Discrete Applied Mathematics 8 (1984) 85-89
North-Holland

A SIMPLIFIED NP-COMPLETE SATISFIABILITY PROBLEM

Craig A. TOVEY*
Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Received 15 October 1982

3-SAT is NP-complete when restricted to instances where each variable appears in at most four clauses. When no variable appears in more than three clauses, 3-SAT is trivial and SAT is NP-complete. When no variable appears in more than two clauses, SAT may be solved in linear time.

1. Introduction

Cook [1] has shown that 3-SAT, the Boolean satisfiability problem restricted to instances with exactly three variables per clause, is NP-complete. This is a tightest possible restriction on the number of variables in a clause because as Even et al. [2] demonstrate, 2-SAT is in P. Horowitz and Sahni [5] point up the importance of finding the strongest possible restrictions under which a problem remains NP-complete. First, this can help clarify the interesting boundary between problems known to be in P and those that are not. Second, it can make it easier to establish the NP-completeness of new problems by allowing easier transformations. (For a comprehensive treatment of the subject, see [3].)

To prove the Euclidean travelling salesman problem NP-hard, Papadimitriou [6] first reduces 3-SAT to 3-SAT where each variable appears in at most five clauses. The question arises, are any further reductions in this direction possible? In this note we show that 3-SAT remains NP-complete even when each variable appears at most four times. Let r,s-SAT denote the class of instances with exactly r variables per clause and at most s occurrences per variable. We prove the 3,4-SAT result to be the strongest possible and show that 3,3-SAT is in fact trivial. In addition we show that the Boolean satisfiability problem is solvable in linear time if no variable appears more than twice, regardless of the number of variables per clause. All Boolean expressions are taken to be in conjunctive normal form with no repeated variables in a clause.

2. The reduction

Start with any 3-SAT instance. For each variable x which appears in more than

*The author was supported by the New Faculty Research Development Program of the Georgia Institute of Technology.

three clauses (by 'appears' we mean that it or its complement is in the clause) perform the following procedure: Suppose x appears in k clauses. Create k new variables $x_1, \ldots, x_k$ and replace the ith occurrence of x with x_i, $i = 1, \ldots, k$. Append the clause $\{x_i \vee \bar{x}_{i+1}\}$ for $i = 1, \ldots, k-1$ and the clause $\{x_k \vee \bar{x}_1\}$.

In the new instance, the clause $\{x_i \vee \bar{x}_{i+1}\}$ implies that if x_i is false, x_{i+1} must be false as well. The cyclic structure of the clauses therefore forces the x_i to be either all true or all false, so the new instance is satisfiable if the original one is. Moreover the transformation requires polynomial time. We have proved:

Theorem 2.1. *Boolean satisfiability is NP-complete when restricted to instances with* 2 *or* 3 *variables per clause and at most* 3 *occurrences per variable.*

An amusing corollary is:

Corollary 2.2. *For any* $s \geq 3$, *either* (i) *every Boolean expression with exactly* 3 *variables per clause and no more than s occurrences per variable, is satisfiable, or* (ii) 3,s-SAT *is NP-complete.*

Proof. Suppose (i) does not hold, so an unsatisfiable expression in the variables $x, y, z, \ldots$ exists. Without loss of generality, the first clause of the expression includes an 'x', uncomplemented. Let B denote the rest of the expression; clearly we may assume that B is satisfiable. B now has the properties, that x appears at most $s-1$ times, and that it can only be satisfied when x is false.

Now consider an arbitrary 3-SAT instance and perform the procedure in the construction from Theorem 2.1. For the ith clause, $\{a \vee b\}$ containing two variables, append an ith copy of B using variables $x_i, y_i, z_i, \ldots$, and change the clause to $\{a \vee b \vee x_i\}$. So if (i) is false, then (ii) is true. Note that (i) and (ii) cannot both be true unless P = NP.

Theorem 2.3. 3,4-SAT *is NP-complete.*

Proof. The only thing lacking in the construction from Theorem 2.1 is that the clauses $(x_i \vee \bar{x}_{i+1})$ contain only two variables. For each such clause, introduce a new variable y_i, so that the clause becomes $(x_i \vee \bar{x}_{i+1} \vee \bar{y}_i)$. Now note that we can force each y_i to be true by means of the clauses below in which y_i appears only three times. The construction is, we suspect, as small as possible. The first three clauses require y_i to be true if any of the pairs a_j, b_j are both false; the other ten clauses force this to happen:

$$\{y \vee a_j \vee b_j\}, \quad j = 1, \ldots, 3,$$
$$\{d_j \vee \bar{a}_j \vee \bar{b}_j\}, \quad \{d_j \vee \bar{a}_j \vee b_j\}, \quad \{d_j \vee a_j \vee \bar{b}_j\}, \quad j = 1, \ldots, 3,$$
$$\{\bar{d}_1 \vee \bar{d}_2 \vee \bar{d}_3\}.$$

In the actual reduction, there would be a copy of the above for each y_i; we have

suppressed the 'i' subscript of each variable for readability. If the original 3-SAT instance has m clauses, the 3,4 instance will have $m+3m+39m=43m$ clauses.

Corollary 2.2 and the algorithm in Section 3 demonstrate that there is no intermediate level of complexity in the 3,s-SAT problems between the virtually trivial and the NP-complete. The only unresolved case is 3,3-SAT.

Theorem 2.4. *Every instance of r,r-SAT is satisfiable.*

Proof. Denote the variables by $x_1, x_2, \ldots, x_n$ and the clauses by $c_1, \ldots, c_m$ where $m \leq n$ (and $m=n$ only if each variable appears in exactly r clauses). Let $A = A_1, \ldots, A_m$ denote the family of (not necessarily distinct) sets of the x_i defined by

$$A_i = \{x_j \mid x_j \in c_i \text{ or } \bar{x}_j \in c_i\}.$$

Consider any union of k of the sets A_i. Since each A_i contains r distinct elements and no x_j is contained in more than r sets, the union contains at least k distinct elements. Then by the Philip Hall Theorem [4] there exists a system of distinct representatives of A. That is, there exists an injection from the clauses c_i to the variables x_j such that each clause contains the variable (or the complement of the variable) it is mapped to. It is trivial given such a mapping to satisfy each clause, and hence the r,r-SAT instance.

Conjecture 2.5. If $s \leq 2^{r-1}-1$, then every instance of r,s-SAT is satisfiable.

The intuitive meaning of the $2^{r-1}-1$ term is the minimal number of clauses of size r required to force a variable to be true, if the 'forcing' variables are restricted to half of their possible truth values. For instance, only 7 of the 16 truth combinations of $y_1, \ldots, y_4$ constrain z to one value in the clauses $\{z \vee y_1 \vee y_2\}\{z \vee y_3 \vee y_4\}$. In the 3,4 case, the analogous fraction is 37/64 which is more than 1/2, so the 'wave' of implications from the y's to z gets stronger and it is possible to force a contradiction.

3. Polynomial algorithm for the 2-occurrence case

In this section we show that the satisfiability problem is in P when no variable appears more than twice.

Algorithm 3.1. *Step 1.*

While there are clauses containing a single variable, **do**
Begin Select such a clause.
Assign to the variable in the clause the truth value necessary to make that clause true.

Simplify or remove the other clause where the variable appears.
If a clause becomes false, stop: the instance has no solution.

End

Step 2. Stop, the instance has a solution.

Note that a properly maintained list of the clauses containing one variable enables the algorithm to run in linear time (on a RAM). Step 2 requires some justification. We now outline a polynomial time constructive algorithm which is guaranteed to work. Define 'flipping' a variable to mean replacing it by its complement wherever it appears. If there is a variable which is uncomplemented in all (one or two) of its occurrences, assign it the value 'true' and remove all clauses containing it. If a variable is always complemented, flip it and do the same. We may now assume that the expression satisfies the requirements of Lemma 3.2.

Lemma 3.2. *An instance of* SAT *is satisfiable if:*
(i) *all of its clauses contain more than one variable, and*
(ii) *each variable appears once complemented and once uncomplemented.*

Proof. We show that after suitable flipping, assigning the value 'true' to all variables will satisfy the expression. If there are no clauses in which all variables are complemented, we are done. For ease we call these 'all-comp' clauses. Otherwise we look for a variable in an 'all-comp' clause which when flipped will reduce the number of all-comp clauses. Such a variable will exist unless every variable in an all-comp clause, in its other appearance, is the only uncomplemented variable in its clause. If this is the case, arbitrarily select one variable from an all-comp clause and flip it. Now consider the other variables (there must be at least one) in the other clause of the flipped variable. By the same reasoning, we can reduce the number of all-comp clauses by flipping one of them, unless each is the only uncomplemented variable in its other clause. In this case we again flip one variable arbitrarily and proceed. Each clause we proceed to in this way contains just one uncomplemented variable, or else the process terminates. But since we always get there via the uncomplemented variable, and no variable appears in more than two clauses, we can never encounter the same clause twice. Therefore the process must terminate with a reduction in the number of all-comp clauses.

Acknowledgements

The question addressed in this paper was brought to the author's attention by Christos Papadimitriou. The idea of finding an SDR in the proof of Theorem 2.4 is due to Adam Rosenberg. The author also wishes to acknowledge the helpful suggestions of the referee's, clarifying the statement of Corollary 2.2 and the proofs of Theorem 2.3 and Lemma 3.2.

References

[1] S.A. Cook, The complexity of theorem-proving procedures, Proc. 3rd Ann. ACM Symp. on Theory of Computing (Assoc. Comput. Mach., New York, 1971) 151–158.

[2] S. Even, A. Itai and A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM J. Comput. 5 (1976) 691–703.

[3] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness (W.H. Freeman, San Francisco, 1979).

[4] P. Hall, On representations of subsets, J. London Math. Soc. 10 (1935) 26–30.

[5] E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms (Computer Science Press, Rockville, MD, 1978).

[6] C.H. Papadimitriou, The Euclidean traveling salesman problem is NP-complete, Theor. Comput. Sci. 4 (1977) 237–244.