A Simplified NP-complete MAXSAT Problem

Venkatesh Raman[1*], *B. Ravikumar*[2] and *S. Srinivasa Rao*[1]

[1] The Institute of Mathematical Sciences, C. I. T. Campus, Chennai 600 113. India

[2] Department of Computer Science, University of Rhode Island, Kingston RI 02881. USA

Abstract

It is shown that the MAX2SAT problem is NP-complete even if every variable appears in at most three clauses. However, if every variable appears in at most two clauses, it is shown that it (and even the general MAXSAT problem) can be solved in linear time. When every variable appears in at most three clauses, we give an exact algorithm for MAXSAT that takes at most $O(3^{n/2}n)$ steps where n is the number of variables.

1 Introduction

A Boolean formula in conjunctive normal form with n variables and m clauses is called a (k,s)-formula[5] if every clause contains exactly k variables, and every variable occurs in at most s clauses. If every clause contains at most k variables, then we call such a formula $(\leq k, s)$ formula. If there is no bound on the number of variables per clause, we will denote the corresponding formula $(\leq n, s)$ formula if every variable appears in at most s clauses.

We denote by (k,s)-SAT (respectively, $(\leq k,s)$-SAT) the SATISFIABILITY problem restricted to (k,s) (respectively, $(\leq k,s)$) formulas. That is, the (k,s)-SAT problem is: given a (k,s)-formula, is it satisfiable? It is known that $(\leq 3,3)$-SAT is NP-complete[1]. Tovey[10] has shown that $(\leq n,2)$-SAT can be solved in polynomial time.

When all clauses have the same number of variables, Papadimitriou[8] has shown that $(3,5)$-SAT is NP-complete. Recently Tovey[10] has improved this to show that $(3,4)$-SAT is NP-complete and that every $(3,3)$-formula is satisfiable. It follows easily that these NP-complete results carry over to the MAXSAT problem (where we want to find the maximum number of clauses satisfiable) when each clause contains either exactly three or at most three variables.

Here we address the analogous question for the MAXSAT problem restricted to instances where each clause contains at most two variables. Such instances are also called MAX2SAT instances. We denote by (k,s)-MAXSAT (respectively, $(\leq k,s)$-MAXSAT) the MAXSAT problem restricted to (k,s) (respectively, $(\leq k,s)$) formulas. The (decision version of the) (k,s)-MAXSAT problem is: given a (k,s) formula, and an integer l, is there an assignment to the variables that satisfies at least l clauses of the formula? We are interested in the

[*]contact author: vraman@imsc.ernet.in

following question: what is the minimum s for which $(2, s)$-MAXSAT is NP-complete? The original reduction from 3SAT to MAX2SAT had some variable appearing 12 times([1], [2]) if we start from a 3SAT instance in which each variable appears at most 3 times. Jaumard and Simeone[3], while discussing the complexity of MAX2SAT for Horn Formulas give a reduction from Vertex Cover to MAXSAT. If we start from a cubic graph in that reduction, then every variable in the resulting MAXSAT formula appears at most four times. This proves that $(\leq 2, 4)$-MAXSAT problem is NP-complete. We reproduce this reduction in the next section for completeness. Then, by a reduction from $(\leq 2, 4)$-MAXSAT, we show that $(\leq 2, 3)$-MAXSAT as well as $(2, 3)$-MAXSAT are NP-complete.

In Section 3, we show that the $(\leq n, 2)$-MAXSAT problem can be solved in $O(n)$ time where n is the number of variables. This also gives an $O(n)$ algorithm which is simpler than Tovey's for the $(\leq n, 2)$-SAT problem. In Section 4, we deal with exact algorithms for the MAXSAT problem along the lines of exact algorithms for the 3-SAT problem([6], [7], [9]). We show that the $(\leq n, 3)$-MAXSAT can be solved in $O(3^{n/2}n)$ time. Section 5 concludes with open problems.

2 $(2, 3)$-MAXSAT

First we observe the following from the reduction of Jaumard and Simeone[3].

Lemma 2.1: *$(\leq 2, 4)$-MAXSAT is NP-complete.*

Proof: It is obvious to see that the problem is in NP. The following is the reduction of Jaumard and Simeone[3] from Vertex Cover to the MAXSAT problem. We will assume that the graph in the instance of the Vertex Cover problem is cubic, i.e. every vertex has degree 3. Vertex Cover problem remains NP-complete for cubic graphs[2].

Given a cubic graph $G(V, E)$ with n vertices and m edges, introduce for each vertex i, $1 \leq i \leq n$, a variable x_i and define the formula

$$\Phi = (\wedge_{i=1}^{n} x_i) \wedge (\wedge_{\{i,j\} \in E} (\bar{x_i} \vee \bar{x_j})).$$

Φ has $n + m$ clauses with each clause having at most two literals and every variable appearing in at most four clauses.

Now, if G has a vertex cover of size at most k, then setting those variables corresponding to the vertices in the vertex cover to 0, all but at most k clauses of Φ can be satisfied. Conversely if there is an assignment that satisfies at least $n + m - k$ clauses of Φ, we can assume without loss of generality that the false clauses are the clauses with single literals. For, if there is an edge $\{i, j\} \in E$ in which both x_i and x_j are 1, arbitrarily setting one of x_i and x_j to 0 satisfies that clause (and perhaps more clauses) and falsifies only one clause, thus keeping the number of clauses satisfied at least $n + m - k$. Now it is easy to see that the set of vertices corresponding to the variables assigned 0, form a vertex cover of size at most k in G. ∎

Now we prove the main result of the paper.

Theorem 1: *$(\leq 2, 3)$-MAXSAT is NP-complete.*

Proof: The reduction is from $(\leq 2, 4)$-MAXSAT. Let F be a $(\leq 2, 4)$ formula. Let U

be the set of variables in F and let $d(x)$, for $x \in U$ be the number of clauses in which the variable x appears in F. Let b be the number of variables x for which $d(x) = 4$.

For every variable x for which $d(x) = 4$, do the following. First, create a separate variable for each of its occurrences: x_1, x_2, x_3, x_4 and replace the i-th occurrence of the variable with x_i. We further introduce the variables $y_1, y_2, ...y_8$ and the following clauses:

$$(x_1 \vee y_1)(x_2 \vee y_2)(x_3 \vee y_3)(x_4 \vee y_4)$$
$$(\bar{x_1} \vee \bar{y_7})(\bar{x_2} \vee \bar{y_8})(\bar{x_3} \vee \bar{y_5})(\bar{x_4} \vee \bar{y_6})$$
$$(\bar{y_1} \vee y_5)(\bar{y_2} \vee y_6)(\bar{y_3} \vee y_7)(\bar{y_4} \vee y_8)$$
$$(\bar{y_1} \vee y_2)(\bar{y_3} \vee y_4)(y_5 \vee \bar{y_6})(y_7 \vee \bar{y_8})$$

Let X be the above set of 16 clauses and let G be the resulting MAXSAT formula (after performing this construction for every variable x with $d(x) = 4$). Now clearly every variable in the formula G appears in at most three clauses (the x_i variables appear once in the original formula, and twice in the new set of clauses, and the y_i variables appear thrice in the new set of clauses). We claim that F has an assignment that satisfies at least k of the clauses if and only there is an assignment to the variables of G that satisfies at least $k + 16b$ clauses of G.

If F has an assignment that satisfies at least k clauses, then by assigning the truth value of x to x_1, x_2, x_3 and x_4 for every x appearing 4 times, we can satisfy the k original clauses and the new $16b$ clauses as follows. If x is 0, then set all the y_i variables to 1, and if x is 1 then set all the y_i variables to 0.

Conversely if there is an assignment to G that satisfies at least $k + 16b$ clauses, then we first show that there is an equivalent assignment that satisfies at least $k + 16b$ clauses of G that also satisfies *all* the new $16b$ clauses. We then show that the only assignments that satisfy all the $16b$ new clauses are the ones in which the truth value of all the x_i values corresponding to the variable x in F are the same (either all 0's or all 1s) and from that we can recover a truth assignment for F that satisfies at least k of the original clauses. The proof will be complete from the following observations.

1. All the 16 clauses of X are satisfiable if $x_1 = x_2 = x_3 = x_4 = 1$ or $x_1 = x_2 = x_3 = x_4 = 0$.

2. If exactly one of the x_is is 1 or if exactly one of the x_is is 0, then at most 15 clauses of X are satisfiable.

3. If exactly two of the x_is are 1, then at most 14 clauses of X are satisfiable.

Claim 1 can be easily verified. To prove Claim 2, let without loss of generality, $x_1 = 1$ and $x_2 = x_3 = x_4 = 0$. Then all the three clauses $(\bar{x_1} \vee \bar{y_7}), (y_7 \vee \bar{y_3})$ and $(y_3 \vee x_3)$ cannot be satisfied. Furthermore, setting $y_i = 1$ for $1 \leq i \leq 8$ satisfies all clauses of X except the clause $(\bar{x_1} \vee \bar{y_7})$. Similarly if $x_1 = 0$ and $x_2 = x_3 = x_4 = 1$, then all the three clauses $(x_1 \vee y_1), (\bar{y_1} \vee y_5)$ and $(\bar{x_3} \vee \bar{y_5})$ cannot be satisfied. Furthermore, setting $y_i = 0$ for $1 \leq i \leq 8$, satisfies all clauses of X except the clause $(x_1 \vee y_1)$.

To prove claim 3, first let $x_1 = x_2 = 1$ and $x_3 = x_4 = 0$. Then not all three clauses $(x_3 \vee y_3), (\bar{y_3} \vee y_7), (\bar{y_7} \vee \bar{x_1})$ can be satisfied. Independently, not all three clauses $(x_4 \vee$

$y_4), (\bar{y}_4 \vee y_8), (\bar{y}_8 \vee \bar{x}_2)$ can be satisfied. Furthermore, setting $y_i = 0$ for all i, satisfies all the clauses of X except the clauses $(x_3 \vee y_3)$ and $(x_4 \vee y_4)$. Exactly the same argument works if the two x_is that are 1 are x_r and x_s where $s = r \ mod \ 4 + 1$. Now consider the case when $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$ (The complementary case is symmetric). Not all the clauses $(x_2 \vee y_2), (\bar{y}_2 \vee y_6), (\bar{y}_6 \vee y_5)$ and $(\bar{y}_5 \vee \bar{x}_3)$ are satisfiable. Independently not all the clauses $(x_4 \vee y_4), (\bar{y}_4 \vee y_8), (\bar{y}_8 \vee y_7)$ and $(\bar{y}_7 \vee \bar{x}_1)$ are satisfiable. Furthermore, setting $y_i = 1$ for all i, satisfies all clauses of X except $(\bar{x}_1 \vee y_7)$ and $(\bar{x}_3 \vee y_5)$.

Now, given an assignment for G that satisfies at least $k + 16b$ clauses, to obtain an equivalent assignment that satisfies all the $16b$ new clauses and still $k + 16b$ clauses in total, we do the following. For each set of the 16 new clauses, in case 1 above do nothing; in case 2 above, we flip the only variable which is 1 or 0 by which we will satisfy one extra (new) clause and we may falsify at most one original clause (the clause containing that variable). In case 3 above, by flipping the two x_i's from 1 to 0, we will satisfy two extra (new) clauses and falsify at most two original clauses (the two original clauses containing these two variables). ∎

Observe that the new clauses introduced in the above reduction are only two literal clauses. Hence the following corollary follows.

Corollary 1: $(2,3)$-*MAXSAT is NP-complete.*

Proof: Start with a $(\leq 2, 3)$-MAXSAT formula. For every variable x appearing in a unit clause, introduce a new variable y and replace the unit clause (x) by the two clauses $(x \vee y) \wedge (x \vee \bar{y})$ to obtain a formula G in which every clause has exactly two literals. If u out of the total m clauses of F are unit clauses, then G has $m + u$ clauses. Furthermore there is an assignment that satisfies at least k clauses of F if and only if there is an assignment that satisfies at least $k + u$ clauses of G.

Now the variables (x) that appeared in unit clauses in F may be appearing in four clauses. Thus the resulting formula becomes a $(2,4)$-formula. Now use the reduction used in the proof of Theorem 1 to obtain a $(2,3)$ formula. ∎

The MINSAT problem[4] asks for the minimum number of clauses of a given CNF formula that must be satisfied by any assignment to the variables. Kohli, Krishnamurti and Mirchandani[4] proved that the decision version of the problem is NP-complete (even for 2-SAT formulas) by a reduction from MAX2SAT. In fact, their reduction preserves the number of appearances of each variable in the original MAX2SAT formula. So, from corollary 1, it follows that

Corollary 2: $(2,3)$-*MINSAT is NP-complete. That is, given a* $(2,3)$ *formula, and an integer* l*, it is NP-complete to decide whether there is an assignment to the variables that satisfies at most* l *clauses of the formula.*

3 $(\leq n, 2)$-MAXSAT

Tovey[10] has shown that when every variable appears in at most two clauses, SAT can be solved in polynomial time. His observation (using Hall's theorem for perfect matching) is that if every variable appears at most twice and if every clause contains at least two variables, then the formula is always satisfiable. So then it suffices to eliminate the unit clauses, and Tovey deals with the unit clauses in the obvious way.

Observe that if every variable appears at most twice, then without loss of generality, a variable appearing in a unit clause can be set so as to make the clause true even for MAXSAT. Thus Tovey's algorithm (in fact, a simpler version of it) can be applied to solve the MAXSAT problem as well. We describe the simpler $O(n)$ algorithm for the problem below.

```
MAXSAT ($F$)

input  A CNF formula $f$ with $m$ clauses on $n$ variables, with each variable appearing
       in at most two clauses. Let $U$ be the set of unit clauses, and $N$ be the set of
       non-unit clauses.

output  An assignment that satisfies the maximum number of clauses.

begin
       In one pass over $F$, set those variables that appear either only positively or
       only negatively appropriately and remove those clauses (from $N$ or $U$). (Now
       if a variable appears twice, then one of its occurrence is pure and the other is
       complemented.)

       while $U \neq \emptyset$ do
           pick the literal $x$ that appears in a unit clause $UC$ and set it to true.
           Remove $UC$ from $U$.
           If $x$ appears elsewhere, then let $C$ be the clause in which its (other) comple-
           mentary occurrence is present. If $C$ is in $U$, remove it from $U$. Otherwise,
           remove the variable $x$ from $C$; if $C$ becomes a unit clause now, then add
           it to $U$ after removing it from $N$.

       endwhile

       If $N \neq \emptyset$, then pick a clause in $N$, and set any literal $y$ appearing in it to true.
       Remove this clause from $N$. If $y$ appears elsewhere, remove $y$ from that clause.
       If that clause now becomes a unit clause, then add it to $U$ after removing it
       from $N$. Recurse on the reduced formula $F'$. I.e. call MAXSAT ($F'$).
       If any variable is still unset, set it arbitrarily.
end
```

We show that the above algorithm correctly computes the answer to the MAXSAT problem and that it can be implemented in $O(n)$ time.

The correctness of the algorithm follows from the following simple observations.

- If there is a unit clause in the formula, then the variable in the clause can be set to make the clause true.

  Suppose in the optimal assignment, the variable is set otherwise. By flipping it we gain this unit clause and may loose at most one clause (the clause in which the variable appears in its complementary form); thus setting the variable to make the clause true gives as good an assignment as the optimal one.

- If there are no unit clauses in the formula, any variable in one of the clauses can be set to make the clause true.

  It is known[10] that if every variable appears at most twice and every clause has at least two variables, then the formula is satisfiable. We show that in such circumstances, setting an arbitrary literal of an arbitrary clause true and reducing the formula by setting the variables of the unit clauses appropriately along the way, does not falsify any clause. Once we set a literal x true, we remove the clause to which x belongs; if x appears elsewhere, then we pick an arbitrary unset literal of the clause in which $\bar{x}$ appears (such a literal should exist in that clause since we have eliminated unit clauses) and set it true, and continue the process. This process stops when there is no complementary occurrence of the recently set literal. Thus, we have reduced the formula to a formula of the kind we started out with, without falsifying any clause.

If we simply want the maximum number of clauses that can be satisfied, then the algorithm can keep appropriate counters and stop after the while loop.

Implementation

We first create a ternary array A of size n where $A[i]$ gets the value 0, 1 or 2 to indicate whether the i-th variable is set to "false", "true" or "unset" respectively. Initially all the array locations are initialized to 2. In one pass over the formula, by keeping appropriate counters, we can identify unit clauses and those variables that appear either only positively or only negatively. These variables are set appropriately by setting their $A[i]$ values and by removing the clauses in which they appear.

The sets U, N and the clauses themselves are represented by linked lists. Furthermore, with each variable, there is a pointer to and from the positions of each of its occurrences in the clauses to which it belongs.

It can be seen that every time a step of the 'while loop' or the (first) 'if statement' is executed, at least one clause is removed from the formula. Using the above data structure, it can also be easily seen that each step in the 'while loop' takes constant time whereas each execution of the (first) 'if statement' takes time proportional to the length of the clause removed. Thus the entire algorithm takes $O(|f|)$ time where $|f|$ is the size of the original formula. Since each variable appears at most twice in f, $|f| \leq 2n$. Thus the algorithm takes $O(n)$ time proving the following theorem.

Theorem 2: *Given a $(\leq n, 2)$ CNF formula with n variables, an assignment that satisfies the maximum number of clauses of the formula can be found in $O(n)$ time.*

Corollary 3: *Given a $(\leq n, 2)$ CNF formula with n variables, it can be verified in $O(n)$ time whether or not it is satisfiable.*

4 Algorithm for $(\leq n, 3)$-MAXSAT

In this section, we address the question of solving the MAXSAT problem exactly in less than 2^n steps. Though considerable progress has been made towards the analogous question for solving SAT (3-SAT to be specific) exactly (see, for example [9], [7], [6]), we know of no algorithm taking less than 2^n steps for solving the MAXSAT problem even when there are at most two variables in each clause. Typically efficient exact algorithms for the 3-SAT problem are branching algorithms, cleverly pruning branches that falsify any particular clause. Such an approach does not generalize to MAXSAT simply because even the best assignment may falsify some clauses. Here, we give a branching algorithm to find the maximum number of clauses that can be satisfied in a given $(\leq n, 3)$ formula in less than 2^n steps. In each branch, some variables are set and the algorithm is recursively called with the reduced formula. Finally we pick the assignment that satisfies the most number of clauses in the branches explored.

If every clause contains at least three variables, then the reduced formula is satisfiable[10]. Otherwise there is a clause with at most two literals. Suppose there is a clause C with two literals $\{a, b\}$. Out of the four possible settings for a and b, abandon the branch where both a and b are false unless the other (possibly) two appearances of both a and b are $\bar{a}$ and $\bar{b}$. For suppose, there is an occurrence of a outside C, and both a and b are set to false. The literal $\bar{a}$ appears in at most one clause in the formula. By flipping a to true, that clause *may* be falsified, but the clause C will be satisfied. Thus setting a or b to true in this case gives at least as good an assignment than setting them both to false.

If both appearances of the variable a and b are as $\bar{a}$ and $\bar{b}$ outside C, then abandon the branch where both a and b are true. Thus, we can eliminate the two variables a and b by doing only a three way branching and so the recurrence for the number of nodes in the branching tree is $T(n) \leq 3T(n-2) + 1$ in this case.

Suppose there are no clauses with two literals. Then there must be some with only one literal. Let $C = \{a\}$ be such a clause. As long as there is at most one occurrence of $\bar{a}$ outside C, setting a to true gives as good an assignment than setting it to false. So consider the case when the variable a appears as $\bar{a}$ in two different clauses U and V. If both U and V are unit clauses, then simply set a to false. Otherwise, let X be the set of literals other than $\bar{a}$ appearing in the clauses U and V. As there are no 2 literal clauses, $|X| \geq 2$. Do a two way branching as follows: In one branch, set all literals in X as well as a to false, and in the other set a to true. If the best assignment has a false, but some of the literals of X true, then by flipping a to true, we gain the clause C and may loose at most one other, and so we obtain at least as good an assignment. Thus one of these branches will find an assignment that satisfies the maximum number of clauses. The recurrence for the number of nodes in the branching tree in this case turns out to be $T(n) \leq 1 + T(n-1) + T(n-3)$.

A concise description of the algorithm in recursive form is given in the next page.

Note that though in the description, the output obtained is only the maximum number of clauses satisfiable, the algorithm actually also constructs the assignment that realizes that maximum. As the dominating branching step is the one that gets rid of two variables through a three way branch, it follows that the total number of nodes in the branching tree is at most $3^{n/2}$. Furthermore, to move from one node to another node of the branching tree, at most $|f|$ steps are required. Since every variable appears at most thrice, $|f| \leq 3n$. Thus we have

Theorem 3: *Given a $(\leq n, 3)$ formula f, an assignment that satisfies the maximum number of clauses of the formula can be found in $O(3^{n/2}n)$ steps.*

```
MAXSAT (f)

input A CNF formula $f$ with $m$ clauses on $n$ variables, with each variable appearing
    in at most three clauses.

output The maximum number of clauses that can be satisfied by any assignment.

begin
    If every clause has at least three variables, then output $m$ and halt (use the
    perfect matching algorithm of Tovey[10] to get a satisfying assignment.)

    Otherwise, if there is a 2 literal clause $\{a, b\}$ then
    if $a$ or $b$ appears (in the same form) in some other clause of $f$,
    then output max $(MAXSAT(f_{1,0}), MAXSAT(f_{0,1}), MAXSAT(f_{1,1}))$ where
    $f_{x,y}$ is the reduced formula obtained by setting $a = x$ and $b = y$.

    Otherwise, { all the other appearances of $a$ and $b$ are as $\bar{a}$ and $\bar{b}$ }
    output max $(MAXSAT(f_{1,0}), MAXSAT(f_{0,0}), MAXSAT(f_{0,1}))$

    Otherwise (there are no 2 literal clauses, and so there must be unit clauses.)
    pick a unit clause $\{a\}$. Again if $\bar{a}$ appears at most once in $f$, then set $a$ to true,
    and call MAXSAT with the reduced formula.
    Otherwise, if $\bar{a}$ appears only in unit clauses, then set $a$ to false and call MAXSAT
    with the reduced formula.

    Otherwise, let $X$ be the set of literals other than $\bar{a}$ appearing in the clauses in
    which $\bar{a}$ appears. Then output max $(MAXSAT(f_{a=0,X=0}), MAXSAT(f_{a=1}))$
    where $f_{a=0,X=0}$ is the reduced formula obtained by setting $a$ and all the literals
    of $X$ false, and $f_{a=1}$ is the reduced formula obtained by setting $a$ to true.

end
```

5 Conclusions

We have shown that the smallest integer s such that $(2, s)$-MAXSAT is NP-complete is 3 by showing that $(2, 3)$-MAXSAT is NP-complete and $(\leq n, 2)$-MAXSAT can be solved in linear time. We also developed an efficient exact algorithm for the $(\leq n, 3)$-MAXSAT that takes much less than 2^n steps. Arguing through a few more cases, we can find an asymptotically better algorithm for this $((\leq n, 3))$ version of MAXSAT. However, we believe that obtaining an exact algorithm for the general MAXSAT (even for MAX2SAT) taking less than 2^n steps is a challenging open problem.

References

[1] M. R. Garey and D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-completeness", Freeman and Company 1979.

[2] M. R. Garey, D. S. Johnson and L. Stockmeyer, "Some simplified NP-complete graph problems", *Theoretical Computer Science* **1** (1976) 237-267.

[3] B. Jaumard and B. Simeone, "On the complexity of the Maximum Satisfiability problem for horn formulas", *Information Processing Letters* **26** (1987/88) 1-4.

[4] R. Kohli, R. Krishnamurti and P. Mirchandani, "The Minimum Satisfiability Problem", SIAM J. Discrete Mathematics **7** (2) (1984) 275-283.

[5] J. Kratochvil, P. Savick and Z. Tuza, "One more occurrence of variables makes satisfiability jump from trivial to NP-complete", *SIAM J. on Computing* **22(1)** (1993) 203-210.

[6] O. Kullmann,, " A systematical approach to 3-SAT decision yielding 3-SAT-decision in less than 1.5045^n steps", manuscript (available from kullmann@mi.informatik.uni-frankfurt.de, 1995).

[7] B. Monien and E. Speckenmeyer, "Solving satisfiability in less than 2^n steps", *Discrete Applied Mathematics* **10** (1985) 287-295.

[8] C. H. Papadimitriou, "The Euclidean traveling salesman problem is NP-complete," *Theoretical Computer Science* **4** (1977) 237-244.

[9] I. Schiermeyer, "Solving 3-satisfiability in less than 1.579^n steps, *Proceedings of the 6th Workshop in Computer Science Logic*, Springer Verlag (1993) 379-394.

[10] C. A. Tovey, " A simplified NP-complete satisfiability problem", *Discrete Applied Mathematics* **8** (1984) 85-89.