Take Home Exam 2

Mahya Jamshidian
Computatinal Complexity

May 25, 2019

Problem 1. Consider Δ as maximum degree of vertices in G. for each vertex i we add Δ - deg(i) self loops. call this t_i, We call the modified graph G' now consider an step of RW in G' which goes through one of those self loops of vertex i. I can be simulated in graph G by taking m_{ii} steps in G.
by using now that graph is $\Delta - Regular$:
we simulate every traverse in G' by n^3 steps and at most exists n^2 edges. so it takes n^5 steps. now instead of this we can expand the graph such that it'll be 4-regular and has n^2 vertices. By replacing each vertex with a cycle of length n and then for every pair (i,j) in G we pair ith node of cycle j with jth node of cycle i.
Transforming G to G' can be done in log-space by accesing to adjancy matrix of G.
Now that Graph has at most degree of 4 we can make it 4-regular by adding self-loops and then :
l * n * d = 4 * n^2 * n^2

Problem 2. a

$*_1$
A $\leq$ B:
$OPT_B(f_1(x_1)) \leq \alpha_1 OPT_A(x_1)$
B $\leq$ C:
$OPT_C(f_2(x_2)) \leq \alpha_2 OPT_B(x_2)$
$x_2 = f_1(x_1) \implies$
$OPT_C(f_2(f_1(x_1))) \leq \alpha_1\alpha_2 OPT_B(f_1(x_1))$
$\implies$
$\alpha_3 = \alpha_1\alpha_2$
$OPT_C(f_3(x_1)) \leq \alpha_3 OPT_B(x_1)$
$*_2$
A $\leq$ B:
$|c_A(g_1(y_1)) - OPT_A(x_1)| \leq \beta_1 |c_B(y_1) - OPT_B(f_1(x_1))|$
B $\leq$ C:
$|c_B(g_2(y_2)) - OPT_B(x_2)| \leq \beta_2 |c_C(y_2) - OPT_C(f_2(x_2))|$
$\implies$
$|c_A(g_2(g_1(y_1))) - OPT_A(x_1)| \leq \beta_1\beta_2 |c_C(g_1(y_1)) - OPT_C(f_2(x_2))|$

$\Longrightarrow$
$\beta_3 = \beta_1\beta_2$
$|c_A(g_3(y_3)) - OPT_C(x_3)| \leq \beta_3|c_C(y_3) - OPT_C(f_3(x_3))|$

form $*_1 and *_2 \Longrightarrow A \leq C$

Problem 2. b
Since B admits an $\rho - approximation$ algorithm and by defenition of reduction of minimization algorithm there exists some α that optimal soloution of B is bounded by α * optimal soloution of A. and on another note, for any soloution of A, we may bound our algorithm error by finding β and using second rule of such a reduction. Now that error of B is ρ and we bound the error by using β, we may choose $\rho^{`} = \beta * \rho$.

Problem 3. a
for any fixed input x any variable of $u \in \{0,1\}^{r(n)}$ and $\psi \in \{0,1\}^{q(n)}$; We can consider it as a boolean function $\{0,1\}^{|r(n)|+|q(n)|} \longrightarrow \{0,1\}$:
this has $2^{|r(n)|+|q(n)|}$ clauses as a CNF φ_x with $|r(n)| + |q(n)|$ per clause. while transforming each clause to 3 variables (which can be done in polynomial for every clause)* :
$total of 2^{|r(n)|+|q(n)|} * (|r(n)| + |q(n)|) \leq$ O$(2^{|r(n)|+|q(n)|} * (|r(n)| * |q(n)|))$;
if $x \in L$ then there exists a combination of u and b s.t. φ_x is satisfiable.
if $x \notin L$ then for every combination of u and ψ there exists at least one unsatsfied clause so it may not be chosen and verifies x.
* -¿ transforming O$(2^{r(n)}2^{q(n)}n^c)$

Problem 3. b
We assume $f_{u,b}$ as a function of verifier which outputs output of turing machine V (which is bounded by time t(n))
As q is constant and $f_{u,b}$ can be evaluated in polynomial time (simply run V), we can calculate in time O(t (n)) also the truth table of $f_{u,b}$ From this truth table (which is of constant size $\{0,1\}^{2|q|}$) we can compute a 3CNF formula $\psi_{u,b}$ which represents $f_{u,b}$ constant time w.r.t. the input.
Now we set
$\psi = \bigwedge u \in \{0,1\}^{r(n)}\ \psi_{b,u}$
Then ψ is satisfiable iff b is satisfiable. Computing ψ amounts to simulating V at most 2^q times for every $u \in \{0,1\}^{|r(||)|}$ which takes time O$(2^{r(||)} * t(||))$.
by previous part we know that $|\psi| \in$ O$(2^{|r(n)|} * r(n))$ which is under O$(2^{2|r(n)|})$
i.e., there is some constant c ¿ 1 s.t. $|\psi| \leq c2^{2|r(||)|}$ if —b— $\geq n_0$ for some n_0.
Now, by making n_0 even larger, we may assume that r(n) $\leq$ 1/4 log(n) for all n $\geq n_0$:
$|\psi| \leq c2^{1/2.log|b|} = c.|b|^{1/2}$

Problem 3. c
Assume now, we apply this construction at most log n-times (i.e., we immediately stop if we obtain a formula of length n_0) where n is the length of the original formula. Every reduction takes time polynomial in the original formula, so the total time is also bounded by some

polynomial. Consider the length of the resulting formula assuming that it still is of length at least n_0 :
$n \longrightarrow c.n^{1/2} \longrightarrow c.c^{1/2}.n^{1/4} \longrightarrow ... \longrightarrow c^{\Sigma_{i=0}^{-1+logn} 2^{-i}}.n^{2^{-logn}} \leq c^2.n^{1/n}$.

Problem 3. d
Now, as $n^{1/n} = e^{1/nlogn}$ goes to 1 for large n, we can choose n_0 even so large, that $c^2 n_0^{1/n_0}$ $\leq n_0$. Within polynomial time we therefore can reduce the original formula φ to a formula of length at most n_0 . Obviously, we can decide for every formula of length at most n_0 in constant time whether it is satisfiable or not. $\longrightarrow$ 3SAT is in P and P = NP.