Worst-Case Study of Local Search for MAX-k-SAT

Edward A. Hirsch [1]

Steklov Institute of Mathematics at St.Petersburg, 27 Fontanka, 191011 St.Petersburg, Russia

Abstract

During the past three years there was a considerable growth in the number 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* of the input formula. However, similar 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* of the input formula. In this paper, we give a randomized $(1 - \epsilon)$-approximation algorithm for MAX-k-SAT whose worst-case time bound depends on the *number of variables.*

Our algorithm and its analysis are based on Schöning's proof of the best current worst-case time bound for k-SAT [40]. Similarly to Schöning's algorithm (which is also very close to Papadimitriou's algorithm [36] and the experimentally successful WalkSAT family by Selman et al. [41,31]), our algorithm makes random walks of polynomial length. We prove that the probability of error in each walk is at most $1 - c_{k,\epsilon}^{-N}$, where N is the number of variables, and $c_{k,\epsilon} < 2$ is a constant depending on k and ϵ. Therefore, making $\lceil -\ln\rho\rceil \cdot c_{k,\epsilon}^{N}$ such walks gives the probability of error bounded from above by any predefined constant $\rho > 0$.

1 Introduction

The maximum satisfiability problem (*MAX-SAT*) is one of the most important $\mathcal{MAXSNP}$-complete optimization problems since many practical optimiza-

[1] Web: `http://logic.pdmi.ras.ru/~hirsch`. Email: `hirsch@pdmi.ras.ru`. Work is partially supported by grants from INTAS (YSF 99-4044), RFBR, and RAS (6th contest-expertise, grant #1, 1999).

tion problems have natural formulations in terms of MAX-SAT (see references in [4]). During the past decade, this problem has been attacked both by designing heuristic algorithms showing good performance in computational experiments [5,19,29,39] and by proving various theoretical results.

Theoretical results related to MAX-SAT mostly concentrate in two directions: polynomial-time approximation algorithms and exponential-time exact algorithms. We start with basic definitions, brief description of known results both in this field and in the closely related field of SAT algorithms, and then describe our results.

Basic definitions. We consider formulas in conjunctive normal form (*CNF*) represented as multisets of clauses. Every clause of a formula in *k-CNF* is an i-clause for $i \le k$. An i-clause consists of exactly i literals (a literal is a Boolean variable or the negation of a Boolean variable). General CNF is just n-CNF, where n is the number of variables.

The satisfiability (*SAT*) problem is to find a truth assignment that satisfies all clauses of the input formula F in CNF. The k-SAT problem is a subproblem of SAT in which the input is restricted to k-CNF.

The maximum satisfiability (*MAX-SAT*) problem is to find a truth assignment that satisfies the maximum possible number $\mathrm{OptVal}(F)$ of clauses of the (possibly unsatisfiable) input formula F in CNF (an *optimal assignment*). The MAX-k-SAT problem is a subproblem of MAX-SAT in which the input is restricted to k-CNF.

An *α-approximation algorithm* for MAX-SAT (or 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 paper, we suppose $\alpha \le 1$ to be a constant, where $\alpha = 1$ corresponds to an exact algorithm.

Worst-Case Time Bounds for SAT. SAT can be easily solved in time of the order [2] 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 to c^N, where $c < 2$ is a constant [7,32,33]. After that, many upper bounds for SAT and its $\mathcal{NP}$-complete subproblems were obtained ([9,10,23,38,40] 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.

[2] Here and in what follows, we write all bounds up to a polynomial factor $\mathrm{poly}(|F|)$, where $|F|$ denotes the size of the bit representation of the input. For example, 2^N is in fact $\mathrm{poly}(|F|)\, 2^N$, etc.

The algorithms corresponding to the best known bounds w.r.t. K and w.r.t. L (these bounds are 1.239^K and 1.074^L [23]) are designed for SAT (and not only for k-SAT). However, nothing better than 2^N is known for 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 [40] and 1.481^N for deterministic algorithms [9,10]).

Worst-Case Time Bounds for the Exact Solution of MAX-SAT. In the past three years, there was a significant progress in proving worst-case time bounds for the MAX-SAT problem. The research [3,8,16,17,21,22,30,35] concentrated on MAX-SAT and MAX-2-SAT, both of these problems are $\mathcal{NP}$-complete. The best known bounds are:

- 1.342^K and 1.106^L for MAX-SAT [3],
- $2^{K/5}$ and $2^{L/10}$ for MAX-2-SAT [16].

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

Approximation Algorithms for MAX-SAT. There are polynomial-time α-approximation algorithms for MAX-SAT and MAX-k-SAT [2,13,28], for example, a 7/8-approximation algorithm for MAX-3-SAT [28,45]. 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., [1,20]). In particular, for MAX-3-SAT the inapproximability ratio is 7/8, and for MAX-2-SAT the best known inapproximability ratio is 0.955 [20]. (Although, given an almost-satisfiable formula in 2-CNF, it is possible to construct an almost-optimal assignment in polynomial time [44]).

Dantsin et al. [8] explain how to construct $(\alpha_0 + \delta)$-approximation algorithms that are faster than the algorithms for the exact solution of MAX-k-SAT (or MAX-SAT). Their construction uses the Davis-Putnam-Logemann-Loveland procedure [11,12] to obtain a guaranteed number δK of satisfied clauses, and then uses a polynomial-time α_0-approximation algorithm. 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.

Local Search Algorithms and Our Results. SAT and MAX-SAT algorithms have been extensively studied experimentally (see [4,18,27] for surveys on the subject). Both complete and incomplete algorithms are studied in this field. The incomplete algorithms mainly use local search (see, e.g., [14,15,19,25,29,39,41]). Theoretical study of worst-case upper bounds for such algorithms was very limited [24,36]. Recently, Schöning [40] proved the upper bound $(2 - 2/k)^N$ on the worst-case running time of a randomized local search algorithm for k-SAT. The algorithm he used is close to Papadimitriou's 2-SAT algorithm [36]; this random walk procedure is also an important component of some of the empirically best performing SAT algorithms, including the WalkSAT family [41,31,26].

Although Schöning's algorithm performs in practice substantially worse than WalkSAT or even state-of-the-art complete SAT algorithms (e.g., such as rel_sat [6] or zchaff [34,43]), it is the only local search algorithm for which a good worst-case upper bound is proved. In this paper, we suggest a modification of this algorithm that finds a $(1-\epsilon)$-approximate solution for MAX-SAT for arbitrary $\epsilon > 0$. We do not expect that the obtained algorithm would be practical. However, this is the first algorithm for MAX-SAT approximation beyond inapproximability ratio that has a "less-than-2^N" worst-case upper bound, where N is the *number of variables* (note that all previous algorithms had worst-case time bounds with the exponent depending on the number of clauses or the length of the input formula).

The algorithm of Papadimitriou and Schöning picks an initial assignment A at random and then performs a local search: at each step, it chooses (using an arbitrary deterministic heuristic [3]) 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 may contain many clauses that are unsatisfied even by an optimal assignment. Therefore, an (arbitrary) deterministic choice of an unsatisfied clause is not suitable here. Instead, we use an algorithm which, similarly to WalkSAT [41], picks an unsatisfied clause at random and prove that this algorithm is able find an $(1 - \epsilon)$-approximate solution of MAX-k-SAT in the worst-case 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 analysis than Schöning's one (the same applies to the derandomization of Schöning's algorithm [9,10]). However, by using Schöning's construction, we can obtain a better constant $c_{k,\epsilon}$.

[3] It is not important for Schöning's and Papadimitriou's algorithms *how* to choose this clause, thus it is not specified in [36,40]. In contrast, WalkSAT [41] chooses an unsatisfied clause at random.

Organization of the Paper. In Sect. 2 we present our main result. Section 3 contains generalizations and improvements. In Sect. 4 we summarize our results, describe directions for further work, and pose open questions.

2 Main Result

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 $1-\frac{1}{e}$ (where $e = 2.71828\ldots$), and with the remaining probability returns an assignment satisfying less clauses. Clearly, repeating such an algorithm $\lceil -\ln \rho \rceil$ times (and selecting the assignment satisfying the largest number of clauses) gives an assignment satisfying at least $(1-\epsilon)\cdot \mathrm{OptVal}(F)$ clauses with probability at least $1-\rho$ (for any predefined $\rho > 0$).

Our algorithm and its analysis are very close to the ones presented by Schöning [40] for k-SAT. 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 coming closer to 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 substantially less clauses than an optimal assignment, then for a significant portion of unsatisfied clauses statement (1) holds. Therefore, for such a current assignment a random choice of an unsatisfied clause gives a constant probability of going in the "right" direction.

In this section, we give the simplest form of our algorithm and prove an upper bound on its worst-case running time. In the next section, 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 the MAX-SAT problem for F.*

Method:

```
(1) Repeat $\lceil (2 - \frac{2\epsilon}{k+\epsilon+k\epsilon})^N \rceil$ 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 $N \cdot \lceil (2 - \frac{2\epsilon}{k+\epsilon+k\epsilon})^N \rceil$ assignments considered by this algorithm,
    choose an assignment satisfying the greatest number of clauses of $F$, and
    return this assignment.
```

$\square$

Theorem 1 *Algorithm 1 returns an assignment satisfying at least $(1-\epsilon)\cdot$ OptVal(F) clauses of the input formula F with probability at least $1-\frac{1}{e}$, where $e = 2.171828\ldots$. Its worst-case running time is poly($|F|$) $\cdot\, c_{k,\epsilon}^N$, where $c_{k,\epsilon} = 2 - \frac{2\epsilon}{k+\epsilon+k\epsilon} < 2$, and $|F|$ denotes the size of the bit representation of the input formula F.*

PROOF. Consider an (optimal) assignment S satisfying $m = \text{OptVal}(F)$ clauses of F. We call an assignment *admissible* if it satisfies at least $(1-\epsilon)m$ 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

$$m \geq \tfrac{1}{2}K, \qquad (2)$$

which can be shown by the following simple probabilistic argument: take a random assignment; it satisfies each clause with probability at least $\frac{1}{2}$; therefore, the expectation of the number of satisfied clauses is at least $\frac{1}{2}K$, and the number of satisfied clauses is greater than or equal to its expectation with positive probability.

Suppose that at step 1(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 an admissible assignment without choosing a different initial assignment at step 1(a) with probability at least $p_{k,\epsilon,n} = (\frac{\epsilon}{k(1+\epsilon)})^n$.

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

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

Note that for any real $x > 1$, $\ln(1-\frac{1}{x}) = -\frac{1}{x} - \frac{1}{2x^2} - \frac{1}{3x^3} - \ldots$, and thus

$$(1-x)^x \leq \frac{1}{e}. \tag{3}$$

Therefore, by choosing at least $x = (2 - \frac{2\epsilon}{k+\epsilon+k\epsilon})^N = \left(\frac{1}{2}(1+\frac{\epsilon}{k(1+\epsilon)})\right)^{-N}$ initial assignments independently at random, we get the probability of error bounded from above by $\frac{1}{e}$.

The bound on the running time is straightforward. □

Remark 2 *Note that to obtain any predefined error probability* $\rho < 1$ *instead of* $\frac{1}{e}$*, it is sufficient to repeat Algorithm 1* $\lceil -\ln\rho \rceil$ *times. For example, the error probability* $\frac{1}{2^{100}}$ *is achievable by repeating the algorithm just* 70 *times.*

3 Generalizations and improvements.

3.1 Weighted MAX-k-SAT

A simple modification of our algorithm solves the weighted MAX-k-SAT problem: Instead of picking a random unsatisfied clause uniformly, we pick it with probability proportional to its weight. The probability $p_{k,\epsilon} = \frac{u-(K-m)}{ku}$ remains the same (though u, K, and m now denote the total weights and not just the cardinalities of the corresponding sets of clauses). Therefore, the bound on the running time of the algorithm does not change.

Arguments from [40] take into account 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. To use Schöning's arguments, step 1(b)(i) should be repeated $3N$ (and not $N-1$) times similarly to [40]. Schöning [40] proves the following fact about a random walk on the line.

Lemma 3 (Schöning, [40]) *Consider the following random walk on* 0, 1, 2, $\ldots$, N. *A particle starts at state* n. *At each step, the particle goes from the current state* i *to the state* $i-1$ *with probability* p ($0 < p < 1$), *and to the state* $i+1$ *with probability* $1-p$. *If the particle reaches* 0, *it remains there forever. From the state* N, *the particle goes to* $N-1$ *with probability one. Then the probability to reach* 0 *in at most* $3N$ *steps is at least* $(p^{-1}-1)^{-n}/\mathit{poly}(N)$.

This random walk corresponds to the behaviour of our algorithm: the state 0 corresponds to an optimal assignment, and the state i corresponds to the set of assignments that differ from the optimal assignment in the values of exactly i variables.

Using this lemma (where $p = p_{k,\epsilon} = \frac{\epsilon}{k(1+\epsilon)}$), the probability $p_{k,\epsilon,n}$ improves to

$$(p_{k,\epsilon}^{-1}-1)^{-n}/\mathrm{poly}(N) = (\tfrac{k(1+\epsilon)}{\epsilon}-1)^{-n}/\mathrm{poly}(N) = (\tfrac{\epsilon}{k(1+\epsilon)-\epsilon})^{n}/\mathrm{poly}(N).$$

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

$$\tfrac{1}{2^N}\Sigma_{n=0}^{N}\binom{N}{n}(\tfrac{\epsilon}{k(1+\epsilon)-\epsilon})^{n}/\mathrm{poly}(N) = \left(\tfrac{1}{2}(1+\tfrac{\epsilon}{k(1+\epsilon)-\epsilon})\right)^{N}\Big/\mathrm{poly}(N).$$

By (3), it suffices to pick only $\mathrm{poly}(N)\cdot\left(\frac{1}{2}(1+\frac{\epsilon}{k(1+\epsilon)-\epsilon})\right)^{-N} = \mathrm{poly}(N)\cdot(2-\frac{2\epsilon}{k(1+\epsilon)})^{N}$ initial assignments to get the probability of error bounded from above by $\frac{1}{e}$. Hence, $c_{k,\epsilon}$ can be improved to $c'_{k,\epsilon} = 2-\frac{2\epsilon}{k(1+\epsilon)}$ if we modify Algorithm 1 in the following way:

(1) A longer local search is allowed at step 1(b) (step 1(b)(i) is repeated $3N$ times instead of $N-1$ times).

(2) The number of initial assignments considered at step 1 is changed to $\lceil(2-\frac{2\epsilon}{k(1+\epsilon)})^{N}\rceil$.

Overall, we obtain the following result.

Theorem 4 *The modified Algorithm 1 returns an assignment satisfying at least* $(1-\epsilon)\mathit{OptVal}(F)$ *clauses of the input formula* F *with probability at least*

$1-\frac{1}{e}$, where $e = 2.171828\ldots$. Its worst-case running time is poly$(|F|) \cdot (c'_{k,\epsilon})^N$, where $c'_{k,\epsilon} = 2 - \frac{2\epsilon}{k(1+\epsilon)} < 2$, and $|F|$ denotes the size of the bit representation of the input formula F.

3.3 Better construction for MAX-2-SAT

The MAX-2-SAT part of Yannakakis's MAX-SAT approximation algorithm [42] 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, the following fact is proved.

Lemma 5 (Yannakakis, [42]) *There is a polynomial-time algorithm that, given a formula G in 2-CNF, outputs a formula F in 2E-CNF such that an α-approximation assignment for G can be reconstructed in polynomial time from any α-approximation assignment for F.*

This allows us to prove the following theorem.

Theorem 6 *For any $\epsilon > 0$, one can construct an algorithm that given a formula F in 2-CNF returns an assignment satisfying at least $(1-\epsilon)\,OptVal(F)$ clauses of F with probability at least $1-\frac{1}{e}$, where $e = 2.171828\ldots$, and runs in time poly$(|G|) \cdot (c''_{2,\epsilon})^N$, where $c''_{2,\epsilon} = 2 - \frac{3\epsilon}{1+3\epsilon} < 2$, and $|G|$ denotes the size of the bit representation of the input formula G.*

PROOF. Transform the input formula G into a formula F in $2E$-CNF using Yannakakis's algorithm. Then apply Algorithm 1 modified as described in Subsection 3.2 to F (but set the number of initial assignments considered at step 1 to $\lceil (2-\frac{3\epsilon}{1+3\epsilon})^N \rceil$). An admissible assignment for G can be reconstructed from an admissible assignment for F using Lemma 5.

A random assignment satisfies every 2-clause with probability $\frac{3}{4}$. Therefore, the expected number of clauses of F satisfied by a random assignment is $\frac{3}{4}K$. Thus there is an assignment satisfying at least $\frac{3}{4}K$ clauses, i.e., the inequality (2) can be made tighter:

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

Thus the bound for $p_{k,\epsilon}$ in Theorem 1 improves to $\frac{\epsilon}{2(1/3+\epsilon)}$. By Lemma 3, the probability $p_{k,\epsilon,n}$ improves to

$$(p_{k,\epsilon}^{-1} - 1)^{-n}/\text{poly}(N) = (\tfrac{2(1/3+\epsilon)}{\epsilon} - 1)^{-n}/\text{poly}(N) = (\tfrac{\epsilon}{2/3+\epsilon})^n/\text{poly}(N).$$

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

$$\frac{1}{2^N}\Sigma_{n=0}^{N}\binom{N}{n}(\tfrac{\epsilon}{2/3+\epsilon})^n/\text{poly}(N) = \left(\tfrac{1}{2}(1+\tfrac{\epsilon}{2/3+\epsilon})\right)^N \Big/ \text{poly}(N).$$

By (3), it suffices to pick only

$$\text{poly}(N)\cdot\left(\tfrac{1}{2}(1+\tfrac{\epsilon}{2/3+\epsilon})\right)^{-N} = \text{poly}(N)\cdot(\tfrac{1/3+\epsilon}{2/3+\epsilon})^N = \text{poly}(N)\cdot(2-\tfrac{3\epsilon}{1+3\epsilon})^N$$

initial assignments to get the probability of error bounded from above by $\frac{1}{e}$.

The bound on the running time is straightforward. □

Remark 7 *The MAX-2-SAT part of Yannakakis's MAX-SAT approximation algorithm has already been used in the context of exponential-time worst-case upper bounds [22]. However, [22] contains an error: Yannakakis' algorithm may introduce clauses with non-integer weights which break the algorithm of [22]. This error is fixed in [21] 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.*

4 Conclusion

In this paper, we presented the first algorithm that is able to find a $(1-\varepsilon)$-approximate solution for the MAX-k-SAT problem in the worst-case time $p_\rho(|F|)\cdot c_{k,\epsilon}^N$ for any predefined probability of error $\rho < 1$ and any $\epsilon > 0$, where N is the *number of variables*, $c_{k,\epsilon} < 2$ is a constant depending only on k and ϵ, $|F|$ denotes the size of the bit representation of the input, and p_ρ is a polynomial whose coefficients depend on ρ. Before, only such bounds w.r.t. the *number of clauses* were known.

We also proved that for MAX-2-SAT, the constant $c_{2,\epsilon}$ can be improved to $c''_{2,\epsilon} = 2 - \frac{3\epsilon}{1+3\epsilon}$.

Our algorithm can be generalized straightforwardly to the weighted MAX-k-SAT problem. Less obvious applications of our results and open questions related to the subject are listed in the two next subsections.

4.1 Further work

The $c_{k,\epsilon}^N$-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 and its analysis generalize to MAX-k-CSP [4] similarly to Schöning's algorithm [40]. Also, combining with the approximation algorithm of [8], the polynomial-time approximation algorithms of [13,28,44], the parametrized bounds of [3], or/and the exact bounds of [3,16] may give some new algorithms and bounds. It would be interesting to compare both theoretically and empirically the behaviour of all these algorithms on various classes of formulas ("long" vs "short", satisfiable vs almost satisfiable vs having many unsatisfied clauses for every assignment, obtained by reductions of other problems, etc.).

4.2 Open Questions

(1) Derandomize the MAX-k-SAT algorithm suggested in this paper. Curiously, the derandomization of Schöning's k-SAT algorithm suggested in [9,10] does (at least directly) not work for our algorithm.
(2) Design an algorithm solving MAX-2-SAT *exactly* in c^N time, where $c < 2$ is a constant and N is the number of variables. Note that the same question is still open for SAT but not for k-SAT.
(3) Design a polynomial-time local search α-approximation algorithm for MAX-k-SAT with a good approximation ratio α. Note that there has been a significant experimental study of local search MAX-SAT algorithms (see, e.g., [5,19,29,39]); moreover, every SAT local search algorithm can be viewed as a MAX-SAT approximation algorithm. However, currently no proof of a good approximation ratio for such an algorithm is known.
(4) Experimental study [27,37] suggests that in many cases, removing the repeated choice of random initial assignments does not affect the performance of a local search algorithm. Even if the repeated choice is essential, it can be replaced by the random walk extension [27]. However, the proofs of Schöning's bounds and of the bound shown in this paper heavily rely on (exponentially many of) such restarts. It would be interesting to show that polynomially many (or even a constant number of) restarts are sufficient for obtaining the same worst-case time bound.

[4] MAX-k-CSP is the problem of finding a solution for a system of k-constraints. A k-constraint is a predicate on at most k variables, where each variable takes values in some finite domain.

Acknowledgement

The useful comments of three anonymous referees of *Discrete Applied Mathematics* greatly improved the presentation of this paper and added the last open question.

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*, pages 96–105, 2000.

[3] N. Bansal and V. Raman. Upper bounds for MaxSat: Further improved. In A. Aggarwal and C. Pandu Rangan, editors, *Proceedings of the 10th International Symposium on Algorithms and Computation, ISAAC'99*, volume 1741 of *Lecture Notes in Computer Science*, pages 247–258. Springer-Verlag, December 1999.

[4] R. Battiti and M. Protasi. Approximate algorithms and heuristics for MAX-SAT. In *Handbook of Combinatorial Optimization*, volume 1, pages 77–148. Kluwer Academic Publishers, 1998.

[5] R. Battiti and M. Protassi. Reactive search, a history-sensitive heuristic for MAX-SAT. *ACM Journal of Experimental Algorithmics*, 2, 1997. Paper 2.

[6] R. Bayardo and R. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'97*, pages 203–208, 1997.

[7] 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.

[8] 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. *Annals of Pure and Applied Logic*, 2001. To appear. Preliminary version: Approximation algorithms for MAX SAT: a better performance ratio at the cost of a longer running time, 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`.

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

[10] E. Dantsin, A. Goerdt, E. A. Hirsch, and U. Schöning. Deterministic algorithms for k-SAT based on covering codes and local search. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proceedings of the 27th International Colloquium on Automata, Languages and Programming, ICALP'2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 236–243. Springer-Verlag, 2000.

[11] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[12] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, July 1960.

[13] 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.

[14] I. P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proceedings of the 11th National Conference on Artificial Intelligence, AAAI'93*, pages 28–33. AAAI Press, 1993.

[15] I. P. Gent and T. Walsh. Unsatisfied variables in local search. In J. Hallam, editor, *Hybrid Problems, Hybrid Solutions*, pages 73–85. IOS Press, 1995.

[16] J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith. New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT. Technical Report 00-037, Electronic Colloquium on Computational Complexity, June 2000. Electronic address: `ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/2000/TR00-037/index.html`. To appear in *Discrete Applied Mathematics.*

[17] J. Gramm and R. Niedermeier. Faster exact solutions for Max-2-Sat. In G. Bongiovanni, G. Gambosi, and R. Petreschi, editors, *Proceedings of the 4th Italian Conference on Algorithms and Complexity, CIAC 2000*, volume 1767 of *Lecture Notes in Computer Science*, pages 174–186. Springer-Verlag, March 2000.

[18] 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, *The 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.

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

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

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

[22] E. A. Hirsch. A new algorithm for MAX-2-SAT. In H. Reichel and S. Tison, editors, *17th International Symposium on Theoretical Aspects of Computer Science, STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 65–73. Springer-Verlag, February 2000. Contains an error, fixed in [21].

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

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

[25] E. A. Hirsch and A. Kojevnikov. Solving Boolean satisfiability using local search guided by unit clause elimination. In T. Walsh, editor, *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, CP'01*, volume 2239 of *Lecture Notes in Computer Science*, pages 605–609. Springer-Verlag, 2001.

[26] H. H. Hoos. On the run-time behaviour of stochastic local search algorithms for SAT. In *Proceedings of the 16th National Conference on Artificial Intelligence, AAAI'99*, pages 661–666, 1999.

[27] H. H. Hoos and T. Stützle. Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning*, 24(4):421–481, 2000.

[28] 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.

[29] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In D. Du, J. Gu, and P. M. Pardalos, editors, *The Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11-13, 1996)*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 573–586. AMS, 1997.

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

[31] D. McAllester, B. Selman, and H. Kautz. Evidence in invariants for local search. In *Proc. AAAI'97*, pages 321–326, 1997.

[32] 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.

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

[34] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 39th Design Automation Conference*, 2001. To appear.

[35] R. Niedermeier and P. Rossmanith. New upper bounds for Maximum Satisfiability. *Journal of Algorithms*, 36:63–88, 2000.

[36] 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.

[37] A. J. Parkes and J. Walser. Tuning local search for satisability testing. In *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI'96*, pages 356–362, 1996.

[38] 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.

[39] M. G. C. Resende, L. S. Pitsoulis, and P. M. Pardalos. Approximate solution of weighted MAX-SAT problems using GRASP. In D. Du, J. Gu, and P. M. Pardalos, editors, *The Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11-13, 1996)*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 393–405. AMS, 1997.

[40] 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.

[41] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the 12th National Conference on Artificial Intelligence, AAAI'94*, pages 337–343, 1994.

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

[43] L. Zhang. zChaff. An implementation of the Chaff solver. Available from `http://www.ee.princeton.edu/~chaff/zchaff.html`.

[44] U. Zwick. Finding almost satisfying assignments. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing, STOC'98*, pages 551–560, 1998.

[45] U. Zwick. Computer assisted proof of spherical volume inequalities. Manuscript, 21 pages, 2001.