CS261: Problem Set #3

Due by 11:59 PM on Tuesday, May 19, 2015

Instructions:

(1) Form a group of 1-3 students. You should turn in only one write-up for your entire group.

(2) Turn in your solutions by email to `cs261submissions@gmail.com`. Please type your solutions if possible and feel free to use the LaTeX template provided on the course home page.

(3) Write convincingly but not excessively.

(4) Some of these problems are difficult, so your group may not solve them all to completion. In this case, you can write up what you've got (subject to (3), above): partial proofs, lemmas, high-level ideas, counterexamples, and so on.

(5) Except where otherwise noted, you may refer to your course notes and the specific supplementary readings listed on the course Web page *only*. You can also review any relevant materials from your undergraduate algorithms course.

(6) You can discuss the problems verbally at a high level with other groups. And of course, you are encouraged to contact the course staff (via Piazza or office hours) for additional help.

(7) If you discuss solution approaches with anyone outside of your group, you must list their names on the front page of your write-up.

(8) Refer to the course Web page for the late day policy.

(9) Refer to Piazza for the course regrade policy.

Problem 13

[ESTIMATED DIFFICULTY LEVEL: Somewhat difficult.] This problem fills in some gaps in our proof sketch of strong linear programming duality.

(a) For this part, assume the version of Farkas's Lemma stated in Lecture #10, that given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, exactly one of the following statements holds: (i) there is an $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \geq 0$; (ii) there is a $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y}^T\mathbf{A} \geq 0$ and $\mathbf{y}^T\mathbf{b} < 0$.

Deduce from this a second version of Farkas's Lemma, stating that for $\mathbf{A}$ and $\mathbf{b}$ as above, exactly one of the following statements holds: (iii) there is an $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} \leq \mathbf{b}$; (iv) there is a $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \geq 0$, $\mathbf{y}^T\mathbf{A} = 0$, and $\mathbf{y}^T\mathbf{b} < 0$.

[Hint: note the similarity between (i) and (iv). Also note that if (iv) has a solution, then it has a solution with $\mathbf{y}^T\mathbf{b} = -1$.]

(b) Use the second version of Farkas's Lemma to prove the following version of strong LP duality: if the linear programs

$$\max \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{Ax} \leq \mathbf{b}$$

with $\mathbf{x}$ unrestricted, and

$$\min \mathbf{b}^T\mathbf{y}$$

subject to

$$\mathbf{A}^T\mathbf{y} = \mathbf{c}, \mathbf{y} \geq 0$$

are both feasible, then they have equal optimal objective function values.

[Hint: weak duality is easy to prove directly. For strong duality, let γ^* denote the optimal objective function value of the dual linear program. Add the constraint $\mathbf{c}^T\mathbf{x} \geq \gamma^*$ to the primal linear program and use Farkas's Lemma to show that the feasible region is non-empty.]

Problem 14

[ESTIMATED DIFFICULTY LEVEL: Not too bad.] Recall the *multicommodity flow* problem from Exercise 21. Recall the input consists of a directed graph $G = (V, E)$, k "commodities" or source-sink pairs $(s_1, t_1), \ldots, (s_k, t_k)$, and a positive capacity u_e for each edge.

Consider also the *multicut* problem, where the input is the same as in the multicommodity flow problem, and feasible solutions are subsets $F \subseteq E$ of edges such that, for every commodity (s_i, t_i), there is no s_i-t_i path in $G = (V, E \setminus F)$. (Assume that s_i and t_i are distinct for each i.) The *value* of a multicut F is just the total capacity $\sum_{e \in F} u_e$.

(a) Formulate the multicommodity flow problem as a linear program with one decision variable for each path P that travels from a source s_i to the corresponding sink t_i. Aside from nonnegativity constraints, there should be only be m constraints (one per edge).

[Note: this is a different linear programming formulation than the one asked for in Exercise 21.]

(b) Take the dual of the linear program in (a). Prove that every optimal 0-1 solution of this dual — i.e., among all feasible solutions that assign each decision variable the value 0 or 1, one of minimum objective function value — is the characteristic vector of a minimum-value multicut.

(c) Show by example that the optimal solution to this dual linear program can have objective function value strictly smaller than that of every 0-1 feasible solution. In light of your example, explain a sense in which there is no max-flow/min-cut theorem for multicommodity flows and multicuts.

Problem 15

[ESTIMATED DIFFICULTY LEVEL: A bit difficult.] Recall the Vertex Cover problem from lecture (and Problem #7): the input is an undirected graph $G = (V, E)$, and the goal is to compute a minimum-cardinality subset $S \subseteq V$ such that, for every edge $e \in E$, at least one of e's endpoints is in S. Recall the following linear programming relaxation of the problem:

$$\min \sum_{v \in V} x_v$$

subject to

$$x_v + x_w \geq 1 \qquad \text{for every edge } e = (v, w) \in E$$

$$x_v \geq 0 \qquad \text{for every vertex } v \in V.$$

Fix a graph G, let $\mathbf{x}^*$ denote an optimal solution to the linear programming relaxation, and define

$$A = \{v \in V : x_v^* > \tfrac{1}{2}\};$$

$$B = \{v \in V : x_v^* = \tfrac{1}{2}\};$$

and

$$C = \{v \in V : x_v^* < \tfrac{1}{2}\}.$$

(a) Prove that there exists an optimal vertex cover of G that includes every vertex of A and excludes every vertex of C.

[Hints: If S^* is an optimal vertex cover, argue that $A \setminus S^*$ and $S^* \cap C$ must have the same size — otherwise obtain a contradiction with either the optimality of S^* or the optimality $\mathbf{x}^*$. Use this to obtain another optimal vertex cover that has the desired form.]

(b) Prove that if G contains a vertex cover of size at most k, then $|B| \leq 2k$.

(c) Building on (a) and (b), give an algorithm that checks whether or not a graph G contains a vertex cover of size at most k and runs in time $\text{poly}(n) + 2^{O(k)}$. (Here $\text{poly}(n)$ denotes some polynomial function of n. You can assume that linear programs can be solved in polynomial time.)

Problem 16

[ESTIMATED DIFFICULTY LEVEL: A bit difficult.] Recall the makespan minimization problem from Lecture #11: the input is an integer $m \geq 1$ (the number of machines) and n jobs with nonnegative processing times $p_1, p_2, \ldots, p_n$; feasible solutions are schedules, meaning an assignment of each job to one of the machines; the load of a machine in a schedule is the sum of the processing times of the jobs assigned to that machine; and the goal is to minimize the makespan — the maximum machine load. In Lecture #11 we saw that greedily scheduling the jobs in arbitrary order (i.e., assign each job j to the currently least loaded machine) always yields a schedule with makespan at most twice the minimum possible.

In this problem, we consider the natural optimization where we first sort the jobs from biggest to smallest. That is, we first re-index the jobs so that $p_1 \geq p_2 \geq \cdots \geq p_n$, and then schedule them greedily as before, in this order. Your task is to prove the following two statements, which together imply that this algorithm is a $\frac{4}{3}$-approximation algorithm for the makespan minimization problem.

(a) Suppose p_k is the last job scheduled on a machine that has the biggest load at the conclusion of the algorithm. Prove that in an instance in which p_k is at most $\frac{1}{3}$ times the minimum-possible makespan OPT, the algorithm outputs a schedule with makespan at most $\frac{4}{3}OPT$.

[Hint: adapt the proof of the 2-approximation from lecture.]

(b) With p_k defined as in (a), prove that in an instance in which p_k is strictly greater than $\frac{1}{3}$ times the minimum-possible makespan OPT, the algorithm outputs a schedule with makespan OPT.

[Hint: one approach is to proceed by contraction. Suppose the algorithm schedules a job that causes a machine load to exceed OPT. Which pairs of jobs could possibly be scheduled on the same machine in a schedule with makespan OPT?]

Problem 17

[ESTIMATED DIFFICULTY LEVEL: Somewhat difficult.] In the *minimum multiway cut* problem, the input is an undirected graph $G = (V, E)$ with a positive capacity u_e for each edge e, and k distinct "terminals" $s_1, s_2, \ldots, s_k \in V$. A multiway cut is a partition of V into $(S_1, \ldots, S_k)$, where $s_i \in S_i$ for each $i = 1, 2, \ldots, k$. The value of a multiway cut is the sum of the capacities of the cut edges (edges with endpoints in different S_i's), and the goal is to compute a multiway cut with minimum-possible value. (The minimum s-t cut problem is the special case of $k = 2$.)

(a) Prove that the problem of computing the minimum-value partition $(A_i, \bar{A}_i)$, where $s_i \in A_i$ and $s_j \in \bar{A}_i$ for all $j \neq i$, can be solved in polynomial time.

(b) Suppose we invoke the subroutine (a) once for each s_i, and let E_i denote the edges cut in $(A_i, \bar{A}_i)$. Removing the edges $\cup_{i=1}^{k} E_i$ from G leaves $\ell \geq k$ connected components $B_1, \ldots, B_\ell$, with each s_i in a separate connected component B_i (why?). Define a multiway cut by setting $S_i = B_i$ for $i = 1, 2, \ldots, k-1$, and S_k to be the rest of the vertices. Prove that the value of this multiway cut is at most twice the minimum possible.

[Hint: Use an optimal multiway cut to define feasible cuts $(A_1, \bar{A}_1), \ldots, (A_k, \bar{A}_k)$ for the problems solved by the subroutine in (a) that have combined value at most twice the value of the multiway cut.]

(c) Can you tweak the algorithm and analysis in (b) to obtain an approximation ratio of $2 - \frac{2}{k}$?

Problem 18

[ESTIMATED DIFFICULTY LEVEL: A bit difficult.] This problem considers the "$\{1, 2\}$" special case of the asymmetric traveling salesman problem (ATSP). The input is a complete directed graph $G = (V, E)$, with all $n(n-1)$ directed edges present, where each edge e has a cost c_e that is either 1 or 2. Note that the triangle inequality holds in every such graph.

(a) Explain why the $\{1, 2\}$ special case of ATSP is NP-hard.

(b) Explain why it's trivial to obtain a polynomial-time 2-approximation algorithm for the $\{1, 2\}$ special case of ATSP.

(c) This part considers a useful relaxation of the ATSP problem. A *cycle cover* of a directed graph $G = (V, E)$ is a collection $C_1, \ldots, C_k$ of simple directed cycles, each with at least two edges, such that every vertex of G belongs to exactly one of the cycles. (A traveling salesman tour is the special case where $k = 1$.) Prove that given a directed graph with edge costs, a cycle cover with minimum total cost can be computed in polynomial time.

[Hint: bipartite matching.]

(d) Using (c) as a subroutine, give a $\frac{3}{2}$-approximation algorithm for the $\{1, 2\}$ special case of the ATSP problem.