CS 221: Computational Complexity

Problem Set 6 (Take-Home Final)

Assigned: Sun. Apr. 27, 2014 Due: Fri. May 9, 2014 (5 PM)

Exam Policies. You must work on this exam *alone*. The only references you may use are notes from lecture and section, the problems sets and solutions, and the Arora–Barak text. You may quote any result *proven* in class or in the text. Except on Problem 1, you may not use results that were only stated without proof (unless you provide justification of your own). No late days are permitted, and no credit will be given for work turned in after the deadline.

You must *type* your solutions. LaTeX, Microsoft Word, and plain ascii are all acceptable. Submit your solutions *via email* to `cs221-hw@seas.harvard.edu`. If you use LaTeX, please submit both the compiled file (`.pdf`) and the source (`.tex`). Please name your files `PS6-yourlastname.*`.

Problem 1. (Various relations) In this problem, you will describe everything that we know (from this course) about the pairwise relationships between the following statements. In doing so, you may use results stated without proof in class or in the texts. First, identify whether any of the statements below are known to be true or false. For the remaining statements, draw a directed graph showing all of the implications that follow from the results we have seen in class. (You do not need to consider negations of the statements. And if you have implications $A \Rightarrow B$ and $B \Rightarrow C$, you do not need to draw the implication $A \Rightarrow C$.) Briefly justify your answers.

1. **NEXP** = **PSPACE**.
2. There is a logspace mapping reduction from 3SAT to 2SAT.
3. The polynomial-time hierarchy collapses.
4. Deciding whether an arithmetic formula over $\mathbb{Z}$ is identically zero can be done by polynomial-sized boolean circuits.
5. (Exactly) counting the number of perfect matchings in a bipartite graph can be done by a probabilistic polynomial-time algorithm with **NP** oracle.
6. There is an interactive proof system for proving that a regular expression with exponentiation generates Σ^*.
7. Every language in **NP** can be decided by boolean circuits of polynomial size and polylogarithmic depth.

Problem 2. (2-CSPs and 2-Query PCPs)

1. Give a (1/4)-approximation algorithm for Max-2CSP.
2. Show that there are constants $1 > c > s > 0$ such that $\text{Gap}_{c,s}$Max-2SAT is **NP**-hard.

   [Hint: consider the gadget

   $(x_1 \vee x_2 \vee x_3) \mapsto (x_1) \wedge (x_2) \wedge (x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee \neg x_1) \wedge (y) \wedge (x_1 \vee \neg y) \wedge (x_2 \vee \neg y) \wedge (x_3 \vee \neg y).$]

3. Prove that for every $s < c/4$, $\mathbf{PCP}_{c,s}(\log n, 2) = \mathbf{P}$.

4. Prove that for some $s < c$, $\mathbf{PCP}_{c,s}(\log n, 2) = \mathbf{NP}$.

5. Prove that for every $s < 1$, $\mathbf{PCP}_{1,s}(\log n, 2) = \mathbf{P}$.

Problem 3. (Multiprover Interactive Proofs) In a p-prover interactive proof system for a language L, p computationally unbounded provers $P_1, \ldots, P_p$ try to convince a probabilistic polynomial-time verifier V that some string x (given as common input to all parties) is in L. Prover P_i does not get to see the communication between V and the other $p-1$ provers, and the various provers are not able to communicate during the interaction. (Thus it is like V is a police officer interrogating several co-conspirators $P_1, \ldots, P_p$ in separate rooms, to help detect inconsistencies in their stories.) We require that on common input $x \in L$, V accepts with probability at least $2/3$ when interacting with $P_1, \ldots, P_p$, and when $x \notin L$, V accepts with probability at most $1/3$ no matter what strategies $P_1^*, \ldots, P_p^*$ the provers follow. p-**MIP** denotes the class of languages with p-prover proof systems, and $\mathbf{MIP} = \bigcup_p p\text{-}\mathbf{MIP}$.

Prove that $\mathbf{MIP} \subseteq \mathbf{PCP}[\text{poly}, \text{poly}] \subseteq 2\text{-}\mathbf{MIP}$, and hence all three classes are equal. (Hint: for the second inclusion, ask first prover all the PCP queries and use the second just to check that the first is 'behaving' like a PCP. You may assume that error reduction can be done without increasing the number of provers, but this is actually a very nontrivial fact — Raz's Parallel Repetition Theorem — and the error does not decrease as quickly as you'd expect.)

Problem 4. (Testing Graph Properties) A graph $G = (V, E)$ is a *biclique* if there is a partition (V_1, V_2) of V such that $E = V_1 \times V_2$. Consider the following promise problem:

$$\begin{aligned} (\textsc{Test}_\varepsilon\textsc{BiClique})_Y &= \{G : G \text{ is a biclique}\}. \\ (\textsc{Test}_\varepsilon\textsc{BiClique})_N &= \{G : G \text{ is } \varepsilon\text{-far from every biclique}\}, \end{aligned}$$

where we represent the graph G by its adjacency matrix, consisting of $N = n^2$ bits, and we say that two graphs are *ε-far* if their adjacency matrices differ on at least $\varepsilon \cdot N$ entries. In this problem, we will explore the possibility of sublinear-time algorithms for this promise problem, and thus assume a model of computation where the algorithm has random access (equivalently, oracle access) to its input.

1. Show that for every constant $\varepsilon > 0$, $\textsc{Test}_\varepsilon\textsc{BiClique}$ is in $\mathbf{prBPTIME}(O(\log N))$. (Hint: fix an arbitrary vertex $v_0 \in V$, randomly pick $u, w \stackrel{R}{\leftarrow} V$, and check which of the edges (v_0, u), (v_0, w), and (u, w) are in G.)

2. Show that for some constant $\varepsilon > 0$, $\textsc{Test}_\varepsilon\textsc{BiClique}$ is not in $\mathbf{prDTIME}(o(N))$.

Thus, even though there is evidence that $\mathbf{BPP} = \mathbf{P}$ (and also $\mathbf{prBPP} = \mathbf{prP}$), when we consider *sublinear time*, randomization provably provides an exponential savings over deterministic algorithms.