in a single figure. To do this, we scale m and K by dividing them by $t(n) = 2(n-1)n$, the total number of clauses for n variables. Now, the horizontal axis represents $x = cn/t(n)$, whose value ranges from 0 to 1, and the vertical axis represents $y = K(n, cn)/t(n)$, whose value ranges from 0 to 0.25. The results are shown in Figure 1 for $n = 20, 25, 30$.

It shows clearly that y and x have a linear relationship in most area ($0.3 \leq x \leq 0.7$). When n increases, the line moves near to the line $y = 0.25x$.

We guess that the relationship for the linear part should look something like

Fig. 2. Relationship between y and x for $(0.3 \leq x \leq 0.7)$



(a) $n = 35$ **(b)** $n = 40$

this:

$$y = ax + \frac{b\sqrt{n} + d}{2(n-1)},$$

where a, b and d are constants.

We used the data of $n = 20, 25, 30$ to compute the linear regression. The result is:

$$y = 0.248067x - \frac{0.26128\sqrt{n} - 0.049115}{2(n-1)} \quad \text{for} \quad (0.3 \leq x \leq 0.7). \tag{1}$$

We then used the data of $n = 35, 40$ to check the result. Surprisingly, they fit very well as shown in Figure 2.

Because $x = cn/t(n) = 2c(n-1)$ and $y = K(n, cn)/t(n)$, the relationship between $K(n, cn)$ and c can be deduced from (1) as

$$K(n, cn) = (0.248067c - 0.26128\sqrt{n} + 0.049115)n \tag{2}$$

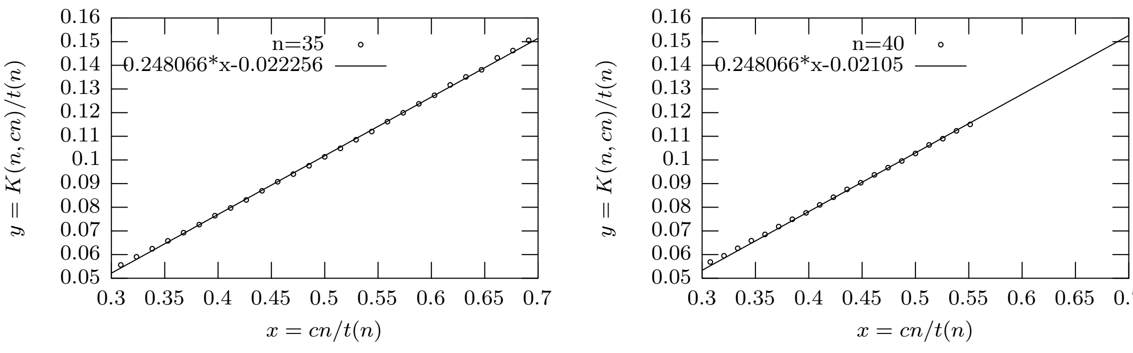
for $0.6(n-1) \leq c \leq 1.4(n-1)$. We still prefer (1) over (2) because x and y in (1) have a fixed range and can be easily plotted in one figure for different n's.

Coppersmith et al. [9] proved by theoretical analysis that, when c large (cf. Theorem 1 (2) in Section 2), the following relation holds.

$$0.25c - 0.343859\sqrt{c} + O(1) \succsim K(n, cn)/n \succsim 0.25c - 0.509833\sqrt{c} \tag{3}$$

From (2), we have $K(n, cn)/n = 0.248067c - 0.26128\sqrt{n} + 0.049115$. This equation looks very much like (3) because $\sqrt{n}$ is close to $\sqrt{c}$ for $0.6(n-1) \leq$

Fig. 3. Relationship between y and x for $n = 35$ $(0.3 \leq x \leq 0.7)$



(a) $n = 35$ **(b)** $n = 40$

$c \leq 1.4(n-1)$. This comparison suggests that the bound of $O(1)$ in (3) can be very small.

Before we attempted to obtain equation (1), we tried another idea for the relation of y and x:

$$y = ax + \frac{bn + d}{n}$$

Using the same data to compute the linear regression, we obtain the following result.

$$y = 0.248067x - \frac{0.012606n + 0.337762}{n} \quad \text{for} \quad (0.3 \leq x \leq 0.7). \tag{4}$$

Then we used the data of $n = 35, 40$ to check (4). It appears that equation (4) fits the data better than equation (1) as shown in Figure 3.

Conjecture 2 *The equation (4) holds for arbitrary large n.*

The relationship between $K(n, cn)$ and c can be deduced from (4) as:

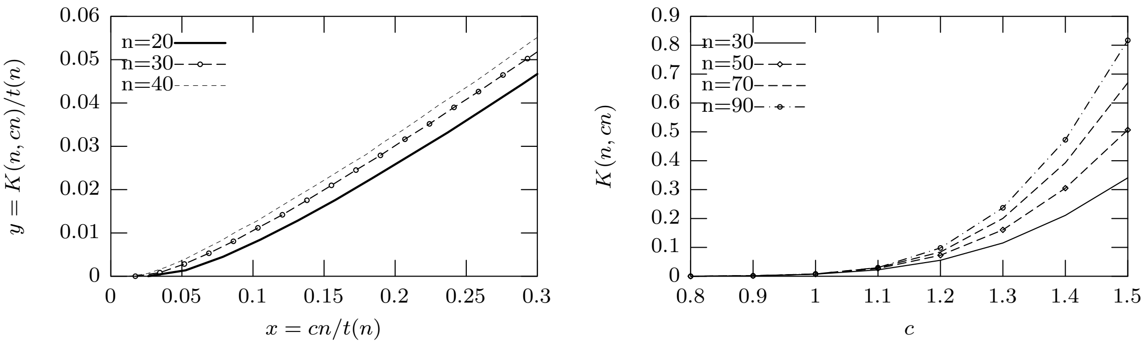
$$K(n, cn) = 0.248067cn - \delta(n) \quad \text{for} \quad 0.6(n-1) \leq c \leq 1.4(n-1). \tag{5}$$

where $\delta(n) = 0.025212n^2 + 0.650312n - 0.675524$.

From (5), we have $K(n, cn)/n = 0.248067c - \delta(n)/n$. This equation also looks very much like (3).

For the region where $0 \leq x < 0.3$, or equivalently, $0 \leq c < 0.6(n-1)$, a close look of the mean value of $K(n, cn)$ is shown in Figure 4(a). We also run some examples for $c = 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5$ to verify the sharp transition in satisfiability for random 2-SAT around $c = m/n = 1$. Figure 4(b) shows the details for $n = 30, 50, 70, 90$ when c near 1. As a future study, we will try

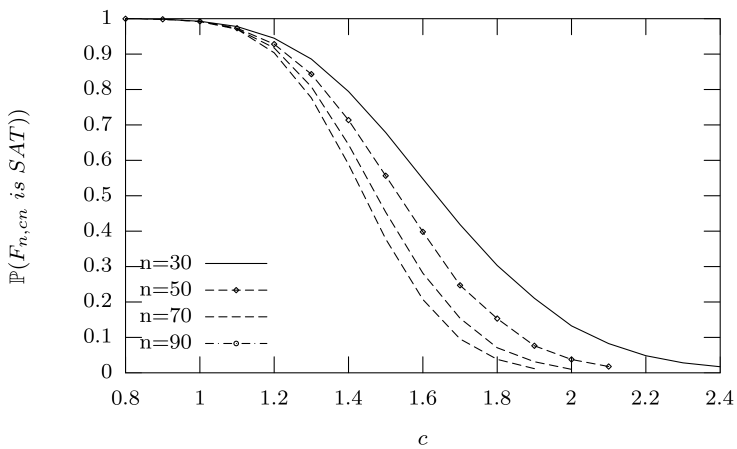
Fig. 4. Detailed view when m is small.



(a) Relation between y and x for $0 \leq x \leq 0.3$

(b) Relation between K and c for $0.8 \leq c \leq 1.5$

Fig. 5. Probability of satisfiable at the phase of transition.



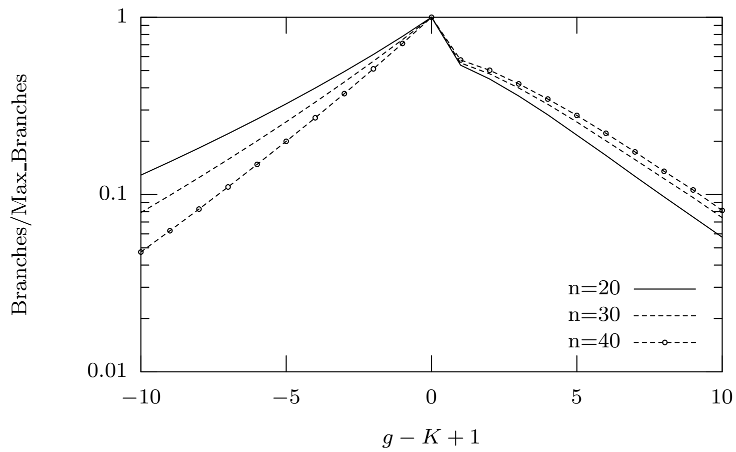
to obtain a more detailed regression function in this region. We tried a second order function and a third order function to fit it, but none of them fit well.

Let $F_{n,cn}$ be a 2-SAT problem with n variables and cn clauses. Chayes, Kim and Borgs [5] gave a theorem about the probability of $F_{n,cn}$ being satisfiable. Let $m = (1+\varepsilon)n$, where $\varepsilon = \lambda_n n^{-1/3}$. It shows that for $\lambda_n < 0$ we have $\mathbb{P}(F_{n,cn}\ is\ SAT) = exp(-\Theta(|\lambda_n|^{-3})$ and for $\lambda_n > 0$ we have $\mathbb{P}(F_{n,cn}\ is\ SAT) = exp(-\Theta(\lambda_n^3)$. From this theorem, they give the result of "scaling window": For all sufficiently small $\delta > 0$, the scaling window is of the form

$$W(n,\delta) = (1 - \Theta(n^{-1/3}), 1 + \Theta(n^{-1/3})).$$

For $n = 30, 50, 70, 90$ and $c = 0.8, 0.9, ..., 2.4$, we have run 10000 instances for each case. Then we computed $\mathbb{P}(F_{n,cn}$ is SAT) for each case. The result is shown in Figure 5. Then we draw a line of $\mathbb{P}(F_{n,cn}) = 0.1$ which cross the probability curves for $n = 90, 70, 50, 30$ at $c \approx 1.7, 1.76, 1.86, 2.06$, respectively, will let $\mathbb{P}(F_{n,cn}) = 0.1$. For $n = 90, 70, 50, 30$, $n^{-1/3} = 0.22314, 0.24264, 0.27144, 0.32183$ and $c - 1 = 0.7, 0.76, 0.86, 1.06$. Does $n^{-1/3}$ and $c - 1$ has a linear relation?

Fig. 6. Computing cost of the decision procedure.



The answer is yes. For $n = 90, 70, 50, 30$,

$$\frac{n^{-1/3}}{c-1} = 0.31877, 0.31926, 0.31563, 0.3065,$$

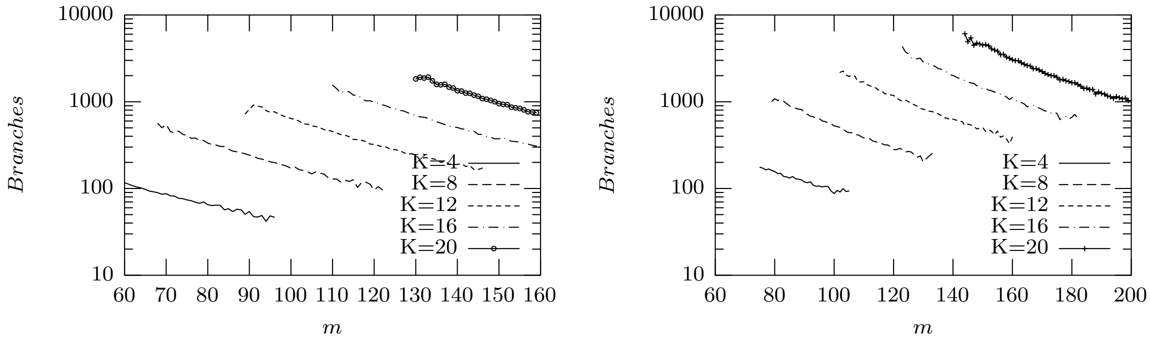
respectively. This result fits the "scaling window" proposed in [5,9] very well.

4 Problem Hardness

Recall that given (F, g) as input, where F is a set of m binary clauses on n variables and g is an integer, the decision procedure for MAX-2-SAT returns true iff $K(n, m) \leq g$. We observed a common easy-hard-easy pattern in the median difficulty of this decision problem, with the hardest problems being associated with the phase transition ($g = K(n, m) - 1$). This observation came as expected because, when $g = K - 1$, the decision procedure presented in [20] has to almost exhaust all the search space before claiming that there exists no assignment under which at most g clauses in F are false. If $g < K - 1$, the procedure will return false early; when $k > K - 1$, the procedure will terminate once a satisfying assignment is found; when $g = K - 1$, the tight bound allows little space to be pruned.

In Figure 6, we plotted the median computing cost of the decision procedure [20] for $n = 20, 30, 40$ and $-10 \leq g - (K(n, m) - 1) \leq 10$. The computing cost is represented by the number of branches of the search tree (the computing time is a linear function of the branches) with respect to the maximum number of branches for each n. It is interesting to notice that as g decreases, the computing cost decreases faster for bigger n on the left side of the threshold point (unsatisfiable cases) but when g increases, the computing cost decreases faster for smaller n on the right side of the threshold point (satisfiable cases).

Fig. 7. Computing cost for $g = K(n, m) - 1$



(a) $n = 20$ **(b)** $n = 25$

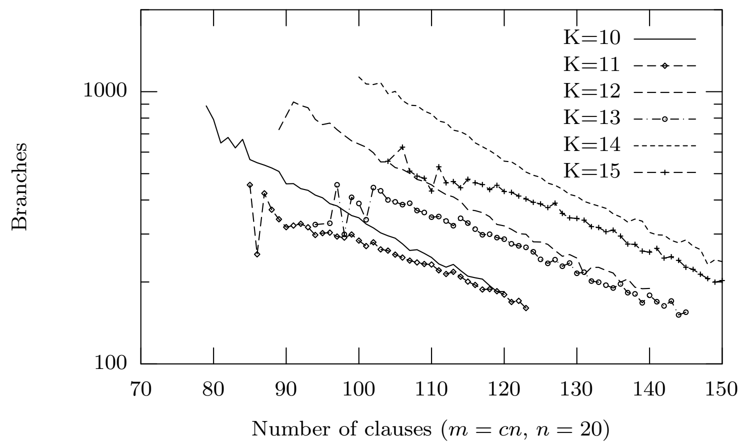
The sharpest drop comes at the threshold point when g goes from $K - 1$ (unsatisfiable) to K (satisfiable).

Another interesting phenomena we observed in the experiment is that, in general, the computing cost increases when K grows. However, the hardest problems are those with higher K values but lower m (or c) values. That is, suppose one instance has n variables and m_1 clauses and another instance has n variables and m_2 clauses. If $K(n, m_1) = K(n, m_2)$ and $m_1 < m_2$, then the first instance intends to be harder than the second instance. To illustrate this observation, we have done some experiments for $n = 20$ and 25. For $n = 20$, $60 \leq m \leq 159$, we generated 4000×100 instances. For $n = 25$, $75 \leq m \leq 199$, we generated 4000×125 instances. Figure 7(a) shows the computing cost of the decision procedure when $g = K - 1$ for $n = 20$ and $K = 4, 8, 12, 16, 20$. Figure 7(b) shows the results when $g = K - 1$ for $n = 25$ and $K = 4, 8, 12, 16, 20$.

One possible explanation for this phenomena is that when more clauses are presented, more information will be obtained from the propagation of an assignment and this information may help to terminate the search on one branch earlier than otherwise.

Finally, we like to point out another interesting phenomena on which we do not have any explanation. for the same n, we said that the computing cost goes up when K goes up. However, it appears tha this observation is true only when comparing even K's with even K's, or when comparing odd K's with odd K's, because it appears that the instances with even K's are harder than those with odd K's. This phenomena is illustrated in Figure 8 for $n = 20$. From the figure, we can see that the instances with $K = 14$ are harder than those with $K = 15$. Similarly, the instances with $K = 12$ are harder than those tieh $K = 13$ (but are easier than those with $K = 15$), and those with $K = 10$ are harder than those with $K = 11$.

Fig. 8. Computing cost for $K = 10, 11, 12, 13, 14, 15,\ n = 20$



5 Conclusion

In this study, we have applied an efficient decision procedure for a great number of random MAX-2-SAT instances. The statistical data of our experiment confirmed not only the "scaling window" of MAX-2-SAT discovered by Chayes, Kim and Borgs [5], but also the recent results of Coppersmith et al. [9]. While there is no easy-hard-easy pattern for the complexity of 2-SAT at the phase transition, we showed that there is such a pattern for the decision problem of MAX-2-SAT associated with the phase transition. We also identified that the hardest problems are among those with high allowance for false clauses but low number of clauses.

Determining bounds for the random k-SAT threshold has been solved for $k = 2$. However, in spite of significant efforts, neither a tight analysis nor the structural properties of this threshold have been determined for $k > 2$. We plan to study the statistical behavior of random MAX-k-SAT as we did for MAX-2-SAT in this paper. We hope our empirical results will provide some insights into the random k-SAT threshold.

References

[1] D. Achlioptas and G.B. Sorkin, Optimal myopic algorithms for 3-SAT, 41st Annual Symposium on Foundations of Computer Science, IEEE Comput. Soc. Press, Los Alamitos, CA 2001, pp. 590-600.

[2] J. Alber, J. Gramm, R. Niedermeier, Faster exact algorithms for hard problems: A parameterized point of view. Preprint, submitted to Elsevier, August, 2000

[3] S. Arora, C. Lund. Hardness of approximation. In D. Hochbaum (ed.): Approximation algorithms for NP-hard problems, Chapter 10, pages 399-446.

PWS Publishing Company, Boston, 1997.

[4] N. Bansal, V. Raman, Upper bounds for MaxSat: Further improved. In Aggarwal and Rangan (eds.): *Proceedings of 10th Annual conference on Algorithms and Computation*, ISSAC'99, volume 1741 of Lecture Notes in Computer Science, pages 247-258, Springer-Verlag, 1999.

[5] B. Bollobas, C. Borgs, J.T. Chayes, J.H. Kim, D.B. Wilson, The scaling window of the 2-SAT transition, Random Structures and Algorithms 18 (2001), no 3, 201-256.

[6] B. Borchers, J. Furman, A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2(4):299-306, 1999.

[7] A.Z. Broder, A.M. Frieze, E. Upfal, On the satisfiability and maximum satisfiability of random 3-CNF formulas, Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX 1993, ACM Press, 1993, pp. 322-330.

[8] V. Chvatal, B. Reed, Mick gets some (the odds are on his side), 33th Annual Symposium on Foundations of Computer Science, Pittsburgh, PA, 1992. IEEE Comput. Soc. Press, 1992, pp. 620-627.

[9] D. Coppersmith, D. Gamarnik, M. Hajiaghay, G.B. Sorkin. Random MAX SAT, random MAX CUT, and their phase transitions. Proceedings of Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), January 12-14, 2003, Baltimore, MD.

[10] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k–SAT based on local search. *Theoretical Computer Science*, 2002.

[11] A. Goerdt, A threshold for unsatisfiability, J. Comput. System Sci. 53 (1996) no. 3, 469–486.

[12] J. Gramm, E.A. Hirsch, R. Niedermeier, P. Rossmanith: New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT. Preprint, submitted to Elsevier, May, 2001

[13] P. Hansen, B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44:279-303, 1990.

[14] E.A. Hirsch. A new algorithm for MAX-2-SAT. In *Proceedings of 17th International Symposium on Theoretical Aspects of Computer Science*, STACS 2000, vol. 1770, Lecture Notes in Computer Science, pages 65-73. Springer-Verlag.

[15] E.A. Hirsch. New worst-case upper bounds for SAT. *Journal of Automated Reasoning*, 24(4):397-420, 2000.

[16] S. Janson, Y.C. Stamatiou, M. Vamvakari, Bounding the unsatisfiability threshold of random 3-SAT, Random Structures and Algorithms **17** (2000), no. 2, 103–116.

[17] M. Mahajan, V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31:335–354, 1999.

[18] R. Niedermeier, Some prospects for efficient fixed parameter algorithms. In *Proc. of the 25th Conference on Current Trends in Theory and Practice of Informatics* (SOFSEM'98), Springer, LNCS 1521, pages 168–185, November, 1998.

[19] R. Niedermeier, P. Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, 36:63-88, 2000.

[20] H. Zhang, H. Shen, New exact algorithms for MAX-2-SAT. Submitted to International Workshop on First-order Theorem Proving (FTP 2003).