PRINCETON UNIVERSITY FALL '24 COS 521:ADVANCED ALGORITHMS

Homework 2

Out: *Oct 11* Due: *Oct 27*

Instructions:

- Upload your solutions (to the non-extra-credit) as *a single* PDF file (one PDF total) to Gradescope. Please anonymize your submission (do not list your name in the PDF title or in the document itself). If you forget, it's OK.
- If you choose to do extra credit, upload your solution to the extra credits as a single PDF file to Gradescope. Please again anonymize your submission.
- You may discuss ideas for solutions with any classmates, textbooks, the Internet, etc. Please attach a brief "collaboration statement" listing any collaborators at the end of your PDF. **You must write up your solutions individually.**
- For each problem, you should aim to keep your writeup below one page. For some problems, this may be infeasible, and for some problems you may write significantly less than a page. This is not a hard constraint, but part of the assignment is figuring out how to easily convince the grader of correctness, and to do so concisely. "One page" is just a guideline: if your solution is longer because you chose to use figures (or large margins, display math, etc.) that's fine.
- Each problem is worth twenty points (even those with multiple subparts).

Problems:

§1 Recall the max-flow problem from undergraduate algorithms: for a directed graph $G(V, E)$ with non-negative capacities c_e for every $e \in E$ and two special vertices s (source, with no incoming edges) and t (sink, with no outgoing edges), a *flow* in G is an assignment $f : E \to \mathbb{R}_{\geq 0}$ such that $f(e) \leq c_e$ for every edge and for every vertex $v \in V \setminus \{s, t\}$, the total incoming flow $\sum_{(u,v)\in E} f((u, v))$ equals the total outgoing flow $\sum_{(v,w)\in E} f((v, w))$. The task is to find a maximum flow f i.e., a flow f such that $\sum_{(s,u)\in E} f((s, u))$ is maximized.

(a) Show that the following LP is a valid formulation for computing the value of the maximum flow in G. There is a variable $f((u, v))$ for all $(u, v) \in E$.

$$\begin{aligned} \max & \sum_u f((u,t)) \\ & \forall e = (u,v) \in E,\ f((u,v)) \leq c_e \\ & \forall v \notin \{s,t\},\ \sum_{(u,v)\in E} f((u,v)) = \sum_{(v,w)\in E} f((v,w)) \\ & \forall e \in E,\ f(e) \geq 0 \end{aligned} \tag{1}$$

(b) Write the dual for the LP (1). Show that this dual LP computes the minimum *fractional* s-t cut in G. A fractional cut places each node v at some point y_v on the unit interval $[0, 1]$, with s placed at 0 and t placed at 1. The value of the fractional cut is $\sum_{(u,v)\in E(G)} c_e \cdot \max\{0, y_v - y_u\}$ (where c_e is the weight of edge e). Observe that if instead each $y_v \in \{0, 1\}$, that this is simply an s-t cut. Use strong LP duality to conclude the *fractional* max-flow min-cut theorem. That is, if the max-flow is C, there exists a fractional s-t cut of value C, and no fractional s-t cut of value $< C$.

(c) Devise a rounding scheme that takes as input a fractional min-cut of value C and outputs a true (deterministic) min-cut of value C. (Hint: try correlated randomized rounding — choose a threshold $c \in [0, 1]$ uniformly at random and set $S = \{v \mid y_v \leq c\}$. What can you say about the probability that a given edge is in this (randomized) cut?) Conclude the max-flow min-cut theorem.

§2 The maximum cut problem asks us to cluster the nodes of a graph $G = (V, E)$ into two disjoint sets X, Y so as to maximize the number of edges between these sets:

$$\max_{X,Y} \sum_{(i,j)\in E} \mathbb{1}[(i \in X, j \in Y) \vee (i \in Y, j \in X)]$$

Consider instead clustering the nodes into **three** disjoint sets X, Y, Z. Our goal is to maximize the number of edges between different sets:

$$\max_{X,Y,Z} \sum_{(i,j)\in E} \mathbb{1}[(i \in X, j \in Y \cup Z) \vee (i \in Y, j \in X \cup Z) \vee (i \in Z, j \in X \cup Y)]$$

Design an algorithm based on SDP relaxation that solves this problem with approximation ration greater then .7.

Note: In the Goemans-Williamson algorithm for maximum cut, we claimed that $\frac{2\theta}{\pi(1-\cos\theta)} \geq 0.878,\ \forall\theta \in [0, \pi]$. This is much easier to verify analytically (e.g. with a plot in MATLAB) than to prove formally. If similar quantities appear in your proof, feel free to bound them analytically, without proof.

Obtain the highest object value you can – partial credit will be given to any non-trivial solution, even if it obtains a weaker bound than .7.

§3 The Ellipsoid algorithm we saw in the lecture solves convex programs assuming a separation oracle. Here, we want to show the opposite. To be more specific, consider the following two tasks regarding a convex body $\mathcal{K}$:

- $\mathsf{OPTIMIZE}(\mathcal{K})$: given a vector $c \in \mathbb{R}^n$, output $\arg\max_{x\in\mathcal{K}} c^\top x$;
- $\mathsf{SEPARATE}(\mathcal{K})$: given a point $x \in \mathbb{R}^n$, output either $x \in \mathcal{K}$, or a separating hyperplane.

We are going to show that if for a specific convex body $\mathcal{K}$, there is a polynomial time algorithm for $\mathsf{OPTIMIZE}(\mathcal{K})$, then there is a polynomial time algorithm for $\mathsf{SEPARATE}(\mathcal{K})$.

(a) Suppose for a given x, we can solve the following LP with *infinitely many* constraints (finding the optimal w and T). Show that we can use such an algorithm to solve $\mathsf{SEPARATE}(\mathcal{K})$.

$$\begin{aligned} \text{Variables: } & w \in \mathbb{R}^n, T \in \mathbb{R} \\ \text{Maximize: } & w^\top x - T \\ \text{Subject to: } & \forall y \in \mathcal{K}, w^\top y \leq T \\ & -1 \leq T \leq 1 \end{aligned}$$

(b) Design a polynomial time separation oracle for the above LP using $\mathsf{OPTIMIZE}(\mathcal{K})$, and conclude.

§4 Describe separation oracles for the following convex sets. Your oracles should run in linear time, assuming that the given oracles run in linear time (so you can make a constant number of black-box calls to the given oracles).

(a) The ℓ_1 ball, $\{x : \|x\|_1 \leq 1\}$. Recall that $\|x\|_1 = \sum_{i=1}^{d} |x_d|$.

(b) Any convex set A that we have a projection oracle for. I.e. we have an oracle to compute $\arg\min_{x \in A} \|x - y\|_2$ for any y.

(c) The ϵ-neighborhood, E of any convex set A:

$$E = \{x : \exists y \in A \text{ with } \|x - y\|_2 \leq \epsilon\},$$

given a projection oracle for A.

§5 In class we designed a 3/4-approximation for MAX-2SAT using LP rounding. The MAX-SAT problem is similar except for the fact that the clauses can contain any number of literals. Formally, the input consists of n boolean variables $x_1, x_2, \ldots, x_n$ (each may be either 0 (false) or 1 (true)), m clauses $C_1, C_2, \ldots, C_m$ (each of which consists of disjunction (an "or") of some number variables or their negations) and a non-negative weight w_i for each clause. The objective is to find an assignment of 1 or 0 to x_is that maximize the total weight of satisfied clauses. As we saw in the class, a clause is satisfied if one of its non-negated variable is set to 1, or one of the negated variable is set to 0. You can assume that no literal is repeated in a clause and at most one of x_i or $\neg x_i$ appears in any clause.

(a) Generalize the LP relaxation for MAX-2SAT seen in the class to obtain a LP relaxation of the MAX-SAT problem.

(b) Use the standard randomized rounding algorithm (the same one we used in class for MAX-2SAT) on the LP-relaxation you designed in part (1) to give a $(1-1/e)$ approximation algorithm for MAX-SAT. Recall that clauses can be of any length. (Hint: there is a clean way to resolve "the math" without excessive calculations).

(c) A naive algorithm for MAX-SAT problem is to set each variable to true with probability 1/2 (without writing any LP). It is easy to see that this *unbiased randomized* algorithm of MAX-SAT achieves 1/2-approximation in expectation.

Show the algorithm that returns the best of two solutions given by the randomized rounding of the LP and the simple unbiased randomized algorithm is a 3/4-approximation algorithm of MAX-SAT. (Hint: it may help to realize that in fact *randomly* selecting one of these two algorithms to run also gives a 3/4-approximation in expectation).

(d) Using the previous part (and in particular, the hint) for intuition, design a direct rounding scheme of your LP relaxation to get a 3/4-approximation (that is, design a function $f(\cdot)$ which assigns a literal x_i to be true independently with probability $f(z)$ when the corresponding variable z_i in your LP relaxation is equal to z). (Hint: here, it may get messy to fully resolve the calculations. You will get full credit if you state the correct rounding scheme and clearly state the necessary inequalities for the proof. You should also attempt to show that the inequalities hold for your own benefit, but not for full credit).

§6 (extra credit) Consider the following problem: there are $n > k$ independent (but not identically distributed) non-negative random variables $X_1, \ldots, X_n$ drawn according to distributions $D_1, \ldots, D_n$. Initially, you know each D_i but none of the X_is.

Starting from $i = 1$, each X_i is revealed one at a time. Immediately after it is revealed, you must decide whether to "accept i" or "reject i," before seeing the next X_{i+1}. You may accept at most k elements in total (that is, once you've accepted k times, you must reject everything that comes after). Your reward at the end is $\sum_{i|i \text{ was accepted}} X_i$.

(a) For general k, design a policy that guarantees expected reward at least $(1 - O(\sqrt{\frac{\ln(k)}{k}})) \cdot \mathbb{E}_{X_1,\ldots,X_n \leftarrow D_1,\ldots,D_n}[\sum_{j=1}^{k} X_{r(j)}]$, where r is a permutation from $[n]$ to $[n]$ satisfying $X_{r(1)} \geq X_{r(2)} \geq \ldots \geq X_{r(n)}$ (i.e. the policy gets expected reward at least $(1 - O(\sqrt{\frac{\ln(k)}{k}}))$ times the expected sum of top k weights, which is the best you could do even if you knew all the weights up front).[1]

Hint: Try to set up a simple policy that can be analyzed using a Chernoff bound.

(b) Come up with an example showing that it is not possible to improve the above guarantee beyond $(1 - \Omega(1/\sqrt{k}))$ (which is optimal - no need to prove this).

Hint: an example exists whose complete proof should fit in half a page. You may use without proof the fact that if X is the number of coin flips which land heads from k independent fair coin flips, then $\mathbb{E}[|X - k/2|] = \Theta(\sqrt{k})$.

[1] For simplicity of exposition, you may assume that each random variable is continuous. You may want to think about how to adapt your analysis in case the random variables are discrete, but do not need to write this in your proof for full credit.