Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Algorithms 54 (2005) 40–44
Journal of Algorithms
www.elsevier.com/locate/jalgor

An algorithm for the satisfiability problem of formulas in conjunctive normal form

Rainer Schuler

Abt. Theoretische Informatik, Universität Ulm, D-89069 Ulm, Germany

Received 26 February 2003
Available online 9 June 2004

Abstract

We consider the satisfiability problem on Boolean formulas in conjunctive normal form. We show that a satisfying assignment of a formula can be found in polynomial time with a success probability of $2^{-n(1-1/(1+\log m))}$, where n and m are the number of variables and the number of clauses of the formula, respectively. If the number of clauses of the formulas is bounded by n^c for some constant c, this gives an expected run time of $O(p(n) \cdot 2^{n(1-1/(1+c\log n))})$ for a polynomial p.

Keywords: Complexity theory; NP-completeness; CNF-SAT; Probabilistic algorithms

1. Introduction

The satisfiability problem of Boolean formulas in conjunctive normal form (CNF-SAT) is one of the best known NP-complete problems. The problem remains NP-complete even if the formulas are restricted to a constant number $k > 2$ of literals in each clause (k-SAT). In recent years several algorithms have been proposed to solve the problem exponentially faster than the 2^n time bound, given by an exhaustive search of all possible assignments to the n variables. So far research has focused in particular on k-SAT, whereas improvements for SAT in general have been derived with respect to the number m of clauses in a formula [2,7]. The best known time bound for unbounded clause size with respect to the number of variables is $2^{n-\varepsilon\sqrt{n}}$ for a constant $\varepsilon > 0$ [6]. The result is derived from the much stronger time bound $2^{n(1-1/k)}$ known for k-SAT [5].

E-mail address: schuler@informatik.uni-ulm.de.

0196-6774/$ – see front matter
doi:10.1016/j.jalgor.2004.04.012

Theorem 1 [5]. *There exists a randomised algorithm such that for every formula $F \in SAT$*

(1) *The algorithm on input F outputs a satisfying assignment with probability at least $2^{-n(1-1/k)}$, where k is a bound on the number of literals in each clause.*
(2) *The running time of the algorithm is polynomially bounded in the size* (*number of literals*) *of the formula.*

The question arises whether time-bounds of the form $O(2^{n(1-\alpha)})$, for some $\alpha > 0$, are possible for formulas with unbounded clause size. A partial answer in this direction was given in [1]. If the number of clauses is linearly bounded, i.e., less than cn for some constant $c > 1$, then satisfying assignments can be found in time $2^{n(1-\alpha)}$ for some constant $\alpha > 0$ depending on c. (It suffices to choose α such that

$$H(\alpha) \leqslant 1/6c,$$

where $H(\alpha)$ is the entropy function.)

2. A probabilistic algorithm for CNF-SAT

As usual, we use n to denote the number of variables of a formula F and m to denote the number of clauses. The size of a clause is the number of literals it contains. Let $c > 1$ be some constant. A satisfiable formula is in $\mathrm{SAT}(n^c)$ if the number of clauses is bounded by n^c. We use $\log n$ to denote the smallest integer larger than the logarithm with base 2 of n.

```
Input CNF-Formula F on variables x_1, x_2, ..., x_n

choose uniformly at random
    a permutation π of 1, 2, ..., n
    a string y ∈ {0, 1}^n

let F_0 = F
for i = 1 to n do
    let x denote the π(i)th variable x_π(i)
    if F_{i-1} contains x in a unit clause
        set the truth value of x to 1
    else if F_{i-1} contains the negation of x in a unit clause
        set the truth value of x to 0
    else set the truth value of x to the ith bit of y
    let F_i denote the formula which is derived from F_{i-1} by
        deleting all literals set to 0
        deleting all clauses which contain a literal set to 1
if F_n contains the empty clause output fail
else (F_n is empty) output truth assignment of x_1, x_2, ..., x_n
```

Fig. 1. Algorithm of Paturi, Pudlák and Zane (PPZ-Algorithm).

Theorem 2. *There exists a randomised algorithm A such that for every formula $F \in$ CNF-SAT*

(1) *The algorithm A on input F outputs a satisfying assignment with probability at least $2^{-n(1-1/(1+\log m))}$.*
(2) *The running time of A is polynomially bounded in the size* (*number of literals*) *of the formula.*

Proof. Using a probabilistic polynomial-time algorithm of Paturi, Pudlák, and Zane [5] a satisfying assignment of a formula can be found with success probability of at least $2^{-n(1-1/k)}$, if every clause of the formula contains at most k literals. In the following we use PPZ-algorithm to refer to this algorithm (see Fig. 1).

Let F denote some satisfiable formula in CNF, n denote the number of variables and m_0 denote the number of clauses of F. The algorithm to find a satisfying assignment is defined as follows.

Input CNF-Formula F

Let $C_1, \ldots, C_m$ denote the clauses of F
for $i = 1$ *to* m *do*
    if the size of C_i is larger than $1 + \log m_0$
        let clause D_i contain the first $1 + \log m_0$ literals of C_i
    else let clause D_i be equal to C_i
Run the PPZ-algorithm on input $G = \bigwedge_{i=1}^m D_i$
if the PPZ-algorithm yields a (satisfying) assignment a *output* a
else if all clauses are of size at most $\log m_0$ *output* fail
else
    uniformly at random choose a clause D_i of size $1 + \log m_0$
    set the truth value of all literals in D_i to 0
    let F' denote the formula which is derived from F by
        deleting all literals set to 0
        deleting all clauses which contain a literal set to 1
    recursively call the algorithm with formula F'

First we note that any assignment satisfying $G = \bigwedge_{i=1}^m D_i$ also satisfies F. The converse, however, is not true.

Let a^* denote some (unknown) satisfying assignment of F. We distinguish two cases. Either G is satisfiable or not. If G is satisfiable, then the PPZ-algorithm will find a satisfying assignment with the required probability.

Now assume G is not satisfiable, i.e., at least one clause of G of size $1+\log m_0$ evaluates to false under a^*. Let i be the smallest index such that D_i evaluates to false. The probability that D_i is selected by the algorithm is at least $1/m_0$. In this case the truth assignments given to the $1 + \log m_0$ variables of D_i is identical to the truth assignments under a^*. Hence, the derived formula F' is satisfiable and a^* restricted to the remaining variables is a satisfying assignment of F'.

Let j denote the number of recursive calls, until all clauses in G are true under a^*. If we require that in each iteration the first clause of size $1 + \log m_0$ which is false under a^* is selected, then j is a fixed number for every formula F and assignment a^*. This follows, since in this case, the literals which are set to 0 are determined deterministically. Note that the number of clauses which evaluate to false under a^* in, e.g., the initial call of the algorithm might be larger or smaller than j.

Let p denote the probability that after j recursive calls the formula G is satisfiable. Then p can be estimated by

$$p \geqslant \left(\frac{1}{m_0}\right)^j \geqslant 2^{-j \log m_0}.$$

Since in each recursive call, the PPZ-algorithm is used to find a satisfying assignment of G, the probability that a satisfiable assignment will be found in the jth recursive call is at least

$$p \cdot 2^{-(n-j(1+\log m_0))(1-1/(1+\log m_0))}.$$

The success probability of the algorithm is therefore at least

$$\begin{aligned} &2^{-j \log m_0} \cdot 2^{-(n-j(1+\log m_0))(1-1/(1+\log m_0))} \\ &\quad = 2^{-j(1+\log m_0)(1-1/(1+\log m_0))} \cdot 2^{-n(1-1/(1+\log m_0))+j(1+\log m_0)(1-1/(1+\log m_0))} \\ &\quad = 2^{-n(1-1/(1+\log m_0))} \qquad \square \end{aligned}$$

The probability to find a satisfying assignment can be amplified by running the algorithm several times. The expected number of iterations until a satisfying assignment is found (with some constant probability) is the inverse of the success probability.

Corollary 3. *Satisfying assignments of formulas in CNF-SAT can be found in expected time*

$$O\bigl(p(n+m) \cdot 2^{n(1-1/(1+\log m))}\bigr),$$

where p is some polynomial and n an m are the number of variables and clauses of the formula, respectively.

A special case occurs when the number of clauses is polynomially bounded in the number of variables.

Corollary 4. *Let $c > 1$ be some constant. Satisfying assignments of formulas in SAT(n^c) can be found in expected time*

$$O\bigl(p(n) \cdot 2^{n(1-1/(1+c \log n))}\bigr),$$

where p is some polynomial.

It remains an open question, whether a similar time bound can be achieved by a deterministic algorithm. Further, it would be interesting to know whether the time bound could be improved to $2^{n(1-\alpha)}$ for some constant $\alpha > 0$ [3,4].

Acknowledgment

The author thanks the anonymous referee for helpful comments.

References

[1] V. Arvind, R. Schuler, The quantum query complexity of 0-1 knapsack and associated claw problems, in: Proceedings of the 14th Annual International Symposium on Algorithms and Computation, in: Lecture Notes in Comput. Sci., Springer-Verlag, 2003, in press.
[2] E.A. Hirsch, New worst-case upper bounds for sat, 2nd Special Issue on SAT, J. Automat. Reason. 24 (4) (2000) 397–420.
[3] R. Impagliazzo, R. Paturi, Complexity of k-SAT, in: Proceedings of the 14th Annual IEEE Conference on Computational Complexity, IEEE Computer Society Press, 1990, pp. 237–240.
[4] P. Pudlák, R. Impagliazzo, A lower bound for dll algorithms for k-SAT (preliminary version), in: Proceedings of the Eleventh Annual ACM–SIAM Symposium on Discrete Algorithms, ACM Press, 2000, pp. 128–136.
[5] R. Paturi, P. Pudlák, F. Zane, Satisfiability coding lemma, in: 38th Ann. IEEE Sympos. on Foundations of Comp. Sci., FOCS'97, 1997, pp. 566–574.
[6] P. Pudlák, Satisfiability—algorithms and logic, in: Proceedings of the 23rd Mathematical Foundations of Computer Science, vol. 1450, Springer-Verlag, 1998, pp. 129–141.
[7] J.M. Robson, Algorithms for maximum independent sets, J. Algorithms 7 (1986) 425–440.