Applications of General Exact Satisfiability in Propositional Logic Modelling

Vilhelm Dahllöf
vilda@ida.liu.se
Linköping University, Sweden

Propositional Logic Modelling

Propositional Logic is a useful modelling tool. For reasons of compactness and convenience a number of additions to the basic language have been proposed.

A *cardinality atom* (c-atom) is a constraint

$$h\{a_1, a_2 \ldots a_k\}j$$

s.t. at least h and no more than j of the literals $a_1, a_2 \ldots a_k$ are true.

Example: Graph Colourability

1. Put constraints on each variable such that it can only assume one colour
2. For each edge (u, v), make sure u and v get different colours

Assume there are k colours. Then each vertex gives rise to k (Boolean) variables $u_1, u_2 \ldots u_k$ and we get a c-atom

$1\{u_1, u_2 \ldots u_k\}1$

How to deal with c-atoms?

- Reduction to SAT common, however, for a c-atom containing n literals, more than n new variables are introduced (Liu and Truszczyński, 2003)

- Exact solvers have been experimentally studied without providing upper time bounds (East and Truszczyński, 2001)

- Incomplete solvers have been proposed (Liu and Truszczyński, 2003)

X_1SAT

Given a Boolean formula in CNF the EXACT SATISFIABILITY PROBLEM, X_1SAT, asks for an assignment to the variables s.t. *exactly* one literal be true in each clause.

- NP-complete as shown by Schaefer 1978
- Well-studied, solvable in $\mathcal{O}(1.1748^n)$ time, where $n = \#Vars$
- Very rich structure, many known polynomial-time pruning rules (a.k.a. canonical rules)
- The case when every variable occurs at most twice is polynomial time decidable by matching techniques.

New Problem

Given a Boolean formula in CNF the General Exact Satisfiability Problem, $\mathrm{X}_i\mathrm{SAT}$, asks for an assignment to the variables s.t. *exactly* i literals be true in each clause.

- $\mathrm{X}_1\mathrm{SAT}$, $\mathrm{X}_2\mathrm{SAT}$, $\mathrm{X}_3\mathrm{SAT}$ etc., are called *sublanguages* of $\mathrm{X}_i\mathrm{SAT}$.
- Still many polynomial time pruning rules.
- Each sublanguage is polynomial-time reducible to $\mathrm{X}_1\mathrm{SAT}$.

$1\{a_1, a_2 \ldots a_k\}1$ corresponds to a clause $(a_1 \vee a_2 \ldots a_k)$ in an instance of $\mathrm{X}_1\mathrm{SAT}$.

$2\{a_1, a_2 \ldots a_k\}2$ corresponds to a clause $(a_1 \vee a_2 \ldots a_k)$ in an instance of $\mathrm{X}_2\mathrm{SAT}$.

$i\{a_1, a_2 \ldots a_k\}i$ corresponds to a clause $(a_1 \vee a_2 \ldots a_k)$ in an instance of $\mathrm{X}_i\mathrm{SAT}$.

Any c-atom $h\{a_1, a_2 \ldots a_k\}i$ can be translated to the clause $(a_1 \vee a_2 \ldots a_k \vee b_1 \vee b_2 \ldots b_{i-h})$, where $b_1, b_2 \ldots b_{i-h}$ are fresh variables.

A *mixed* instance of $\mathrm{X}_i\mathrm{SAT}$ contains different types of clauses, but contains no clause that needs more than i true literals.

Properties of $\mathrm{X}_i\mathrm{SAT}$

$\mathrm{X}_i\mathrm{SAT}$ has a rich structure in the sense that there are numerous canonical rules:

- Pick a clause $(a \vee b)^{\mathrm{X}_1\mathrm{SAT}}$, remove it and let $F := F(a/\bar{b})$
- If there are two clauses $x = (A \vee b \vee c)^{\mathrm{X}_2\mathrm{SAT}}$ and $y = (A \vee \bar{b} \vee \bar{c})^{\mathrm{X}_2\mathrm{SAT}}$, then let $F := F(b/\bar{c})$
- Pick an $\mathrm{X}_i\mathrm{SAT}$ clause with $i + k$ singletons. Remove k singletons. (A singleton is a variable that occurs only in one clause).

Exact Poly-space Algorithms

DPLL-style branching using one variable at a time does not seem to be a good idea.

Note that for a clause $(a_1 \vee a_2 \ldots a_k)$, setting i literals to true means that the other $k - i$ literals will have to be set to false. Hence we only need to test $\binom{k}{i}$ possibilities for any clause.

Using this scheme we trivially end up with an $\mathcal{O}\left(\binom{k}{i}^{n/k}\right)$ time algorithm.

Unlike the ordinary SAT problem, long clauses are not a problem.

Worst clause length

Question: for a given i, which k maximizes $\binom{k}{i}^{1/k}$? Looking at X_1SAT and X_2SAT we see that $k = 2i$ is wrong answer.

Clause Length	X_1SAT	X_2SAT
1	*	*
2	*	*
3	$\binom{3}{1}^{n/3} = 3^{n/3} < 1.45^n$	$\binom{3}{2}^{n/3} = 3^{n/3} < 1.45^n$
4	$\binom{4}{1}^{n/4} = 4^{n/4} < 1.42^n$	$\binom{4}{2}^{n/4} = 6^{n/4} < 1.57^n$
5	$\binom{5}{1}^{n/5} = 5^{n/5} < 1.38^n$	$\binom{5}{2}^{n/5} = 10^{n/5} < 1.59^n$
6	$\binom{6}{1}^{n/6} = 6^{n/6} < 1.35^n$	$\binom{6}{2}^{n/6} = 15^{n/6} < 1.58^n$

An instance of $\textsc{x}_i\textsc{sat}$ consists of a number of clauses. These clauses may differ in length (and differ in the number of true literals needed.) We say that among these clauses, the ones with the smallest $(\binom{k}{i}^{1/k})$ are *preferable*.

A variable that occurs in more than 2 clauses is *heavy*.

Consider the two clauses

$x = (a \vee b \vee c \vee d \vee e)^{\mathrm{X_2SAT}}$ and $y = (e \vee f \vee g \vee h \vee i)^{\mathrm{X_2SAT}}$

Note that when testing the assignment $a = 1, b = 0, c = 0, d = 0, e = 1$ y becomes

$(f \vee g \vee h \vee i)^{\mathrm{X_1SAT}}$

A Poly-space Exact Algorithm $D_i(F)$

1. If there is an X_iSAT clause with $i + k$ singletons then remove k of those singletons;

2. If 3/5 or less of the variables are heavy then try all possible assignments to these variables. For each such partial assignment to the variables, transform the instance to an X_1SAT instance, then use matching techniques to decide whether there is a model. Answer 'Yes' if such a model is found and 'No' otherwise;

3. Pick a preferable clause y with as few singletons as possible and make $\binom{|y|}{i}$ recursive calls, each call having the form $D_i(F(a_1/true; a_2/true \ldots a_i/true; a_{i+1}/false \ldots))$.

Time Complexity

The running time of D_i is in $o(2^n)$.

For $\textsc{x}_2\textsc{sat}$, $\textsc{x}_3\textsc{sat}$ and $\textsc{x}_4\textsc{sat}$, the running time of $D(i)$ is in $\mathcal{O}\left(1.5157^n\right), \mathcal{O}\left(1.6202^n\right)$ and $\mathcal{O}\left(1.6844^n\right)$, respectively.

Using a dedicated algorithm built on D_i, it can be shown that $\textsc{x}_2\textsc{sat}$ is solvable in time $\mathcal{O}\left(1.4511^n\right)$.

An old result by Shroeppel and Shamir (1981) shows that $\textsc{x}_1\textsc{sat}$ is solvable in time $\mathcal{O}\left(2^{n/2}\right) \subset \mathcal{O}\left(1.4143^n\right)$ and space $\mathcal{O}\left(2^{n/4}\right) \subset \mathcal{O}\left(1.1893^n\right)$. Sursprisingly, this also holds for the general $\textsc{x}_i\textsc{sat}$ problem, regardless of i.

Conclusions

c-atoms quite nicely transform to clauses in an $\mathrm{X}_i\mathrm{SAT}$ instance.

$\mathrm{X}_i\mathrm{SAT}$ is solvable in time $\mathcal{O}(1.4143^n)$ if exponential space is allowed. For polynomial space we have proposed a $o(2^n)$ time algorithm.

$\mathrm{X}_i\mathrm{SAT}$ seems to have a more benign structure than SAT.

$\mathrm{X}_i\mathrm{SAT}$ has a rich structure with many polynomial time pruning rules and it should be possible to construct even faster algorithms.

Due to the low-exponential time exact algs for $\mathrm{X}_1\mathrm{SAT}$, approximative and heuristic methods are not well-studied. However, this could be of interest for $\mathrm{X}_i\mathrm{SAT}$.