15-850: Advanced Algorithms **CMU, Spring 2024**
HW #4 (out Saturday Mar 30, 2024) **Due:** Friday April 19, 2024

Same as HW3: Collaboration in a group of 2-3 is encouraged. Please solve *two* of the four problems.

1. **Zero-Sum Games using LP Duality.** Recall the zero-sum game setup: we're given a matrix $M \in \mathbb{R}_{m\times n}$; if the row player plays a strategy $\mathbf{x} \in \Delta_m$ and the column player plays strategy $\mathbf{y} \in \Delta_n$, the payoff to the row player is $\mathbf{x}^\intercal M\mathbf{y}$.

   If we define $C(\mathbf{x}) = \min_{\mathbf{y}\in\Delta_n} \mathbf{x}^\intercal M\mathbf{y}$, and $R(\mathbf{y}) = \max_{\mathbf{x}\in\Delta_m} \mathbf{x}^\intercal M\mathbf{y}$, the minimax theorem proves that (a) for all $\mathbf{x}, \mathbf{y}$, $C(\mathbf{x}) \leq R(\mathbf{y})$, and moreover (b) there exist $\mathbf{x}^*, \mathbf{y}^*$ such that $C(\mathbf{x}^*) = R(\mathbf{y}^*)$.

   (a) Show an LP to compute $\max_{\mathbf{x}} C(\mathbf{x})$, the optimal strategy for the row player. (Hint: be careful, the definition of $C()$ has a min sitting in there, so you're looking to find $\max_{\mathbf{x}} \min_{\mathbf{y}} \mathbf{x}^\intercal M\mathbf{y}$, which certainly does not look like a linear program.)

   (b) Show an LP to compute $\min_{\mathbf{y}} R(\mathbf{y})$, the optimal strategy for the column player.

   (c) Show that you can, in fact, find LPs for both the above parts, such that the dual of the first LP is a solution to the second part.

   (d) (Do not submit.) Use weak duality to infer the first part of the minimax theorem, and strong duality to infer the second part.

2. **Solving Very Large LPs.** We saw that the Ellipsoid algorithm can solve LPs with an exponential number of constraints in polynomial time, given a strong separation oracle for the polytope. Ellipsoid can often help if we have many variables but few constraints. An example is the bin-packing problem, with n items $[n] = \{1, 2, \ldots, n\}$ and with item sizes $s_i \in \mathbb{Z}_+$. The bin size B is poly(n).

   (a) A *configuration* is a subset $C \subseteq [n]$ such that $s(C) := \sum_{i \in C} s_i \leq B$. Suppose each item i also has a *value* $v_i \in \mathbb{R}_{\geq 0}$. Show how to find the max-value configuration in poly(n) time. Your algorithm should work even if the values are not integers or not bounded by poly(n). (Hint: DP.)

   (b) (Do not submit.) Consider a bin-packing LP, with a variable x_C for each configuration C (meant to indicate if we choose configuration C or not).

$$\begin{aligned} & \min \textstyle\sum_C x_C & & \text{(P)} \\ & \textstyle\sum_{C: i \in C} x_C \geq 1 & & \forall \text{ items } i \in [n] \\ & x_C \geq 0 & & \forall \text{ configurations } C. \end{aligned}$$

   Show that the dual for this LP is:

$$\begin{aligned} & \max \textstyle\sum_{i \in [n]} y_i & & \text{(D)} \\ & \textstyle\sum_{i \in C} y_i \leq 1 & & \forall \text{ configurations } C \\ & y_i \geq 0 & & \forall \text{ items } i \in [n]. \end{aligned}$$

   Let OPT_{LP} denote the optimal solution value for these two LPs.

   (c) The strong separation problem for the dual is: given a purported solution $\hat{y} \in \mathbb{R}^n$, either correctly claim that $\hat{y}$ satisfies all the dual constraints, or output any one dual constraint that is not satisfied. Solve this dual strong separation problem in time poly(n).

   (d) Using this separation oracle, suppose Ellipsoid returns an optimal solution $y^* \in \mathbb{R}^n$ to the dual LP in poly(n). (There is no dependence on the numbers, because they are all 0 or 1.) Recall that during its run, Ellipsoid looks at only some m = poly(n) of the constraints (namely those which were returned by the strong separation oracle as being violated). Show that if $\mathcal{C}^* = \{C_1, C_2, \ldots, C_m\}$ is the list of configurations corresponding to those constraints, then y^* is also an optimal solution to the following poly(n)-sized LP, with $\sum_i y_i^* = OPT_{LP}$.

$$\begin{aligned} & \max \textstyle\sum_{i \in [n]} y_i & & \text{(SmallD)} \\ & \textstyle\sum_{i \in C} y_i \leq 1 & & \forall \text{ configurations } C \in \mathcal{C}^* \\ & y_i \geq 0 & & \forall \text{ items } i \in [n]. \end{aligned}$$

   (Hint: what happens if run Ellipsoid again on the smaller LP? Will its behavior change?)

   (e) Using duality again, show how to get a poly(n)-sized (primal) LP, which can be solved in poly(n) time, and whose solution can be extended to an optimal solution for the original primal LP (P).

To wrap up: if you have many constraints in your LP (but few variables), Ellipsoid works in poly(n) time if you can solve the primal separation problem. If you have many variables in your LP (but few constraints), you may still get an optimal solution to (P) in time poly(n), as long as you can solve the *dual* separation problem. This technique is essentially the same as *column-generation* idea, which is widely used in practice. The primal LP has an exponential number of columns and the dual separation oracle tells you which are the "interesting" columns to "generate" for your primal.

3. **(Hmm, That's Odd...)** To solve the max-weight perfect matching problem on general graphs, we need to optimize over the perfect matching polytope. In turn, this requires that we find a *separation oracle* for the odd-set constraints. Specifically, given edge weights $x \in \mathbb{R}^{|E|}$ with $x \geq 0$, we wish to find a set $S \subseteq V$ such that $|S|$ is odd, and $x(\partial S)$ is minimized, where $x(\partial S) := \sum_{i \in S, j \notin S} x_{ij}$. Then, comparing this min-odd-cut value to 1, we can find a violated constraint (if one exists). Assume that $|V|$ is even, else the LP has no feasible solution anyways.

   (a) A function $f : 2^V \to \mathbb{R}$ is called *submodular* if for all $A, B \subseteq V$, we have

   $$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

   Show $f(A) := x(\partial A)$ is submodular. Observe f is symmetric, i.e., $f(A) = f(V \setminus A)$.

   Define a *min-cut* to be a set $C \subseteq V$ such that $x(\partial C)$ is the least among all cuts where both $C, V \setminus C$ are non-empty, and a *min-odd-cut* to be a set $C \subseteq V$ with $|C|$ odd, where $x(\partial C)$ is the least among all cuts with odd size.

   (b) (Do not submit, instead submit part (c) instead.) Let C be a min-cut, and let $\overline{C}$ be its complement. Show that if $|C|$ is even, then there exists a min-odd-cut contained within either C or $\overline{C}$.

   We now extend part (b) slightly: Given a set T with an even number of vertices, define a *min-T-cut* to be a set $C \subseteq V$ where both $C \cap T, T \setminus C$ are non-empty, having smallest $x(\partial C)$. And a *min-T-odd cut* such that both $C \cap T, T \setminus C$ are odd, having smallest $x(\partial C)$.

   (c) Let C be a min-T-cut, and let $\overline{C}$ be its complement. Show that if $|C \cap T|$ is even, then there exists a min-T-odd-cut contained within either C or $\overline{C}$.

   (d) Give an algorithm to find a min-odd-cut in polynomial time. (Hint: when you recurse, think about how to use the ability to choose the set T. Also, how would you find a min-T-cut in polynomial time?)

4. **Covering using Small Sets.** In the (min-weight) set cover problem we're given a set system $(U, \mathcal{S})$, with each set $S \in \mathcal{S}$ having weight $w_S \geq 0$, and the goal is to pick some sets $\mathcal{S}' \subseteq \mathcal{S}$ so that their union equals U, and their total weight $\sum_{S \in \mathcal{S}'} w_S$ is minimized. We have $n := |U|$ elements and $m := |\mathcal{S}|$ sets.

   In lecture we solved the unweighted case using (i) a combinatorial analysis for greedy, and (ii) solving the LP optimally and then randomized rounding. Now suppose each set $S \in \mathcal{S}$ has size at most $B \leq n$, and sets have weights. We'll show $O(\log B)$-approximations for the weighted setting.

   (a) Show that the greedy algorithm has weight $O(\log B)$ times the weight of the optimal solution. Here the greedy algorithm picks a set S that maximizes the number of new elements covered by S, divided by its weight. (Hint: each time the greedy algorithm picks a set S, split the weight of set S evenly among its newly covered elements. The weight of the greedy solution equals the total weight assigned to the elements. Now bound the total weight assigned to the elements in a different way using the optimal set cover.)

   (b) For each element $e \in U$, let $S(e) \in \mathcal{S}$ be the lightest set that contains e. Show that $\frac{1}{B} \sum_e w_{S(e)} \leq OPT$. (This is unrelated to the part above.)

   (c) Give an algorithm that solves the LP and then picks some of the sets (randomly) based on the optimal LP solution, and has cost at most $O(\log B)$ times the optimal weight, in expectation. (You may need part (b) above.)