CS261: Problem Set #2*

Due by 11:59 PM on Tue, Feb 11, 2020

Instructions:

(1) We encourage you to pair up with another student in working on this homework. You should turn in only one write-up for your group.

(2) Submission instructions: We will use Gradescope for the homework submissions. Go to www.gradescope.com to either login or create a new account. Use the course code 9GZK2R to register for CS261. Only one group member needs to submit the assignment. When submitting, please remember to add all group member names in Gradescope.

(3) You must use LaTeX, LyX, Microsoft Word, or a similar editor to typeset your write-up.

(4) Write convincingly but not excessively.

(5) 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.

(6) Except where otherwise noted, you may refer to the materials listed on the course web page *only*. You can also review any relevant materials from your undergraduate algorithms course.

(7) 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.

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

(9) Refer to the course web page for the late day policy.

Problem 7

The goal of this problem is to revisit two problems you studied in CS161 — the minimum spanning tree and shortest path problems — and to prove the optimality of Kruskal's and Dijkstra's algorithms via the complementary slackness conditions of judiciously chosen linear programs.

(a) For convenience, we consider the maximum spanning tree problem (equivalent to the minimum spanning tree problem, after multiplying everything by -1). Consider a connected undirected graph $G = (V, E)$ in which each edge e has a weight w_e.

For a subset $F \subseteq E$, let $\kappa(F)$ denote the number of connected components in the subgraph (V, F). Prove that the spanning trees of G are in an objective function-preserving one-to-one correspondence with the 0-1 feasible solutions of the following linear program (with decision variables $\{x_e\}_{e \in E}$):

$$\max \sum_{e \in E} w_e x_e$$

*We thank Tim Roughgarden for permission to include problems from his Winter 2016 edition of CS 261.

subject to

$$\sum_{e \in F} x_e \leq |V| - \kappa(F) \qquad \text{for all } F \subseteq E$$
$$\sum_{e \in E} x_e = |V| - 1$$
$$x_e \geq 0 \qquad \text{for all } e \in E.$$

(While this linear program has a huge number of constraints, we are using it purely for the analysis of Kruskal's algorithm.)

(b) What is the dual of this linear program?

(c) What are the complementary slackness conditions?

(d) Recall that Kruskal's algorithm, adapted to the current maximization setting, works as follows: do a single pass over the edges from the highest weight to lowest weight (breaking ties arbitrarily), adding an edge to the solution-so-far if and only if it creates no cycle with previously chosen edges. Prove that the corresponding solution to the linear program in (a) is in fact an optimal solution to that linear program, by exhibiting a feasible solution to the dual program in (b) such that the complementary slackness conditions hold.[1]

[Hint: for the dual variables of the form y_F, it is enough to use only those that correspond to subsets $F \subseteq E$ that comprise the i edges with the largest weights (for some i).]

(e) Now consider the problem of computing a shortest path from s to t in a directed graph $G = (V, E)$ with a nonnegative cost c_e on each edge $e \in E$. Prove that every simple s-t path of G corresponds to a 0-1 feasible solution of the following linear program with the same objective function value:[2]

$$\min \sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta^+(S)} x_e \geq 1 \qquad \text{for all } S \subseteq V \text{ with } s \in S,\ t \notin S$$
$$x_e \geq 0 \qquad \text{for all } e \in E.$$

(Again, this huge linear program is for analysis only.)

(f) What is the dual of this linear program?

(g) What are the complementary slackness conditions?

(h) Let P denote the s-t path returned by Dijkstra's algorithm. Prove that the solution to the linear program in (e) corresponding to P is in fact an optimal solution to that linear program, by exhibiting a feasible solution to the dual program in (f) such that the complementary slackness conditions hold.

[Hint: it is enough to use only dual variables of the form y_S for subsets $S \subseteq V$ that comprise the first i vertices processed by Dijkstra's algorithm (for some i).]

[1] You can assume without proof that Kruskal's algorithm outputs a feasible solution (i.e., a spanning tree), and focus on proving its optimality.

[2] Recall that $\delta^+(S)$ denotes the edges sticking out of S.

Problem 8

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 #7, 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 9

Recall the *multicommodity flow* problem from Exercise 20. 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 20.]

(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 10

This problem gives a linear-time (!) randomized algorithm for solving linear programs that have a large number m of constraints and a small number n of decision variables. (The constant in the linear-time guarantee $O(m)$ will depend exponentially on n.)

Consider a linear program of the form

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

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

For simplicity, assume that the linear program is feasible with a bounded feasible region, and let M be large enough that $|x_j| < M$ for every coordinate of every feasible solution. Assume also that the linear program is "non-degenerate," in the sense that no feasible point satisfies more than n constraints with equality. For example, in the plane (two decision variables), this just means that there does not exist three different constraints (i.e., halfplanes) whose boundaries meet at a common point. Finally, assume that the linear program has a unique optimal solution.[3]

Let $C = \{1, 2, \ldots, m\}$ denote the set of constraints of the linear program. Let B denote additional constraints asserting that $-M \leq x_j \leq M$ for every j. The high-level idea of the algorithm is: (i) drop a random constraint and recursively compute the optimal solution $\mathbf{x}^*$ of the smaller linear program; (ii) if $\mathbf{x}^*$ is feasible for the original linear program, return it; (iii) else, if $\mathbf{x}^*$ violates the constraint $\mathbf{a}_i^T\mathbf{x} \leq b_i$, then change this inequality to an equality and recursively solve the resulting linear program.

More precisely, consider the following recursive algorithm with two arguments. The first argument C_1 is a subset of inequality constraints that must be satisfied (initially, equal to C). The second argument is a subset C_2 of constraints that must be satisfied with equality (initially, $\emptyset$). The responsibility of a recursive call is to return a point maximizing $\mathbf{c}^T\mathbf{x}$ over all points that satisfy all the constraints of $C_1 \cup B$ (as inequalities) and also those of C_2 (as equations).

Linear-Time Linear Programming

Input: two disjoint subsets $C_1, C_2 \subseteq C$ of constraints

Base case #1: if $|C_2| = n$, return the unique point that satisfies every constraint of C_2 with equality

Base case #2: if $|C_1| + |C_2| = n$, return the point that maximizes $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{a}_i^T\mathbf{x} \leq b_i$ for every $i \in C_1$, $\mathbf{a}_i^T\mathbf{x} = b_i$ for every $i \in C_2$, and the constraints in B

Recursive step:
choose $i \in C_1$ uniformly at random
recurse with the sets $C_1 \setminus \{i\}$ and C_2 to obtain a point $\mathbf{x}^*$
if $\mathbf{a}_i^T\mathbf{x}^* \leq b_i$ **then**
    return $\mathbf{x}^*$
else
    recurse with the sets $C_1 \setminus \{i\}$ and $C_2 \cup \{i\}$, and return the result

(a) Prove that this algorithm terminates with the optimal solution $\mathbf{x}^*$ of the original linear program.

[Hint: be sure to explain why, in the "else" case, it's OK to recurse with the ith constraint set to an equation.]

(b) Let $T(m, s)$ denote the expected number of recursive calls made by the algorithm to solve an instance with $|C_1| = m$ and $|C_2| = s$ (with the number n of variables fixed). Prove that T satisfies the following recurrence:

$$T(m,s) = \begin{cases} 1 & \text{if } s = n \text{ or } m + s = n \\ T(m-1,s) + \frac{n-s}{m} \cdot T(m-1,s+1) + 1 & \text{otherwise.} \end{cases}$$

[3] All of these simplifying assumptions can be removed without affecting the asymptotic running time; we leave the details to the interested reader.

[Hint: you should use the non-degeneracy assumption in this part.]

(c) Prove that $T(m, 0) = O(n! \cdot m)$.

[Hint: it might be easiest to make the variable substitution $\delta = n - s$ and proceed by simultaneous induction on m and δ.]

(d) Conclude that, for every fixed constant n, the algorithm above can be implemented so that the expected running time is $O(m)$ (where the hidden constant can depend arbitrarily on n).

Problem 11

Let $G = (V, E)$ be an arbitrary directed graph with weighted vertices; vertex weights may be positive, negative, or zero. A *prefix* of G is a subset $P \subseteq V$, such that there is no edge $u \to v$ where $u \notin P$ but $v \in P$. A *suffix* of G is the complement of a prefix. Finally, an *interval* of G is the intersection of a prefix of G and a suffix of G. The weight of a prefix, suffix, or interval is the sum of the weights of its vertices.

(a) Describe a linear program whose optimal solution describes the *maximum-weight prefix* of G. Your linear program should have one variable per vertex, indicating whether that vertex is or is not in the chosen prefix.

(b) Show that your linear program for (a) has an optimal solution that is integral.

[Hint: Here is a sketch of an argument:

Step 1: Starting with any feasible solution $\mathbf{x}$, for each $t \in [0, 1]$ define $S(t) = \{v : x_v \geq t\}$ and $W(t) = \sum_{v \in S(t)} w_v$, where x_v is the value of the variable corresponding to vertex v in $\mathbf{x}$, and w_v is the weight of v. Check that:

(i) For every $t \in [0, 1]$, $S(t)$ is a prefix;

(ii) The objective function value of the linear program on $\mathbf{x}$ is $\int_0^1 W(t)dt$.

Step 2: Conclude that $\mathbf{x}$ is a convex combination of integral solutions.

Step 3: Now what can you say when $\mathbf{x}$ is optimal?]

(c) Describe a linear program whose 0-1 feasible solutions are in one-to-one correspondence with *intervals* of G. Your linear program should have one variable per vertex, indicating whether that vertex is or is not in the chosen interval.

Problem 12

Consider the following optimization problem with *robust conditions*, where we do not know the constraint matrix A exactly, but know some bounds on the entries of A and want to optimize with respect to the worst-case A in that family. The objective is $\min c^T x$ for $x \in \mathbb{R}^n$, and the constraints are

$$Ax \geq b \text{ for any } A \in F$$

where $b \in \mathbb{R}^m$ and F is the following set of $m \times n$ matrices:

$$F = \{A : \forall i, j,\ a_{ij}^{\min} \leq a_{ij} \leq a_{ij}^{\max}\}$$

(the values $a_{ij}^{\min}, a_{ij}^{\max}$ are part of the input, as b and c are).

(a) Considering F as a polytope in $\mathbb{R}^{m \times n}$, what are the vertices of F?

(b) Show that instead of solving the optimization problem with constraints for every $A \in F$, it is enough to have constraints for the vertices of F. Write the resulting linear program. What is the size of the LP? Is it polynomial in the size of the input (m, n, and the space required to store a real number such as $a_{ij}^{\min}$)?

(c) Consider a fixed x and one row of A. Formulate an LP that checks whether the fixed vector x satisfies the condition on this row of A for all $A \in F$. What is the dual program to this LP?

(d) Use the dual program from part (c) to formulate a linear program of polynomial size that solves the original problem.

[Hint: for each row, add a constraint on the objective function value, as in Problem 8(b).]