Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Emo Welzl, lecturer, and Robin Moser, teaching assistant

Monday, April 16, 2012

Satisfiability of Boolean Formulas Spring 2012
Special Assignment Set 2

- The solution is due on **Friday, May 4, 2012**. Please bring a print-out of your solution with you to the lecture. If you cannot attend (and please only then), you may alternatively send your solution as a PDF to `robin.moser@inf.ethz.ch`. We will send out a confirmation that we have received your file. Make sure you receive this confirmation within the day of the due date, otherwise complain timely.
- Please solve the exercises carefully and then write a nice and complete exposition of your solution using a computer, where we strongly recommend to use LaTeX. A tutorial can be found at `http://www.cadmo.ethz.ch/education/thesis/latex`.
- For geometric drawings that can easily be integrated into LaTeX documents, we recommend the drawing editor IPE, retrievable at `http://ipe7.sourceforge.net/` in source code and as an executable for Windows.
- You are welcome to discuss the tasks with your colleagues, but we expect each of you to hand in your own, individual write-up.
- There will be three special assignments this semester. Each of them will be graded and the average grade will contribute 30% to your final grade.
- This is a theory course, which means: if an exercise does not explicitly say "you do not need to prove your answer" or "justify intuitively", then a formal proof is **always** required.
- As with all exercises, the material covered in special assignments is relevant for the final exam.

Exercise 1 (Efficient Falsifiers for an NP-complete Problem) *(50 Points)*

Let the computational problem Bi-Subgraph be defined as follows. An instance is a pair (G, k) consisting of a graph $G = (V, E)$ on n vertices and a number $k \in \{1..n\}$ and it is a Yes-instance if and only if there exist vertex sets $A \dot\cup B \subseteq V$, both of size $k = |A| = |B|$ such that all edges $\{u, v\}$ with $u \in A, v \in B$ are present in E (a full bipartite subgraph).

(a) Express Bi-Subgraph as a CNF problem over $\mathcal{O}(n^2)$ variables and deduce that by standard machinery, there exists a 3-falsifier for Bi-Subgraph accepting proofs[1] of length $\mathcal{O}(n^2)$.

Hint: Use, e.g., variables $x_{i,j}$ (and $y_{i,j}$) representing whether vertex i is the j-th vertex of partite set A (or B, respectively).

(b) Prove that Bi-Subgraph also admits a 3-falsifier accepting proofs of length only $\mathcal{O}(n \log n)$ when hand-crafting the proof schema. For simplicity, you are allowed to assume that n is a power of 2.

Hint: The difficult part is to check that the sizes of A and B are correct. For this, you can for example (maybe you find a much better way than I did) use a divide-and-conquer approach to model a circuit that counts bits in a unary format. This nicely illustrates how CNF formulas can be viewed not only as combinatorial problems but also as a model of computation.

[1] to 'accept a proof of length $\mathcal{O}(f(n))$' means that the falsifier matches the definition in the lecture notes with the polynomial $\hat{p}_F(|w|)$ replaced by $\mathcal{O}(f(|w|))$, i.e. the falsifier must ignore all positions beyond bit $g(|w|)$ for some function $g \in \mathcal{O}(f(|w|))$.

Exercise 2 (Angels and Devils) *(50 Points)*

You recall that in Exercise 3.12, we established the tightness of Theorem 3.5 by proving that for certain formulas (for example a maximal satisfiable 2-CNF), if the right strategy is used for selecting violated clauses in each iteration of the loop, then the expected running time of Papadimitriou's algorithm is exactly n^2. In this task, we want to explore the power of the clause selection strategy in this game.

(a) Exhibit a strategy for selecting violated clauses in each iteration of the loop such that for a maximal satisfiable 2-CNF on variables V, and any arbitrary initial assignment $\alpha_0 \in \{0,1\}^V$, the expected running time $\mathcal{O}(n)$.

It looks like the clause selection strategy can make all the difference. A 'bad strategy' (which we use to call a *devil*) can force a running time of $\Theta(n^2)$, a 'good strategy' (which we can call, analogously, an *angel*), can make the algorithm reach a satisfying assignment in $\mathcal{O}(n)$ time. In the sequel we demonstrate that this is only because the maximal satisfiable 2-CNF offers the freedom of picking all possible clauses. For other formulas, even such with sufficiently many constraints to only admit a unique satisfying assignment, the clause selection rule does not make much difference.

(b) Consider the 2-CNF formula $F_n^\star$ on variables $V = \{x_1, x_2, \ldots, x_n\}$ which consists of the clauses $\{x_1, x_2\}$, $\{\bar{x}_1, x_2\}$, $\{x_1, \bar{x}_2\}$, as well as all clauses $\{\bar{x}_2, x_i\}$ for $3 \leq i \leq n$. Prove that even when starting from an initial assignment α_0 selected uniformly at random, the expected running time of the algorithm is $\Theta(n)$, no matter what clause selection strategy is used.

HINT: For any fixed selection rule, consider a run of the algorithm to be partitioned into phases, where each phase ends whenever the rule decides to select some clause $\{\bar{x}_2, x_i\}$ for $3 \leq i \leq n$. How many such phases are there and what is the expected time it takes for each phase to complete?

(c) Consider the 2-CNF formula $F_n^\sim$ on variables $V = \{x_1, x_2, \ldots, x_n\}$ which consists of the clauses $\{x_1, x_2\}$, $\{\bar{x}_1, x_2\}$, $\{x_1, \bar{x}_2\}$, as well as all clauses $\{\bar{x}_{i-1}, x_i\}$ for $3 \leq i \leq n$. Prove that even when starting from an initial assignment α_0 selected uniformly at random, the expected running time of the algorithm is $\Theta(n^2)$, no matter what clause selection strategy is used.

HINT: Again break down the steps executed into categories, this time not into contiguous phases, but into separate "processes" per variable. Note that each variable x_i with $i \geq 3$ which is set to zero in α_0 must be flipped at some point. Each attempt at flipping x_i can either solve the problem and "close the case" of variable x_i, or it causes a new zero to appear to the left. This new zero must be fixed first before any attempt at fixing x_i can be made. Fixing it can in turn cause new zeroes to appear further to the left. 'Blame' all zeroes so appearing on the original culprit x_i and then analyze how the number of mistakes each variable x_i is to blame for evolves over time. Model these numbers as independent symmetric reflecting random walks and deduce a bound on the expected total running time.

Open Problem: If you are interested in doing some actual research, try to find out *what* really characterizes formulas that behave like the example in (a), what characterizes formulas of the type in (b) and what characterizes formulas that behave like the example in (c). If we draw these dependency graphs of these formulas, the one in (b) looks like a star while the one from (c) looks like a long chain. In the star, all variables are "close" to the "core" of the formula which always contanis "good" clauses. In the chain, there are variables that are very far from any "good" clause (that is what we exploit for proving (b)). Since (a) contains both the star and the chain, it depends on the selection rule which of the two shapes we use for "playing the game". My guess would be that this is what it somehow depends on, but I do not really know.