CMSC 651, Analysis of Algorithms, University of Maryland, Fall 2013
Homework (somewhat) related to Arora-Hazan-Kale, due at the beginning of class on December 12, 2013

Instructions: (i) Submit a written assignment, or email to Khoa by the deadline. (ii) Submit one writeup per group; please discuss within your group – other resources including the Web are not allowed for consultation. Write your solutions neatly.

Consider the following type of maximum-flow problem, where we do not have the usual assumption that there is only one source and only one sink. As usual, we are given a graph \( G = (V, E) \) (undirected, say), and a positive capacity \( c_e \) for every edge \( e \). In contrast with the type of flow-problem we studied in class, we are also given a set \( \mathcal{P} \) of paths in \( G \), and want to assign some non-negative flow value \( f_p \) to every path \( p \in \mathcal{P} \) in order to maximize the total amount of flow assigned, with the (familiar) constraint that for every edge \( e \in E \), the total flow, over all paths \( p \in \mathcal{P} \) that use \( e \), is at most \( c_e \).

(a) Formulate this problem as a linear program.

(b) We will now show that there is some universal constant \( K_0 > 0 \) such that for any given parameter \( \epsilon > 0 \), we can approximate this problem to within \( (1 - K_0 \epsilon) \) by means of a fast algorithm, without resorting to LP as a black-box. We will assume that \( \epsilon \) is small enough: say, \( \epsilon \leq 0.1 \).

We will assume throughout that \( \min_{e \in E} c_e \geq 1 \); convince yourself that this is without loss of generality. Given a current flow-assignment \( f \), we will always let \( f(e) \) denote the total current flow on \( e \), i.e., \( \sum_{p \in \mathcal{P}} f_p \). (Note that \( \sum_{e \in E} f(e) = \sum_{p \in \mathcal{P}} (L(p) \cdot f_p) \), where \( L(p) \) denotes the number of edges in path \( p \in \mathcal{P} \).) Also, to avoid confusing edges \( e \) with \( e = 2.71 \cdots \), we will let \( \exp(x) \) denote \( e^x \). Our four-step iterative algorithm is as follows:

1. Initialize \( f_p := 0 \) for all \( p \in \mathcal{P} \).
2. \textbf{Repeat} until there exists some \( e \in E \) with \( f(e) \geq (c_e/\epsilon) \ln m \):
   2a. Let \( p \in \mathcal{P} \) be any path in \( \mathcal{P} \) that minimizes \( \sum_{e \in P} (1/c_e) \cdot \exp(f(e)/c_e) \); set \( f_p := f_p + \epsilon \).
3. Let \( \lambda := \max_{e \in E} f(e)/c_e \). \textbf{(Comment:} \( \lambda \) is the maximum ratio by which this flow \( f \) exceeds some edge-capacity; we will scale \( f \) down by \( \lambda \) in the next step in order to not exceed capacities.\textbf{)}
4. Let \( f_p := f_p/\lambda \) for all \( p \in \mathcal{P} \); return \( f \).

You will now prove that our algorithm returns a flow of value at least \( (1 - K_0 \epsilon) \) times maximum, in a number of iterations that is not too large. For this, we will define the following key “potential function” \( \Phi(f) \) that depends on the current flow \( f \) (i.e., the current assignment of flow-values to the paths in \( \mathcal{P} \)):

\[
\Phi(f) = \ln \left( \sum_{e \in E} \alpha_e(f) \right).
\]

Observe that \( \max_e f(e)/c_e \) is always bounded by \( \Phi(f) \). Also note that as we keep adding to the flow, \( \Phi \) keeps increasing.

(i) How can finding \( p \) in step 2(a) be done efficiently? Just give a short high-level description.

(ii) Prove that there is some constant \( K_1 > 0 \) such that in any iteration where we add \( \epsilon \) to the flow on a path \( p \) to update \( f \) to a new flow \( f' \), the increase in \( \Phi \) is at most

\[
(1 + K_1 \epsilon) \cdot \frac{\sum_{e \in P} \alpha_e(f)/c_e}{\sum_{e \in E} \alpha_e(f)}.
\]

(Use the assumption “\( \min_{e \in E} c_e \geq 1 \)”, some basic algebra, as well as some simple bounds relating to the logarithmic and exponential functions (e.g., similar to some from Arora-Hazan-Kale) to simplify your calculations.)
(iii) Let $f^*$ be some optimal flow for our problem; as usual, we of course do not know $f^*$, but we can use it in our analysis. Let $val(f^*)$ denote the optimal solution value: i.e., the value of

$$\sum_{p \in P} f_p^*.$$ 

You will now prove the following key bound: that in every iteration of the algorithm, $\Phi$ increases by at most $(1 + K_1 \epsilon) \cdot \epsilon / val(f^*)$. **Hint:** Let $q \in P$ be a random path, such that for any path $r \in P$, $Pr[q = r] = f_r^*/val(f^*)$. What is a good upper-bound on the expected value of $\sum_{e \in q} \alpha_e(f)/c_e$? (In finding this upper-bound, use the relationship between $f_e^*$ and $c_e$.) Now note that a close relative of this term appears in (iv), and recall how $p$ was chosen in step 2(a) of the algorithm.

(iv) Show (e.g., by induction) that our algorithm maintains the invariant

$$val(f^*) \cdot \frac{\Phi(f) - \ln m}{1 + K_1 \epsilon} \leq \sum_{r \in P} f_r.$$ 

(v) Use (iv) to prove that when the algorithm terminates, we have $\sum_{p \in P} f_p \geq (1 - K_0 \epsilon) \cdot val(f^*)$.

(vi) Prove that the number of iterations is at most $O(val(f^*) \cdot (\ln m)/\epsilon^2)$. 

2