X3SAT is Decidable in Time $O(2^{n/5})$

Stefan Porschen, Bert Randerath and Ewald Speckenmeyer
Institut für Informatik, Universität zu Köln, Germany
E-Mail {porschen,randerath,esp}@informatik.uni-koeln.de

February 6, 2002

Abstract

We study the problem exact 3-SAT (X3SAT) for propositional formulas in conjunctive normal form (CNF) $C = c_1 \vee \cdots \vee c_m$ each clause c_i containing at most three literals. Then $C \in$ X3SAT iff there is a truth assignment $t : V_n \to \{0, 1\}$, s.t. exactly one literal in each clause is set to true (1). The corresponding decision problem also is called X3SAT. As is well known problem X3SAT is NP-complete [5] and also XSAT as a generalization is NP-complete. By adapting and improving branching techniques like that in [4] we are able to prove that XSAT $\in O(2^{n/5})$ thereby improving a more recent result found in [1] stating XSAT $\in O(2^{0.2072n})$. As outlook we announce that it can be proven X3SAT $\in O(2^{n/6})$ by a more careful case analysis.

Key Words: exact satisfiability; branching techniques; formula graph; perfect matching

1 Introduction

Let V be a set of $n \in \mathbb{N}$ propositional variables, denoted as $a, b, \ldots$ Each variable a induces a positive literal (variable a) or a negative literal (negated variable: $\overline{a}$). The problem exact satisfiability (XSAT) considers propositional formulas C in conjunctive normal form (CNF) over V, i.e., C is a disjunction of a finite number of clauses $c_i : C = c_1 \vee \cdots \vee c_m, m \in \mathbb{N}$, and each clause c_i is a conjunction of a finite number of literals over V, where it is assumed that no literal appears twice in the same clause. The language XSAT for such C is then defined by: $C \in$ XSAT iff there is a truth assignment $t : V \to \{0, 1\}$, s.t. exactly one literal in each clause is set to truth value true (1). Thus XSAT is the decision problem to decide whether a given such CNF formula is in XSAT.

In this paper we study a specialization of the general problem XSAT, namely X3SAT, which is XSAT restricted to formulas in which any clause contains at most three literals, let this set of formulas be denoted 3-CNF. As is well known X3SAT is NP-complete [5] and also XSAT as a generalization is NP-complete.
For solving XSAT in [4] an algorithm is presented having time bound $O(2^{0.24404n})$. More recently, Drori and Peleg found the result X3SAT $\in O(2^{0.2072})$ [1, 2]. By branching techniques like that in [4] and exploiting a suitable graph theoretic formulation which allows to translate the solvation of a main branch into a perfect matching problem we now give an algorithm showing X3SAT $\in O(2^{0.18674n})$.

2 The concept of formula graphs

First we describe the translation of special formulas into a graph theoretic framework, s.t. the test for exact satisfiability is equivalent to the test whether the corresponding formula graph has a perfect matching.
Consider the special case where C is a formula having the property that every clause $c \in C$ that contains a variable occuring at least three times in C also contains a variable occuring once in C, i.e., occuring only in c. Here it is to be understood, that it is irrelevant, whether the considered variable appears as a positive or negative literal.
Then in a first step delete all the variables from C occuring ≥ 3 times, resulting in a formula containing a 2-clause for each such clause and all remaining 3-clauses only contain variables appearing at most twice in C. This first step does not affect the exact satisfiability state of C, because for each such clause one can rest on its contained unique variable, which thus is independent from the rest of the formula. Advantageous of the remaining formula, also denoted C, is the possibility in a second step to construct a corresponding *formula graph* G_C as follows: Suppose C contains only positive literals. If there is no clause containing a unique variable in C, then for each clause c introduce a node labeled by c. This delivers the node set $V(G_C)$. For any variable in C (by construction appearing in exactly two clauses of C) join the two clause nodes containing it by a simple edge labeled by this variable. This delivers the edge set $E(G_C)$ for that case.
If there is any clause in C containing a unique variable, then first carry out the construction just described leading to node set V_1 and edge set E_1. Second, make a copy of both sets resulting to V_2 and E_2. To complete the edge set introduce an edge joining the copies of each node labeled by a clause containing a unique variable, and label this edge by the corresponding variable, resulting to edge set E_{12}. Now define $G_C = (V, E)$ with $V := V_1 \cup V_2, E := E_1 \cup E_2 \cup E_{12}$ as disjoint unions. It is obvious that the restriction of maximal appearences of any variable to two is crucial for the graph not to be a hypergraph. An analogue of this matching approach first was introduced in [4].

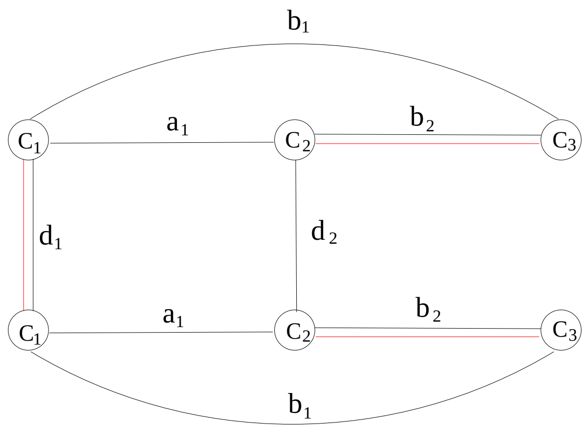

Figure 1: Formula graph G_C corresponding to C, matching edges are coulored red.

Of course, in general in C there may appear negative literals, too. But one can show that by applying the resolution principle (see e.g. [3]) that it is possible to eliminate negative literals. By this of course there may occure larger clauses, but this can be handled by the matching approach, which in any case is polynomially bounded and will only lead to an additive term of the overall time bound.

The translation of a formula C into its formula graph G_C may be illustrated by the following example: Let after the first step the resulting formula be $C = c_1 \vee c_2 \vee c_3$, where

$$\begin{aligned} c_1 &= a_1 \vee b_1 \vee d_1 \\ c_2 &= a_1 \vee b_2 \vee d_2 \\ c_3 &= b_1 \vee b_2 \end{aligned}$$

The corresponding formula graph is shown in Figure 1. To summarize, we have:

Lemma 2.1 *Let C be a formula as above then G_C as constructed above is a well defined simple, connected, node and edge labeled graph. Moreover, $C \in$ X3SAT iff G_C contains a perfect matching. If there is a perfect matching M contained in G_C, then for each $e \in M$, set the corresponding (variable-)label to true in order to exact satisfy C.*

3 A branching procedure leading to $O(2^{n/5})$

Now we describe the general branching procedure which is based on the following consideration: For a clause $c = a_1 \vee a_2 \vee a_3$ we have the branches $a_1 = 0$ leading to a subproblem of $n-1$ variables and $a_1 = 1$ leading to a subproblem of $n-3$ variables, because a_2, a_3 are forced to be set to false (0). That is, the upper bound $T(n)$ for the number of recursive calls the following branching procedure performs until a decision is estimated by $T(n) \leq T(n-3) + T(n-1) + 1$. Thus $T(n)$ is upper bounded by the maximum over all $T_i(n)$ where i indexes all branches carried out in the procedure.

Procedure Exact3SAT(C) :
comment: $C = c_1 \vee \cdots \vee c_m \in 3-CNF$ over n variables
comment: in the following it is assumed that the same steps are also carried out if all variables are negated simultaneously in the corresponding clauses
comment: simplification steps:
if C contains no more clauses **then** C exact satisfiable
if C contains the empty clause **then** C not exact satisfiable
remove from C all duplicate clauses
if C contains a variable b which occurs only negated **then** substitute b for $\overline{b}$
if C contains a one literal clause $c = a, a \in V$ **then** evaluate C according to $a \leftarrow 1$
if C contains a clause $c = a \vee b$ **then** substitute $\overline{b}$ for a
if C contains clauses $c_1 = a \vee b \vee e$ and $c_2 = a \vee b \vee f$ **then** $e \leftarrow f$
if C contains clauses $c_1 = a \vee b \vee e$ and $c_2 = a \vee \overline{b} \vee f$ **then** $a \leftarrow 0, f \leftarrow b, e \leftarrow \overline{b}$
if C contains clauses $c_1 = a \vee b \vee e$ and $c_2 = \overline{a} \vee \overline{b} \vee f$ **then** $e \leftarrow f \leftarrow 0, a \leftarrow \overline{b}$
if each clause in C which contains a variable occuring ≥ 3 times in C also contains a variable occuring exactly once in C **then** remove from C all variables occuring ≥ 3 times and carry out matching procedure described above
comment: main branching:
if C not exact satisfied **then** choose clauses

$$\begin{aligned} c &= a \vee b \vee d \\ c' &= a \vee e \vee f \\ c'' &= \overset{(-)}{a} \vee g \vee h \end{aligned}$$

comment: we have to distinguish the following two subcases:
if $a \in c''$: **then**
i) evaluate C according to $a \leftarrow 1 \wedge b \leftarrow e \leftarrow d \leftarrow f \leftarrow g \leftarrow h \leftarrow 0$
comment: eliminating $(n-7)$ variables
ii) evaluate C according to $a \leftarrow 0 \wedge b \leftarrow \overline{d}, e \leftarrow \overline{f}, g \leftarrow \overline{h}$
comment: eliminating $(n-4)$ variables
else
comment: $\overline{a} \in c''$:
i) evaluate C according to $a \leftarrow 1 \wedge b \leftarrow d \leftarrow e \leftarrow f \leftarrow 0, g \leftarrow \overline{h}$

comment: eliminating $(n-6)$ variables
ii) evaluate C according to $a \leftarrow 0 \wedge g \leftarrow h \leftarrow 0, b \leftarrow \overline{d}, e \leftarrow \overline{f}$
comment: eliminating $(n-5)$ variables
fi
C exact satisfiable iff C according to i) or ii) exact satisfiable
fi

A decision, whether $C \in$ X3SAT thus is achieved after $T(n) \leq \max\{T(n-7)+T(n-4)+1, T(n-6)+T(n-5)+1\}$ recursive iterations of the above procedure. From this and lemma 2.1 it is not difficult to derive:

Theorem 3.1 *Let C an arbitrary 3-CNF formula over n variables, then within time $O(2^{0.18674n})$ it is decidable, whether $C \in$ X3SAT.*

Of course any improvement within this approach can only be achieved by steadily increasing the branching depth carefully distinguishing new cases. We have strong evidence that doing so will lead to upper bound $O(2^{n/6})$ for X3SAT.

References

[1] L. Drori and D. Peleg, Faster Solutions for Exact Hitting and Exact SAT. Technical Report MCS99-15 of the Weizmann Institute of Science, Israel, September, 1999.

[2] L. Drori and D. Peleg, Faster Exact Solutions for Some NP-Hard Problems, in: "J. Nesetril (Ed.), Proceedings of the European Symposium on Algorithms (ESA 1999), Prague, Czech Republic," Lect. Notes in Comp. Science, vol. 1643, pp. 450-461, Springer-Verlag, Berlin, 1999.

[3] H. Kleine Büning and T. Lettmann, "Propositional logic: deduction and algorithms," Cambridge Tracts in Theoretical Computer Science, No. 48, Cambridge University Press, Cambridge, 1999.

[4] B. Monien, E. Speckenmeyer, O. Vornberger, Upper Bounds for Covering Problems, Methods of Operations Research 43 (1981) 419-431.

[5] T. J. Schaefer, The complexity of satisfiability problems, in: "Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, San Diego, California, 1-3 May 1978," pp. 216-226, ACM, 1978.