Satisfiability of Boolean Formulas - Combinatorics and Algorithms
Autumn 2008

Special Exercises

Regulations:

- There will be a total of four special exercise sets during this semester.
- You are expected to solve them carefully and then write a nice and complete exposition of your solutions using LaTeX.
- You are welcome to discuss the tasks with your colleagues, but we expect each of you to hand in your own, individual writeup.
- Your solutions will be graded. The three highest out of your four achieved grades will account for 10% of your final grade for the course each (so 30% of the grade in total).

Special Exercise Set 2

Due date: Friday, November 7, 2008 (at the beginning of the 10:15 lecture)

Problem 1 [4 points] Different Clause Types

Let us call a CNF formula F *allowed*, if every clause $C \in F$ is either

- a 3-clause with 3 positive literals, e.g. $\{x, y, z\}$
- a 3-clause with 3 negative literals, e.g. $\{\bar{x}, \bar{y}, \bar{z}\}$
- a 2-clause with a negative and a positive literal, e.g. $\{x, \bar{y}\}$

Prove: For every allowed formula, there is an assignment that satisfies at least $(1 - \Phi^3)$-fraction of the clauses, where $\Phi = \frac{\sqrt{5}-1}{2}$ is the golden ratio conjugate.

Problem 2 [6 points] XSAT

Let F be a CNF formula and α some assignment to its variables. We say that α *exactly satisfies* F, if in each clause of F, exactly one literal evaluates to true under α. A CNF formula is called *exactly satisfiable* if it admits an assignment which exactly satisfies it.

a. Let F be an exactly satisfiable k-CNF formula, $k \geq 2$. Show that there are at least 2 satisfying (in the normal SAT sense!) assignments for F.

b. Improve on the result of task a. Give a tight bound $f(n, k) \in \mathbb{N}$ such that for $n \geq k \geq 2$, for every exactly satisfiable k-CNF formula F with n variables, the number of satisfying (in the normal SAT sense!) assignments for F is at least $f(n, k)$.
HINT: To show that your bound is tight, you have to exhibit for all $n \geq k \geq 2$ an exactly satisfiable k-CNF formula F which has n variables and exactly $f(n, k)$ satisfying assignments.

c. The *XSAT* problem is to recognize whether a given CNF formula F is exactly satisfiable. Demonstrate that SAT is at least as hard to solve as XSAT, i.e. give a polynomial-time algorithm which takes a CNF formula F and transforms it into another CNF formula F' such that F' is satisfiable if and only if F is exactly satisfiable.

d. Demonstrate that XSAT is at least as hard to solve as SAT, i.e. give a polynomial-time algorithm which takes a CNF formula F and transforms it into another CNF formula F' such that F' is exactly satisfiable if and only if F is satisfiable in the normal sense. For simplicity, let F be a 3-CNF.

Problem 3 [6 points] Tree-like 2-CNFs

Let F be a 2-CNF formula. Let the *variable-dependency graph* $G'[F]$ be defined as the graph with vertex set $\mathrm{vbl}(F)$ and in which there exists an edge between two variables $x, y \in \mathrm{vbl}(F)$ if $\exists C \in F : \mathrm{vbl}(C) = \{x, y\}$. We say that F is *tree-like*, if its variable-dependency graph is a tree and no two clauses are over the same variable set, i.e. for all $C, D \in F$ with $C \neq D$ we have $\mathrm{vbl}(C) \neq \mathrm{vbl}(D)$.

a. Show that a tree-like 2-CNF is satisfiable.

b. Let F be a tree-like 2-CNF and let $x \in \mathrm{vbl}(F)$ be any variable. Consider x to be the root vertex of $G'[F]$. Let us say that the *subtrees* of $G'[F]$ are the trees that we obtain by picking any vertex y and all of its descendants. Now recall the procedure `uc` in the script which performs unit clause reduction. Demonstrate that $G'[\mathrm{uc}(F^{[x \mapsto 1]})]$ consists of disjoint subtrees of $G'[F]$.

c. Give a recursive algorithm that counts the number satisfying assignments of a tree-like 2-CNF. What is the running time of your algorithm?

d. Give a polynomial-time algorithm for the same task using dynamic programming.
HINT: In case you do not know this concept, consider
`http://en.wikipedia.org/wiki/Dynamic_programming`
a reading assignment.