Homework Assignment 4

Submission Guidelines.

- Latex submissions are preferred but not mandatory. However, please write your solutions on unruled paper and upload a clearly scanned copy. Please do not use poorly lit photographs. Keep your solutions brief and to the point. Meaningless rambles fetch negative credits.
- Collaboration is allowed in teams of up to two students, but each student must write and submit their own individual solutions in their own words. Also mention the name of your collaborator.
- "Bibles" such as textbooks, lecture notes from other courses, or online forums (e.g., Stack Exchange) may not be used as references for your write-up.
- Limited use of large language models (LLMs) is permitted, but you may not directly copy-paste text from them; use them only for clarification and then produce your own reasoning and explanations.

Problem 1. (20 points)

This problem fills in some gaps in our proof sketch of strong linear programming duality.

(a) (10 points) Assume the version of Farkas's Lemma stated in Lecture #9: given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, exactly one of the following statements holds:

(i) there exists an $x \in \mathbb{R}^n$ such that $Ax = b$ and $x \geq 0$;

(ii) there exists a $y \in \mathbb{R}^m$ such that $y^T A \geq 0$ and $y^T b < 0$.

Deduce from this a second version of Farkas's Lemma, stating that for A and b as above, exactly one of the following statements holds:

(iii) there exists an $x \in \mathbb{R}^n$ such that $Ax \leq b$;

(iv) there exists a $y \in \mathbb{R}^m$ such that $y \geq 0$, $y^T A = 0$, and $y^T b < 0$.

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

Solution. We start from the following version of Farkas's Lemma:

Exactly one of the following holds:

(a) There exist variables $x_1, \ldots, x_n \geq 0$ such that, for every $i = 1, \ldots, m$,

$$\sum_{j=1}^{n} A_{ij} x_j = b_i.$$

(b) There exist variables $y_1, \ldots, y_m$ such that, for every $j = 1, \ldots, n$,

$$\sum_{i=1}^{m} y_i A_{ij} \geq 0 \quad \text{and} \quad \sum_{i=1}^{m} y_i b_i < 0.$$

Now consider the system of inequalities

$$\sum_{j=1}^{n} A_{ij} x_j \leq b_i \qquad \text{for all } i = 1, \ldots, m.$$

Introduce slack variables $s_1, \ldots, s_m \geq 0$ and rewrite the system as

$$\sum_{j=1}^{n} A_{ij} x_j + s_i = b_i \qquad \text{for all } i = 1, \ldots, m.$$

Apply the above version of Farkas's Lemma to this system of equations, viewing $(x_1, \ldots, x_n, s_1, \ldots, s_m)$ as the variables, where only the s_i's are required to be nonnegative.

Then exactly one of the following holds:

(a) There exist $x_1, \ldots, x_n$ such that all equations are satisfied with $s_i \geq 0$ for all i, which is equivalent to

$$\sum_{j=1}^{n} A_{ij} x_j \leq b_i \quad \text{for all } i,$$

i.e., statement (iii).

(b) There exist $y_1, \ldots, y_m$ such that

$$\sum_{i=1}^{m} y_i A_{ij} = 0 \quad \text{for all } j = 1, \ldots, n, \qquad y_i \geq 0 \quad \text{for all } i,$$

and

$$\sum_{i=1}^{m} y_i b_i < 0,$$

which is exactly statement (iv).

Thus, exactly one of (iii) or (iv) holds. □

Rubric : Correct reformulation of $Ax \leq b$ using slack variables (+3), Correct application of the given version of Farkas's Lemma (+4), Clear identification of statements (iii) and (iv) (+3)

(b) (10 points) Use the second version of Farkas's Lemma to prove the following version of strong LP duality: if the linear programs

$$\max c^T x \qquad \text{subject to } Ax \leq b$$

with x unrestricted, and

$$\min b^T y \qquad \text{subject to } A^T y = c,\ y \geq 0$$

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

You are not allowed to use any LLM for this problem.

Solution. Consider the primal–dual pair

$$\max\{c^T x : Ax \le b\}, \qquad \min\{b^T y : A^T y = c,\ y \ge 0\}.$$

Weak duality. For any primal-feasible x and dual-feasible y,

$$c^T x = y^T A x \le y^T b = b^T y.$$

Hence the primal optimum is at most the dual optimum.

Let γ^* denote the optimal value of the dual. Add the constraint $c^T x \ge \gamma^*$ to the primal and consider the system

$$Ax \le b, \qquad c^T x \ge \gamma^*.$$

Suppose this system is infeasible. Rewrite it as

$$Ax \le b, \qquad -c^T x \le -\gamma^*.$$

By the second version of Farkas's Lemma, there exist vectors $y \ge 0$ and a scalar $\lambda \ge 0$, not both zero, such that

$$y^T A - \lambda c^T = 0 \quad \text{and} \quad y^T b - \lambda \gamma^* < 0.$$

If $\lambda = 0$, then $y^T A = 0$ and $y^T b < 0$, contradicting feasibility of the primal. Hence $\lambda > 0$. Dividing by λ, we obtain

$$A^T \left(\frac{y}{\lambda}\right) = c, \qquad b^T \left(\frac{y}{\lambda}\right) < \gamma^*,$$

which contradicts the optimality of γ^* for the dual.

Therefore the augmented primal system is feasible, so there exists a feasible x with $c^T x \ge \gamma^*$. By weak duality, $c^T x \le \gamma^*$, and hence

$$c^T x = \gamma^*.$$

Thus, the primal and dual linear programs have equal optimal objective values. □

Rubric: Statement and correct use of weak duality (+2), Correct setup using the dual optimal value $\gamma^\star$ (+3), Correct application of the second version of Farkas's Lemma (+3), Deriving the contradiction and concluding strong duality (+2)

[Hint: first recap weak duality without proof. For strong duality, let γ^* denote the optimal objective function value of the dual linear program. Add the constraint $c^T x \ge \gamma^*$ to the primal linear program and use Farkas's Lemma to show that the feasible region is nonempty. In other words, if the feasible region after adding the constraint is empty, then invoke the second condition (iv) from above to show the γ^* cannot be the optimal solution to the dual.]

Problem 2. (20 points) Recall the multicommodity flow problem from Exercise 17. 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 e.

Consider also the multicut problem, where the input is the same but 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 $G = (V, E \setminus F)$. The value of a multicut F is

$$\sum_{e \in F} u_e.$$

(a) (5 points) 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 only be m constraints (one per edge).

Solution. LP for multicommodity flow.

For each commodity $i = 1, \ldots, k$, let $\mathcal{P}_i$ denote the set of all directed s_i–t_i paths. For each path $P \in \mathcal{P}_i$, introduce a variable $f_P \geq 0$ representing the amount of flow sent on path P.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} f_P \\ \text{subject to} \quad & \sum_{i=1}^{k} \sum_{\substack{P \in \mathcal{P}_i \\ e \in P}} f_P \;\leq\; u_e \qquad \text{for every edge } e \in E, \\ & f_P \geq 0 \qquad \text{for all paths } P. \end{aligned}$$

Rubric: +2 for correctly defining variables, +1 for correct objective , +2 for each of the constraints.

(b) (5 points) Take the dual of the linear program in (a). Prove that every optimal 0–1 solution of this dual—that is, every feasible solution assigning each decision variable the value 0 or 1 and of minimum objective function value—is the characteristic vector of a minimum-value multicut.

Solution. Introduce a dual variable $y_e \geq 0$ for each edge e.

The dual linear program is

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} u_e y_e \\ \text{subject to} \quad & \sum_{e \in P} y_e \geq 1 \qquad \text{for every } s_i\text{–}t_i \text{ path } P,\ i = 1, \ldots, k, \\ & y_e \geq 0 \qquad \text{for all } e. \end{aligned}$$

Now suppose $y_e \in \{0, 1\}$ for all e. Let

$$F = \{e \in E : y_e = 1\}.$$

For every commodity i, every s_i–t_i path contains at least one edge of F; hence removing F disconnects all (s_i, t_i) pairs. Thus F is a multicut.

The objective value is

$$\sum_{e \in F} u_e,$$

which is exactly the value of the multicut. Therefore, every optimal 0–1 solution of the dual corresponds to a minimum-value multicut. □

Rubric: +1 for correct objective and +1 for each of the constraints, +2 for proving the connection to multi-cut.

(c) (10 points) 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 analogous to the usual max-flow/min-cut theorem.

Solution. Consider a graph consisting of a single edge e of capacity 1, and two commodities whose source–sink pairs both require this edge.

- Any multicut must remove e, so every 0–1 feasible solution has value 1.
- In the dual LP, setting $y_e = \frac{1}{2}$ satisfies all constraints (each path contains e) and has objective value $\frac{1}{2}$.

Thus, the optimal dual LP value is strictly smaller than that of any integral solution.

In the single-commodity case, the max-flow/min-cut theorem states that the value of the maximum flow equals the minimum capacity of an $s-t$ cut. Equivalently, the flow LP and the cut LP have the same optimal value, and both admit optimal integral solutions.

In contrast, in the multicommodity setting, the example above shows that:

$$\underbrace{\max\{\text{multicommodity flow}\}}_{\text{primal LP}} = \underbrace{\min\{\text{fractional multicut}\}}_{\text{dual LP}} < \underbrace{\min\{\text{integral multicut}\}}_{\text{true multicut}}.$$

Hence, there cannot be an analogous max-flow/min-cut theorem for the integer multicommodity flow problem.

Rubric: +5 for the correct example, +5 for explaining the conclusion precisely (+3 for showing the chain of inequalities or something analogous, +2 for correct conclusion)

Problem 3. For any $\alpha \geq 1$, define an α-approximate cut in a graph G as any cut whose cardinality is within a multiplicative factor α of the cardinality of a global min-cut in G.

(a) (5 point) Determine the probability that a single iteration of the randomized algorithm for min-cuts will produce as output some α-approximate cut in G.

Solution. At any stage of Karger's randomized contraction algorithm, the number of edges is at least $\frac{n\lambda}{2}$, since the average degree is at least λ.

Any α-approximate cut has size at most $\alpha\lambda$. Hence, in a single contraction step, the probability of contracting an edge of a fixed α-approximate cut is at most

$$\frac{\alpha\lambda}{n\lambda/2} = \frac{2\alpha}{n}.$$

Therefore, the probability that this cut survives all contractions down to two vertices is at least

$$\prod_{i=n}^{3}\left(1-\frac{2\alpha}{i}\right) = e^{\sum_{i=n}^{3}\ln\left(1-\frac{2\alpha}{i}\right)} \geq e^{-\sum_{i=n}^{3}\frac{2\alpha}{i}} \geq e^{-2\alpha\ln n} = n^{-2\alpha}$$

Major Issue. In the above calculation, especially, while applying the second inequality, we assumed that $\alpha < 1.5$. In fact, for $i \leq 2\alpha$, the probability of survival *as per this calculation* will start becoming zero and not give anything meaningful. In fact, it is very tricky to determine a good lower bound for survival probability for approximate min-cuts. So here is the

Rubric: +2 for any reasonable bound and +3 for correct justification of that bound. Some of you have assumed that the contraction goes down till we have 2α vertices remaining and determined the probability according to that (see part (c)). I have given them full credit.

(b) (5 point) Using the analysis of the randomized min-cut algorithm, show that the number of distinct min-cuts in a graph G cannot exceed $\binom{n}{2}$, where n is the number of vertices in G.

Solution. Since a single run outputs exactly one cut, the total survival probability over all minimum cuts is at most 1 (by law of total probability). Hence,

$$(\text{number of minimum cuts}) \cdot n^{-2} \leq 1,$$

which implies that the number of distinct minimum cuts is at most n^2.

Rubric: +3 for realizing and stating the total probability thing . +2 for applying it correctly

(c) (5 point) Formulate and prove a similar upper bound on the number of α-approximate cuts in G using your analysis from part (a).

Solution. This one is really really tricky and does not follow at all in the same way as part (b). The main reason is the following - all of you have assumed that the survival probability is $n^{-2\alpha}$ and applied the same logic as part (b). This is wrong since unless you assume $\alpha < 1.5$, the usual calculation does not give you the above probability.

Further, if you do the calculation assuming that the algorithm stops at 2α vertices, there is a major issue. The point is, unless you go down to two vertices, the survival probability of different cuts are *not disjoint* and hence you cannot apply law of total probability directly. This is a conceptual error. Here is an alternate solution (directly from Karger's paper) assuming that 2α is an integer.

Let us revisit the survival probability as obtained in part (a), but now only till the iteration when 2α vertices remain. The probability of any specific α-approximate cut surviving till now is

$$\prod_{i=n}^{2\alpha+1}\left(1-\frac{2\alpha}{i}\right) = \binom{n}{2\alpha}^{-1}$$

But, as explained earlier, we cannot proceed as part(b) yet since we do not even have a cut. However, here is a way of creating a cut from here - make two sets and uniformly at random throw each surviving node in to either of them. Now, how many possible actual cuts can be

formed using these 2α remaining vertices ? The number is $2^{2\alpha-1}$. Hence, any specific cut can be obtained with a probability of $2^{1-2\alpha}$. Note that the probabilities of any two α-approximate surviving this (artificial) round are now disjoint, Hence, after this point, we can proceed as in part (b). Combining with the above probability calculation, the number of such cuts is at most $2^{2\alpha-1}\binom{n}{2\alpha}^{-1}$ which can be shown to be at most $n^{2\alpha}$.

Rubric: Pretty much +5 or 0.

You are not allowed to use any LLM for this problem