CS 224 Advanced Algorithms — Fall 2014

Problem Set 5

Due: 11:59pm, Monday, October 27th
Submit to: cs224-f14-assignments@seas.harvard.edu
Solution maximum page limit: 5 pages

See homework policy at http://people.seas.harvard.edu/~minilek/cs224/hmwk.html

Problem 1: In this problem we will develop a PTAS for a scheduling problem. There are
n jobs that we would like to schedule on m machines. Job i requires processing time \( p_i \), and
each job must be assigned to exactly one machine. The completion time of a machine is then
the sum of processing times of jobs assigned to it, and the “load” of an assignment is the
maximum completion time of any machine in that assignment. We want to minimize load.

(a) (2 points) Let \( p_{\text{max}} \) be the maximum processing time of any job. Show that the quantity
\[ \max\{p_{\text{max}}, (1/m) \sum_{i=1}^{n} p_i\} \]
is a lower bound on \( \text{OPT} \).

(b) (3 points) Consider a greedy algorithm which loops over the jobs from 1 to n, and for
job i assigns it to the currently least loaded machine (the one whose completion time
when considering jobs assigned so far is smallest). Show that the greedy algorithm
achieves completion time at most \((2 - 1/m)\text{OPT}\). \textbf{Hint:} Use (a).

Suppose we want a PTAS, i.e. to achieve load at most \((1 + \varepsilon)\text{OPT}\) with an algorithm whose
running is time polynomial in the input size (the exponent of this polynomial can be any
function of \(1/\varepsilon\)). Define the “big” jobs to be \( B = \{i \in [n] : p_i \geq \varepsilon\text{OPT}\} \) and the “small”
jobs to be \( S = \{i \in [n] : p_i < \varepsilon\text{OPT}\} \).

(c) (2 points) Show that if the jobs in \( B \) are assigned to machines to achieve load \( L \), then
greedily assigning the remaining jobs in \( S \) achieves load at most \( \max\{L, (1 + \varepsilon)\text{OPT}\} \).

(d) (4 points) Show that if there are at most \( k \) distinct job processing times across all
jobs, then for any \( T \) there is a dynamic programming algorithm which finds a schedule
achieving load at most \( T \) in time \( O(n^{2k}) \) (or reports if no such schedule exists).

(e) (5 points) Conclude that, if we know \( \text{OPT} \), we can schedule all jobs with load at most
\((1 + \varepsilon)\text{OPT}\) in time \( n^{O(\log(1/\varepsilon)/\varepsilon)} \). \textbf{Hint:} For \( i \in B \), round \( p_i \) to \( \varepsilon(1 + \varepsilon)^j \) for integer \( j \).

(f) (4 points) In actuality we don’t know \( \text{OPT} \). Show, nonetheless, that there is a PTAS
for our scheduling problem.

Problem 2: (5 points) Show an integrality gap between the quadratic programming formu-
lation of MAXCUT in class and its vector programming relaxation. You just need to give
a single graph demonstrating a gap (not an infinite family of graphs). Full points are given
for showing any non-trivial gap. \textbf{Hint:} consider the input graph being a cycle.
Problem 3: Given an undirected graph $G = (V, E)$ with $V = [n]$, a $k$-coloring of $G$ is a function $c : [n] \to [k]$ such that for all $e = (u, v) \in E$, it holds that $c(u) \neq c(v)$. That is, we would like to “color” the elements of $[k]$ so as to avoid monochromatic edges.

As we know from undergraduate algorithms, a 2-coloring of a graph can be found in polynomial time (if it exists). Unfortunately, deciding whether a graph is 3-colorable is NP-hard. Imagine a different but related problem: we are given a graph $G$ which is promised to be 3-colorable, and we are asked to color it with as few colors as possible. It is known that given this promise, 4-coloring the graph is still NP-hard. Coloring a 3-colorable (or any) graph with $n$ colors is of course trivial. In this problem we will develop methods to color 3-colorable graphs with $o(n)$ colors. In what follows, $G$ is a 3-colorable graph.

(a) (1 points) Let $\Delta$ be the max degree of $G$. Show $G$ can be efficiently $(\Delta + 1)$-colored.

(b) (2 points) Consider the following algorithm. Initialize $C$ to 1. While there exists a vertex $v$ with degree at least $\delta$, color $v$ with color $C$, and 2-color $v$'s neighbors using colors $C + 1$ and $C + 2$ (note this is possible since $G$ is 3-colorable). Now remove $v$ and its neighborhood from the graph, then increment $C$ by 3. If no such $v$ exists, then $\delta$-color the remaining vertices using colors $\{C, \ldots, C + \delta - 1\}$ via (a).

Question: How should $\delta$ be chosen to minimize the number colors used in this algorithm, and what is that number of colors? Answering in big-Oh notation is fine.

(c) (3 points) Consider the following vector programming relaxation of the coloring problem. Rather than color vertices with elements of $[k]$, we will “color” them with vectors.

$$\min \sum_{i,j=1}^{n} 0 \cdot \langle v_i, v_j \rangle$$

s.t. $\langle v_i, v_j \rangle \leq -\frac{1}{2}$ $\forall (i, j) \in E$

$$\langle v_i, v_i \rangle = 1$ $\forall i \in [n]$

Note the objective function to be minimized is trivial, thus the goal is just to come up with any feasible solution. Show that if $G$ is 3-colorable, then a feasible solution exists. Hint: Represent the three different colors as three different vectors; you will only need these vectors to live in $\mathbb{R}^2$.

(d) (8 points) Given a feasible vector solution in (c), show that it is possible to $O(\Delta \log_2^2)$-color $G$ using actual integer colors so that in expectation at most $n/6$ edges are monochromatic. Hint: Generate $t$ independent random vectors on the sphere $r_1, \ldots, r_t$ and color vertices using $t$-bit integers, where the $j$th bit of vertex $i$’s color is dictated by $\text{sign}(\langle v_i, r_j \rangle)$.

(e) (2 points) Conclude that it is possible to efficiently $O(\Delta \log_2^2 \log n)$-color $G$ with high probability.

(f) (3 points) Combine (e) with the approach of (b) to efficiently $O(n^\gamma)$-color $G$ for some $\gamma < 1/2$. 

2
Problem 4: (1 point) How much time did you spend on this problem set? If you can remember the breakdown, please report this per problem. (sum of time spent solving problem and typing up your solution)