Electronic Colloquium on Computational Complexity, Report No. 19 (2000)

Worst-case time bounds for MAX-k-SAT w.r.t. the number of variables using local search

Edward A. Hirsch*

March 2000

Abstract

During the past three years there was an explosion of algorithms solving MAX-SAT and MAX-2-SAT in worst-case time of the order c^K, where $c < 2$ is a constant, and K is the *number of clauses* in the input formula. Such bounds w.r.t. the *number of variables* instead of the number of clauses are not known.

Also, it was proved that approximate solutions for these problems (even beyond inapproximability ratios) can be obtained faster than exact solutions. However, the corresponding exponents still depended on the *number of clauses* in the input formula. In this paper we give a randomized $(1-\epsilon)$-approximation algorithm for MAX-k-SAT. This algorithm runs in time of the order $c_{k,\epsilon}^N$, where N is the *number of variables*, and $c_{k,\epsilon} < 2$ is a constant depending on k and ϵ.

1 Introduction

SAT (the problem of satisfiability of a propositional formula in conjunctive normal form (*CNF*)) can be easily solved in time of the order 2^N, where N is the number of variables in the input formula. In the early 1980s this trivial bound was improved for formulas in 3-CNF (every clause contains at most three literals) to c^N, where $c < 2$ is a constant [4, 17, 18]. After that, many upper bounds for SAT and its NP-complete subproblems were obtained ([6, 13, 20, 21] are the most recent). Most authors consider bounds w.r.t. three main parameters: the *length* L of the input formula (i.e. the number of literal occurrences), the *number* K *of its clauses* and the *number* N *of the variables* occurring in it. The algorithms corresponding to the best known bounds w.r.t. K and w.r.t. L (these bounds are[1] 1.239^K and 1.074^L [13]) are designed for general SAT. However, nothing better than 2^N is known for general SAT w.r.t. the number of variables, such bounds are known only for k-SAT (the best known bounds for 3-SAT are $(4/3)^N$ for randomized algorithms [21], and 1.481^N for deterministic algorithms [6]).

In the past three years there was a significant progress in proving worst-case time bounds for MAX-SAT problem which is an important generalization of SAT. An algorithm for MAX-SAT has to find an assignment satisfying the maximum possible number of clauses even if the input formula is unsatisfiable. The research concentrated on MAX-SAT and MAX-2-SAT (every clause contains at most two literals), both these problems are $\mathcal{NP}$-complete. The best known bounds are:

*Steklov Institute of Mathematics, St.Petersburg, Russia. Supported by INTAS project 96-0760 and RFBR grant 99-01-00113. Email: hirsch@pdmi.ras.ru. Web: http://logic.pdmi.ras.ru/~hirsch/

[1]Here and in what follows, we write the bounds upto a polynomial factor poly(L), for example, 1.239^K is in fact poly(L) 1.239^K etc.

ISSN 1433-8092

- 1.342^K and 1.106^L for MAX-SAT [3],
- $2^{K/4}$ and $2^{L/8}$ for MAX-2-SAT [11].

No non-trivial upper bounds w.r.t. the number of variables are known for MAX-SAT and MAX-2-SAT.

An α-approximation algorithm for MAX-SAT finds an assignment satisfying at least αm clauses of the input formula (m is the maximum possible number of simultaneously satisfiable clauses). There are polynomial-time α-approximation algorithms for MAX-SAT and MAX-k-SAT [2, 7, 15], for example, a 7/8-approximation algorithm for MAX-3-SAT [15]. On the other hand, for each of the MAX-k-SAT/MAX-SAT problems there is an α_0 (*inapproximability ratio*) such that polynomial-time $(\alpha_0 + \delta)$-approximation algorithms ($\delta > 0$) do not exist unless $\mathcal{P} = \mathcal{NP}$ (see, e.g., [1, 10]). In particular, for 3-SAT the inapproximability ratio is $7/8$ [10]. The paper [5] explains how to construct faster $(\alpha_0 + \delta)$-approximation algorithms than the algorithms for the exact solution of MAX-k-SAT/MAX-SAT. However, the exponential-time bounds obtained in this way are w.r.t. the number of clauses (and not w.r.t. the number of variables), for example, for MAX-3-SAT there is a $(7/8 + \delta)$-approximation algorithm running in $2^{8\delta K}$ time.

Experimental study of SAT and MAX-SAT algorithms is also very extensive ([9] is a survey). Both complete and incomplete algorithms are studied in this field, the incomplete algorithms mainly use local search or/and random walk. Theoretical study of worst-case upper bounds for such algorithms was very limited [14, 16, 19]. Recently, Schöning [21] presented a striking randomized local search algorithm for k-SAT resembling Papadimitriou's polynomial-time algorithm for 2-SAT [19]. Schöning's algorithm runs in the time $(2(k-1)/k)^N$. This algorithm picks an initial assignment A at random and then performs a local search: at each step, it (deterministically[2]) chooses a clause unsatisfied by the current assignment A, picks a variable from this clause at random, and changes the value of this variable in A. Clearly, at each step the assignment A is getting closer to some satisfying assignment with probability at least $1/k$ (since at least one variable of the chosen unsatisfied clause has different values in A and in the satisfying assignment).

Unfortunately, this trick does not work for MAX-k-SAT: a MAX-k-SAT instance can contain many clauses that are unsatisfied even by an optimal assignment. Therefore, a deterministic choice of an unsatisfied clause is not satisfactory here. What is done in this paper:

- We allow the algorithm to pick an unsatisfied clause at random.
- We prove that the obtained algorithm is able to find an $(1-\epsilon)$-approximate solution of MAX-k-SAT in time $c_{k,\epsilon}^N$, where $c_{k,\epsilon} < 2$ is a constant depending on k and ϵ.

In fact, to prove this result it suffices to use even a simpler algorithm than Schöning's one (the same situation is in [6] which derandomizes also a simpler algorithm). However, the constant $c_{k,\epsilon}$ is better if we use Schöning's construction.

Curiously, the derandomization of Schöning's algorithm suggested in [6] does not work for our algorithm (at least, literally).

[2]It is not important for Schöning's algorithm *how* to choose this clause, and thus it is not specified in Schöning's paper.

2 Results

2.1 A less-than-2^N bound for $(1-\epsilon)$-approximating MAX-k-SAT.

We consider formulas in k-CNF represented as multisets of clauses. Every clause of a formula in k-CNF is an i-clause for $i \leq k$. An i-clause consists of exactly i literals (a literal is a Boolean variable or the negation of a Boolean variable).

The MAX-k-SAT problem is to find a truth assignment that satisfies the maximum possible number $\mathrm{OptVal}(F)$ of clauses of the input formula F in k-CNF (an *optimal assignment*). An α-approximation algorithm for MAX-k-SAT is an algorithm that for every input formula F finds an assignment satisfying at least $\alpha \cdot \mathrm{OptVal}(F)$ clauses of F. In this section we describe a *randomized* $(1-\epsilon)$-approximation algorithm for MAX-k-SAT (for arbitrary constant $\epsilon > 0$). This algorithm returns an assignment satisfying at least $(1-\epsilon) \cdot \mathrm{OptVal}(F)$ clauses with probability at least $1 - 1/e$ (where $e = 2.71828\ldots$), and otherwise returns an assignment satisfying less clauses.

Our algorithm is very close to Schöning's k-SAT randomized algorithm [21]. Starting from a random initial assignment, we perform a local search. The local search procedure iteratively chooses an unsatisfied clause and changes the value of one of its variables. Schöning's proof uses the fact that for k-SAT this procedure has a constant probability of going in the direction of a satisfying assignment because

> The value of at least one variable from an unsatisfied clause is different in the current (not satisfying) assignment and in an optimal (satisfying) assignment. (1)

For MAX-k-SAT this is not guaranteed because even an optimal (maybe not satisfying!) assignment can make many clauses false. However, if the current assignment satisfies much less clauses than an optimal assignment, then for a significant portion of unsatisfied clauses statement (1) holds. Therefore, for such current assignment a random choice of an unsatisfied clause gives a constant probability of going in the "right" direction.

In this subsection we give the simplest form of our algorithm and prove the worst-case bound on its running time. In the next subsection we describe how to improve the obtained exponent (using Schöning's arguments and other constructions) and how to generalize our result.

Algorithm 1.
Input: A formula F in k-CNF with N variables.
Output: A $(1-\epsilon)$-approximation solution of MAX-SAT problem for F.
Method:

1. Repeat $(2 - \frac{2\epsilon}{k+\epsilon+k\epsilon})^N$ times the following steps:
   (a) Pick an assignment A at random.
   (b) Repeat $N-1$ times the following step:
      (i) If A satisfies every clause of F, then return A. Otherwise pick an unsatisfied clause of F at random, pick a variable from this clause at random, and change its value in A.
2. Among the $N \cdot (2 - \frac{2\epsilon}{k+\epsilon+k\epsilon})^N$ assignments considered by this algorithm, choose an assignment satisfying the greatest number of clauses of F, and return this assignment.

Theorem 1. Algorithm 1 returns a correct answer with probability at least $1 - 1/e$, where $e = 2.171828\ldots$. Its worst-case running time is $\text{poly}(N)c_{k,\epsilon}^N$, where $c_{k,\epsilon} = 2 - \frac{2\epsilon}{k+\epsilon+k\epsilon} < 2$.

Proof. Consider an (optimal) assignment S satisfying $m = \text{OptVal}(F)$ clauses of F. Let K be the total number of clauses in F. If at some moment of time the current assignment A satisfies at least $(1-\epsilon)m$ clauses, then we are done. Otherwise, A does not satisfy $u > K - (1-\epsilon)m$ clauses of F, among them there are at least $u - (K - m)$ clauses satisfied by S. Therefore, the algorithm changes the value of a variable that has different values in A and S with probability at least

$$p_{k,\epsilon} = \frac{u-(K-m)}{ku} = \frac{1}{k} - \frac{K-m}{ku} \geq \frac{1}{k} - \frac{K-m}{k(K-(1-\epsilon)m)} = \frac{\epsilon m}{k(K-(1-\epsilon)m)} \geq \frac{\epsilon m}{k(2m-(1-\epsilon)m)} = \frac{\epsilon}{k(1+\epsilon)}$$

(the second inequality is based on the fact that

$$m \geq \tfrac{1}{2}K, \tag{2}$$

this fact can be shown by a simple probabilistic argument).

Suppose at step (a) the algorithm chooses an assignment that differs from S by the values of exactly n variables (this happens with probability $\frac{\binom{N}{n}}{2^N}$). For such initial assignment, the algorithm finds a required assignment without choosing a different initial assignment at step (a) with probability at least $p_{k,\epsilon,n} = (\frac{\epsilon}{k(1+\epsilon)})^n$.

Summing over all possible choices of n, we have that the probability of success of local search for one initial assignment is at least

$$\frac{1}{2^N}\Sigma_{n=0}^{N}\binom{N}{n}\left(\frac{\epsilon}{k(1+\epsilon)}\right)^n = \left(\frac{1}{2}\left(1 + \frac{\epsilon}{k(1+\epsilon)}\right)\right)^N.$$

Since we choose $(2 - \frac{2\epsilon}{k+\epsilon+k\epsilon})^N = \left(\frac{2}{1+\frac{\epsilon}{k(1+\epsilon)}}\right)^N$ independent initial assignments, the probability of error is at most $1/e$.

The running time bound is straightforward. ☐

2.2 Generalizations and improvements.

Weighted MAX-k-SAT. A simple modification of our algorithm solves weighted MAX-k-SAT problem (with arbitrary reasonable weights). Instead of picking a random unsatisfied clause uniformly, we pick it with probability proportional to its weight. The running time bound remains the same.

Allowing longer local search. Arguments from [21] count not only the probability of obtaining the answer by making n steps in the "right" direction, but also the probability of doing it by making i steps in the "wrong" and $i+n$ steps in the "right" direction. For this, step (i) should be repeated $3N$ (and not $N-1$) times. Although in [21] the probability of going in the "right" direction is $p \geq \frac{1}{t}$ for an integer t, the construction works for arbitrary p and gives the probability $\frac{1}{(t-1)^n}/\text{poly}(N)$ of reaching a required assignment. In this way, the probability $p_{k,\epsilon,n}$ can be improved to $(\frac{\epsilon}{k(1+\epsilon)-\epsilon})^n/\text{poly}(N)$.

Arguments similar to the proof of Theorem 1 then give the probability

$$\left(\frac{1}{2}\left(1 + \frac{\epsilon}{k(1+\epsilon)-\epsilon}\right)\right)^N \Big/ \text{poly}(N)$$

of success before choosing another initial assignment. Therefore, it is enough to pick only $\text{poly}(N) \cdot (2 - \frac{2\epsilon}{k(1+\epsilon)})^N$ initial assignments to get a constant probability of error. Thus $c_{k,\epsilon}$ can be improved to $2 - \frac{2\epsilon}{k(1+\epsilon)}$.

MAX-2-SAT particular case. The MAX-2-SAT part of Yannakakis's MAX-SAT approximation algorithm [22] contains a maximum symmetric flow algorithm which reduces (weighted) MAX-2-SAT to weighted MAX-2E-SAT, i.e. to the instances containing only weighted 2-clauses (and no 1-clauses). More precisely, this algorithm, given a formula F in 2-CNF, outputs a formula F' in 2E-CNF, such that an α-approximation assignment for F can be reconstructed in polynomial time from any α-approximation assignment for F'. Therefore, the inequality (2) can be made tighter:

$$m \geq \frac{3}{4}K.$$

Thus the bound for $p_{k,\epsilon}$ improves to $\frac{\epsilon}{k(1/3+\epsilon)}$ for $k = 2$. Combining with the previous arguments gives an algorithm with $c_{2,\epsilon} = 2 - \frac{2\epsilon}{k(1/3+\epsilon)} = 2 - \frac{6\epsilon}{2+6\epsilon}$.

Remark. The MAX-2-SAT part of Yannakakis' MAX-SAT approximation algorithm has already been used in the context of exponential-time worst-case upper bounds [12]. However, [12] contains an error: Yannakakis' algorithm may introduce clauses with non-integer weights which break the algorithm of [12]. This error is fixed in [11] at the cost of replacing Yannakakis' algorithm by another procedure. Note that for the algorithm of this paper it is not important that Yannakakis' algorithm may introduce non-integer weights and increase the number of clauses.

3 Further research and open questions

The $c^N_{k,\epsilon}$-time $(1-\epsilon)$-approximation algorithm for MAX-k-SAT suggested in this paper may lead to other new exponential-time algorithms for optimization problems. For example, the algorithm generalizes to a MAX-k-CSP approximation algorithm in the same way as Schöning's algorithm. Also, combining with approximation algorithm of [5] or/and parametrized bounds of [8] may give some new bounds.

It is still an open question whether MAX-2-SAT can be solved *exactly* in c^N time, where $c < 2$ is a constant and N is the *number of variables*. The same question is still open for SAT. Also, it would be interesting to derandomize the algorithm suggested in this paper (we have already mentioned that the construction of [6] does not work here).

References

[1] S. Arora and C. Lund. Hardness of approximation. In D. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, chapter 10. PWS Publishing Company, Boston, 1997.

[2] T. Asano and D. P. Williamson. Improved approximation algorithms for MAX SAT. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'00*, 2000.

[3] N. Bansal and V. Raman. Upper bounds for MaxSat: Further improved. In A. Aggarwal and C. Pandu Rangan, editors, *Algorithms and Computation (Proceedings of ISAAC'99)*, volume 1741 of *Lecture Notes in Computer Science*, pages 247–258. Springer-Verlag, December 1999.

[4] E. Dantsin. Two propositional proof systems based on the splitting method (in Russian). *Zapiski Nauchnykh Seminarov LOMI*, 105:24–44, 1981. English translation: *Journal of Soviet Mathematics*, 22(3):1293–1305, 1983.

[5] E. Dantsin, M. Gavrilovich, E. A. Hirsch, and B. Konev. Approximation algorithms for MAX SAT: a better performance ratio at the cost of a longer running time. Technical Report PDMI preprint 14/1998, Steklov Institute of Mathematics at St.Petersburg, 1998. Electronic address: `ftp://ftp.pdmi.ras.ru/pub/publicat/preprint/1998/14-98.ps.gz`.

[6] E. Dantsin, A. Goerdt, E. A. Hirsch, and U. Schöning. Deterministic algorithms for k-SAT based on covering codes and local search. To appear in Proceedings of ICALP'2000, January 2000.

[7] U. Feige and M. X. Goemans. Approximating the value of two proper proof systems, with applications to MAX-2SAT and MAX-DICUT. In *Proceeding of the 3rd Israel Symposium on Theory and Computing Systems*, pages 182–189, 1995.

[8] J. Gramm and R. Niedermeier. Faster exact solutions for Max-2-Sat. In G. Bongiovanni, G. Gambosi, and R. Petreschi, editors, *Algorithms and Complexity (Proceedings of CIAC 2000)*, volume 1767 of *Lecture Notes in Computer Science*, pages 174–186. Springer-Verlag, March 2000.

[9] J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. Algorithms for satisfiability (SAT) problem: A survey. In D. Du, J. Gu, and P. M. Pardalos, editors, *Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11-13, 1996)*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 19–152. AMS, 1997.

[10] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, STOC'97*, pages 1–10, 1997.

[11] E. A. Hirsch. A $2^{K/4}$-time algorithm for MAX-2-SAT: Corrected version. Technical Report 99-036, Revision 02, Electronic Colloquim on Computational Complexity, February 2000. Electronic address: `ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/1999/TR99-036/revisn02.ps`.

[12] E. A. Hirsch. A new algorithm for MAX-2-SAT. In H. Reichel and S. Tison, editors, *Proceedings of STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 65–73. Springer-Verlag, February 2000. Contains an error, fixed in [11].

[13] E. A. Hirsch. New worst-case upper bounds for SAT. *Journal of Automated Reasoning*, 2000. To appear. A preliminary version appeared in *Proceedings of SODA'98*.

[14] E. A. Hirsch. SAT local search algorithms: Worst-case study. *Journal of Automated Reasoning*, 24(1/2):127–143, February 2000.

[15] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, FOCS'97*, pages 406–415, 1997.

[16] E. Koutsoupias and C. H. Papadimitriou. On the greedy algorithm for satisfiability. *Information Processing Letters*, 43(1):53–55, 1992.

[17] B. Monien and E. Speckenmeyer. 3-satisfiability is testable in $O(1.62^r)$ steps. Technical Report Bericht Nr. 3/1979, Reihe Theoretische Informatik, Universität-Gesamthochschule-Paderborn, 1979.

[18] B. Monien and E. Speckenmeyer. Solving satisfiability in less then 2^n steps. *Discrete Applied Mathematics*, 10:287–295, 1985.

[19] C. H. Papadimitriou. On selecting a satisfying truth assignment. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science, FOCS'91*, pages 163–169, 1991.

[20] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, FOCS'98*, pages 628–637, 1998.

[21] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, FOCS'99*, pages 410–414, 1999.

[22] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17(3):457–502, November 1994.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'