Title	On the Complexity of Subproblems of SAT (New Developments of Theory of Computation and Algorithms)
Author(s)	Matsuura, Akihiro; Iwama, Kazuo
Citation	数理解析研究所講究録 (2001), 1205: 113-118
Issue Date	2001-05
URL	http://hdl.handle.net/2433/41010
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録 1205 巻 2001 年 113-118

SAT のいくつかの部分問題の複雑さについて
On the Complexity of Subproblems of SAT

松浦 昭洋　　岩間 一雄
京都大学大学院 情報学研究科 通信情報システム専攻
Akihiro Matsuura　　Kazuo Iwama
Department of Communications and Computer Engineering,
Graduate School of Informatics, Kyoto University
{matsu, iwama}@kuis.kyoto-u.ac.jp

Abstract

In this paper, we show some complexity results of subproblems of Satisfiability Problem (SAT). We introduce a decision problem that asks if, given a k-CNF formula with n variables, there is a satisfying assingment with at most pn variables set to 1. For $k \geq 2$, we show that (1) when $p = n^c$ $(-1 < c \leq 0)$, the problem is NP-complete, and (2) when $p = \log n/n$, it is solvable in $O(n^{\log k}|\phi|)$ time for a formula ϕ. We also show some results on complexity of a class of formulas for which we know some information on the number of satisfying assignments.

1 Introduction

The satisfiability problem of CNF formulas (SAT) is one of the well-known NP-complete problems [1]. Due to its direct connection to many combinatorial problems, deep understanding of the complexity of SAT is considered to be one of the important problems in computer science. Extensive research has been done to clarify the complexity and to develop efficient algorithms. The complexity is investigated not only for the whole problem but also for the subproblems. Some of them are shown to be NP-complete and some are solved in polynomial time. For example, k-SAT for $k \geq 3$, which consists of CNF formulas with k literals per clause, is proved to be NP-complete. On the other hand, 2-SAT and Horn SAT are shown to be in P. (Refer to a survey [2].) Along this line, it should be a very important and intriguing problem to clarify which part of the problem is hard and which is not.

Papadimitriou et al. [3] studied one of such problems called LOG ADJUSTMENT: Given a CNF formula and a truth assignment a, is there is a satisfying truth assignment whose Hamming distance from a is at most $\log n$? This problem originally comes from artificial intelligence [4], and was proved to be *LOGSNP-complete* [3]. LOGSNP is an intermediate complexity class between P and NP, and LOGSNP-complete problems are likely to require quasipolynomial time to solve exactly. Then, the following natural question arises: *What if a formula is restricted to be k-CNF and if the condition on Hamming distance is more general?* More formerly, we consider a decision problem called k-SAT$_p$ that, given a k-CNF formula ϕ with n variables, asks if there is a satisfying truth assignment that has only pn variables set to 1. Namely, p is the ratio of the number of variables set to 1. We note that LOG ADJUSTMENT is equivalent to k-SAT$_p$ with $k = n$ and $p = \log n/n$. In this paper, we clarify the complexity of k-SAT$_p$ for some typical p's. Namely, for any $k \geq 2$, we show that (1) when $p = n^c$, where c is a constant such that $-1 < c \leq 0$, k-SAT$_p$ is NP-complete, and (2) when $p = \log n/n$, k-SAT$_p$ is in P and solvable in $O(k^{\log n} \cdot |\phi|)$ time, where $|\phi|$ is the size of the formula ϕ. The result for $p = \log n/n$ contrasts with the LOGSNP-completeness of LOG ADJUSTMENT.

Next, we consider formulas that have *at least* $q2^n$ satisfying assignments. When $q = 1/2$, for example, this implies that 50 % of the 2^n truth assignments are satisfying ones. Such a formula was

first considered in [5]. They showed that for any polynomial $p(n)$ and for $q = 1/p(n)$, there is a polynomial time algorithm that finds one satisfying assingment with $O(\log n)$ variables set to 1. In this case, the algorithm for k-SAT$_{\log n/n}$ can be also applied although the previous algorithm is slightly faster. Furthermore, we consider a formula that has less number of satisfying assingments; namely, we consider the case the formula has at least $q2^n$ satisfying assignments where $q = O(1/2^{cn})$ and c is a constant such that $0 < c \leq 1$. Then, the corresponding decision problem that asks the satisfiability of such a formula turns out to be NP-complete for $k \geq 3$.

This paper is organized as follows: In Section 2, we give the definitions concerning SAT. In Section 3, we consider the complexity of k-SAT$_p$ for two typical p's. Section 4 is devoted to another type of formulas that have a certain number of satisfying assingments. Finally, we make concluding remarks in Section 5.

2 Preliminaries

In this section, we define some concepts on CNF formulas and some complexity class.

Definition 1. **k-SAT** is the problem of satisfiability of a formula in k-CNF. **SAT** is the problem of satisfiability of a general formula in CNF.

Given a (k-)CNF formula ϕ of n variables, when ϕ is satisfiable and has a satisfying truth assignment a^*, we count the number of variables set to 1. The ratio of the number of variables set to 1 over n is written as p. We explore how difficult it is to find a satisfying assignment with a specified number of variables set to 1.

Definition 2. **(k-)SAT$_p$**: Given a (k-)CNF formula with n variables, is there a satisfying truth assignment that has at most pn variables set to 1?

We note that for $p = O(1/n)$, since $pn = O(1)$ means that there are only constant number of variables set to 1, so it can be trivially solved in polynomial time by exhaustive search. Also fo $p = O(1)$, (k-)SAT$_p$ coincides with (k-)SAT.

There is an intermediate class called LOGSNP that is defined using logical expression.

Definition 3. **LOGSNP** is a subclass of NP whose language can be polynomially reduced to a problem in the following class of problems:

$$\{I : \exists S \in [n]^{\log n}\, \forall x \in [n]^r\, \exists j \in [\log n]^s\, \phi(I, s_j, x, j)\},$$

where $[n] = \{1, 2, \cdots, n\}$, I is the input relation, r and s are some constant integers, x and j are are tuples of first-order variables, and ϕ is a quantifier-free first-order expression.

(k-)SAT$_{\log n/n}$ is easily shown to be in LOGSNP. Especially, SAT$_{\log n/n}$ that is called LOG ADJUSTMENT in [3] is proved to be LOGSNP-complete. LOGSNP-complete problems are supposed to require quasipolynomial time to be solved exactly.

In the next section, we study the complexity of (k-)SAT$_p$ for general cases.

3 Complexity of k-SAT$_p$

3.1 Complexity of k-SAT$_{n^c}$

In this section, we show the following:

Theorem 1. For $k \geq 2$ and for $p = n^c$ $(-1 < c \leq 0)$, k-SAT$_p$ is NP-complete.

Proof. First, we prove for the case 2-SAT$_{n^c}$. It is reducible from VERTEX COVER (VC). VC is first reduced to its variant called VC$_{n^c}$. VC$_{n^c}$ is the following decision problem:
VC$_{n^c}$: Given a graph G with n vertices, is there a vertex cover of G that has at most size n^{1+c}?

Given an instance of VC such that (G, K) with a connected graph $G = (V, E)$ such that $|V| = n$ and integer K, we transform it to an instance of VC$_{n^c}$ as follows. The idea is to add a reasonable number of vertices to one (arbitrarily) chosen vertex v_0 of V. Namely, let W be a set of vertices such that $|W| = (K+1)^{\frac{1}{1+c}} - n$ and suppose that each vertex in W is connected with only v_0. If $|W|$ is less than 0 with above definition, we set $W = \phi$ and add no vertex. Now, we define the instance of VC$_{n^c}$ so that $G' = (V', E')$, $V' = V \cup W$, and $E' = E$. If V_0 is a vertex cover of G of size $\leq K+1$, $V_0 \cup \{v_0\}$ is a vertex cover of G'. The ratio of the size of the vertex cover is

$$\frac{|V_0 \cup \{v_0\}|}{|V| + |W|} \leq \frac{K+1}{n + |W|} \leq (n + |W|)^c,$$

since $K + 1 \leq (n + |W|)^{1+c}$. Therefore, G' is a yes instance of VC$_{n^c}$. Conversely, if V_0 is a vertex cover of G' whose ratio is at most $(n + |W|)^c$, then $|V_0| \leq (n + |W|)^{1+c} \leq K$. Therefore, VC$_{n^c}$ is shown to be NP-complete. The next step is to reduce VC$_{n^c}$ to 2-SAT$_{n^c}$. For a edge $\{v_i, v_j\}$, we correspond a disjunct $x_i + x_j$. Then, we see that a graph in VC$_{n^c}$ has a vertex cover of size at most n^{1+c} if and only if the corresponding 2-CNF formula has a satisfying assignment with at most n^{1+c} variables set to 1. Therefore, 2-SAT$_{n^c}$ is shown to be NP-complete.

For 3-SAT$_{n^c}$, the reduction is done from 2-SAT$_{n^c}$ by transforming a 2-CNF clause C to $(C + y) \cdot (C + \overline{y})$, where y is a new variable. It is clear that C is satisfiable if and only if $(C + y) \cdot (C + \overline{y})$ is satisfiable. Furthermore, the new 3-CNF formula has a satisfying assignment whose ratio of variables set to 1 is less than n^c. □

Remark. For $p = \log^i n/n$ $(i \geq 1)$, the above transformation makes the number of clauses exponential on n, so such reduction does not work. This case will be treated in the next section.

3.2 Algorithm for k-SAT$_{\log n/n}$

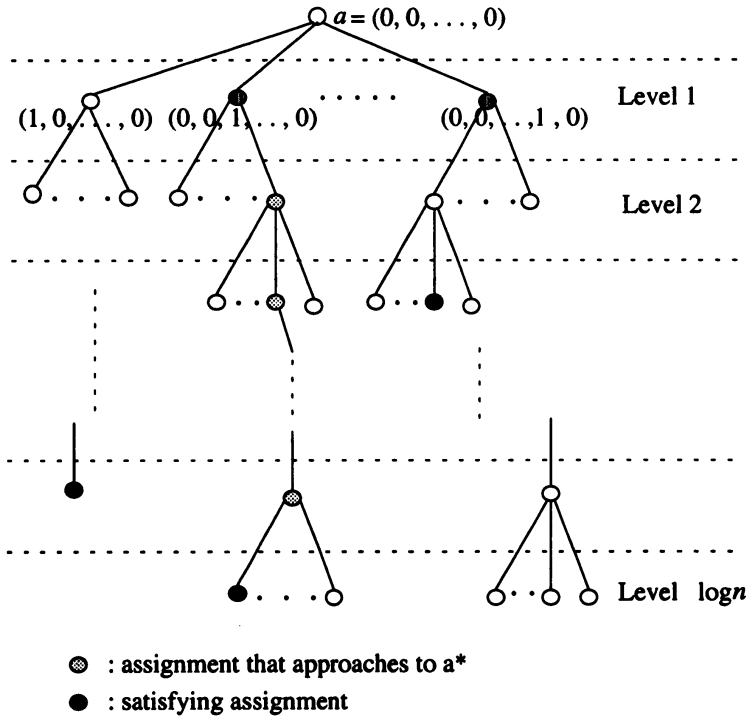
In this section, we show that k-SAT$_p$ for $p = \log n/n$ can be solved deterministically in polynomial time. The result is the following.

Theorem 2. *For $p = \log n/n$, k-SAT$_p$ is in P. For a k-CNF ϕ with n variables for which the answer is yes, it is solvable in $O(n^{\log k} \cdot |\phi|)$ time.*

Proof. The algorithm is based on the derandomized version of Schöning's probabilistic algorithm [6], [7]. We first summarize the probabilistic algorithm by Shöning. Given a k-CNF formula with n variables, the algorithm proceeds as follows:

1. guess an initial assignment $a_0 \in \{0, 1\}^n$, uniformly at random;
2. repeat $3n$ times: If the formula is satisfied by the actual assignment, stop and accept. Let C be some clause not being satisfied by the actual assignment, pick one of the $\leq k$ literals in the clause at random and flip its value in the current assignment.

This can be derandomized by setting specific initial assignments and by making computation for all the choices of literals to be flipped in the above $3n$ iterations. There is a derandomized version of this algorithm [7] and it takes exponential time to obtain the satisfying assignment for a general formula.



However, since we want to know if there is a satisfying assignment with at most $\log n$ variables set to 1, this procedure is applied quite well.

Suppose that there is a satisfying assingment with at most $\log n$ 1's and denote it by a^*. We first set the initial assignment to be $a = (0,\ 0,\ \cdots, 0)$. Then, the algorithm is the following:

Algorithm for k-SAT$_{\log n/n}$

1. evaluate the value of the formula by the present assignment;
2. pick up (any) unsatisfied clause and apply all the k kinds of flips of the k variables it has;
3. for all the k flips in 2, apply the procedures 1 and 2 recursively;
4. at each branch of the recursion, if the depth of the recursion is $\log n$ or satisfying assingment is found, stop.

This procedure is expressed using a tree T as shown in Fig. 1 with the following properties: (i) Let $a_0 = (0,\ 0,\ \cdots,\ 0)$ be a root node; (ii) each node has a label of a truth assignment; (iii) nodes whose label has a Hamming distance of i from the root is said to be at Level i; and (iv) node u is a leaf if either of the two conditions hold: (a) u satisfies the formula, or (b) u is at Level $\log n$.

We can now show the following key claim.

Claim. T has at least one leaf that has a label of a satisfying assignment.

Proof of Claim. Recall that there is a satisfying assingment with at most $\log n$ 1's. First, if initial assignment a_0 satisfies ϕ, then the algorithm stops and accepts. If a_0 does not, there must be at least one assignment at Level 1 which approaches to a^* for one Hamming distance. Since this property holds at every level, for the whole tree of T up to Level $\log n$, there must be at least one leaf in T

that approaches to a^* for one Hamming distance successfully at every Level, or has a label of different satisfying assignment from a^*. □

If a formula is a "no" instance of k-SAT$_{\log n/n}$, suppose that there is no satisfying assignment with at most $\log n$ 1's. Then, the tree T with at most $\log n$ levels can not find a satisfying assingment since otherwise, it can not be a no instance. Therefore, the algorithm outputs no.

As for the computing time, Since the size of the tree is $O(k^{\log n}) = O(n^{\log k})$ and it takes $O(|\phi|)$ time at each node for evaluating the formula, the total running time is $O(n^{\log k} \cdot |\phi|)$. □

Finally, we note that for the case $p = \log^2 /n$, k-SAT$_p$ is only known to be in LOG^{2}SNP, which is defined similarly to LOGSNP.

3.3 On formulas that have a certain number of satisfying assingments

There is a strong relation between the number of variables set to be 1 in the satisfying assignment and the number of satisfying assignments. Hirsch showed a polynomial time algorithm for k-CNF formula that has *many* satisfying assignments [5].

Theorem 3. ([5]) *For $0 < \delta < 1$, suppose that k-CNF formula ϕ with n variables has at least $\delta 2^n$ satisfying assignments, there is a deterministic algorithm to find one satisfying assignment in $O(n^{B(k)\log(2/\lambda(k))} \cdot |\phi|)$ time, where ϕ is the size of a formula ϕ. Furthermore, the number of variables set to be 1 in the assignment can be at most $\log_{2/\lambda(k)}(\frac{2}{\delta}) + k - 1$.*

Here, $\lambda(k)$ is the only positive root of $f(x) = 1 - x^{-1} - x^{-2} - \cdots - x^{-k}$ and $B(k) = (\log_{\lambda(k)} 2 - 1)^{-1}$. Note that $1 < \lambda(k) < 2$ and $B(3) = 7.27$, $B(4) = 17.79$, and etc. If we consider the case that it finds one satisfying assignment with $O(\log n)$ variables set to 1, the algorithm for k-SAT$_{\log n/n}$ can also find a satisfying assignment. Namely, the following corollary of Theorems 2 and 3 hold:

Corollary 1. *For any polynomial $p(n)$ and for any k-CNF formula with n variables that has at least $2^n/p(n)$ satisfying assignments, the algorithm for k-SAT$_{\log n/n}$ finds one satisfying assignment in polynomial time.*

As for the running time, however, in the case both algorithms output a satisfying assingment with $\log n$ variables set to 1, and for $k = 3$, the previous algorithm takes time $O(n^{0.88} \cdot |\phi|)$ and ours takes time $O(n^{1.58} \cdot |\phi|)$, where $|\phi|$ is the size of the formula, and for all k's (≥ 3), the previous algorithm is faster.

On the number of satisfying assignments, the following natural question arises: *What if $p(n)$ is larger?* Namely, we are further interested in a case the number of satisfying assignments is at least $2^n/p(n)$ for $p(n)$ larger than any polynomial. Typical cases are when the numbers of satisfying assignments are at least $2^n/n^{\log n}$, and 2^{cn} for $0 \leq c < 1$. In each case, we know by Theorem 3 that the corresponding formula has a satisfying assignment with $O(\log^2 n)$ or $O(n)$ 1's, respectively.

In the former case, we only note that the formula is a yes instance of k-SAT$_{O(\log^2 n/n)}$. In the latter case, let us consider a CNF formula ϕ with n variables which is either unsatisfiable or satisfiable with at least 2^{cn} satisfying assignments. Then, we show that the satisfiability of such a formula is shown to be NP-complete for any c such that $0 \leq c < 1$. The idea is to simply increase the number of satisfying assignments. More precisely, we transform a k-CNF formula ϕ to a CNF formula $\widetilde{\phi} = \phi \cdot \psi$, where

$$\psi = \prod_{i=3}^{m} (y_1 + y_2 + \cdots + y_{i-1} + y_i),$$

where $y_1, y_2, \cdots, y_m$ are newly introduced variables. It is easy to check that the number of unsatisfying assignments of ψ is less than $2^{m-3} + 2^{m-4} + \cdots 2 + 1 = 2^{m-2} - 1$. Therefore, by taking sufficiently

large m (still polynomial on n), the number of satisfying assignments for $\tilde{\phi}$ can be more than $2^{c(m+n)}$ for any c such that $0 \leq c < 1$. To summarize, the following theorem holds:

Theorem 4. *For $k \geq 3$, $0 < \delta \leq 1$, and $0 \leq c < 1$, let ϕ be a k-CNF formula with n variables which is either unsatisfiable or has at least $\delta 2^{cn}$ satisfying assignments. Then, the decision problem to ask if ϕ is satisfiable is NP-complete.*

It might be interesting to note that with respect to parameter c, $c = 1$ is the tight threshold; namely, for $c = 1$, a satisfying assignment of the corresponding formula is computed in polynomial time and for $c < 1$, the decision problem is NP-complete.

4 Concluding remarks

In this paper, we presented some results on the complexity of some problems related to SAT. There are some problems left open, e.g., the complexity of k-SAT$_p$ for $p = \log^i n/n$ ($i \geq 2$), the complexity of general CNF formulas with at least $\delta 2^n$ satisfying assignments for $0 < \delta < 1$, and so forth. The related maximization and minimization problems should be also studied.

Reference

[1] S. A. Cook, "The Complexity of Theorem-Proving Procedures," *Proc. of 3rd STOC*, pp. 151-158, ACM, New York, 1971.

[2] E. Dantsin, "The Algorithmics of the Propositional Satisfiability Problem," in Problems of Reducing the Exhaustive Search (V. Kreinovich and G. Mints, editors), AMS Translation-Series 2, Vol. 178, 1997.

[3] C. H. Papadimitriou and M. Yannakakis, "On Limited Nondeterminism and the Complexity of the V-C Dimension," *JCSS* 53, pp. 161-170, 1996.

[4] B. Selman, "Tractable Default Reasoning," PhD. thesis, Chap. 6, University of Tronto; Also, Collection open problems. Also presented at the Workshop on Coping with NP-completeness, UCSD, 1990.

[5] E. A. Hirsch, "A Fast Deterministic Algorithm for Formulas That Have Many Satisfying Assignments," *L. J. of the IGPL*, Vol. 6, No. 1, pp. 59-71, Oxford University Press, 1998.

[6] U. Schöning, "A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems," *Proc. of 40th FOCS*, pp. 410-414, 1999.

[7] E. Dantsin, A. Goerdt, E. A. Hirsch, and U. Schöning, "Deterministic Algorithms for k-SAT Based on Covering Codes and Local Search," *Proc. of ICALP2000*, LNCS 1853, pp. 236-243, Springer-Verlag, 2000.