PRINCETON UNIVERSITY FALL '19 COS 521:ADVANCED ALGORITHMS

Homework 2

Out: *Sep 30* Due: *Oct 14*

Instructions:

- Upload your solutions (to the non-extra-credit) as *a single* PDF file (one PDF total) to Mechanical TA. 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 Mechanical TA. Please again anonymize your submission.
- You may collaborate with any classmates, textbooks, the Internet, etc. Please attach a brief "collaboration statement" listing any collaborators at the end of your PDF (if you forget, it's OK). You should 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).

§1 (Approximate LP Solving via Multiplicative Weights) This exercise develops an algorithm to approximately solve Linear Programs.

Consider the problem of finding if a system of linear inequalities as below admits a solution - i.e., whether the system is feasible. This is an example of a feasibility linear program and while it appears restrictive, one can use it solve arbitrary linear programs to obtain approximate solutions.

$$\begin{aligned} a_1^\top x &\geq b_1 \\ a_2^\top x &\geq b_2 \\ &\vdots \\ a_m^\top x &\geq b_m \\ x_i &\geq 0 \ \forall \ i \in [n] \\ \sum_{i=1}^{n} x_i &= 1. \end{aligned} \tag{1}$$

(a) Design a simple algorithm to solve the following linear program, which has only two non-trivial constraints. Below, the weights $w_1, w_2, \ldots, w_m$ are fixed (along

with the vectors $a_j^\top$ and numbers b_j), and $x_1, \ldots, x_n$ are the variables.

$$\begin{aligned} \max & \sum_{j=1}^{m} w_j (a_j^\top x - b_j) \\ & x_i \geq 0 \ \forall i \in [n] \\ & \sum_{i=1}^{n} x_i = 1. \end{aligned} \tag{2}$$

(b) Prove that if there exist non-negative weights $w_1, w_2, \ldots, w_m$ such that the value of the program above is negative, then the system (1) is infeasible.

(c) The above setting of finding weights that certify infeasibility of (1) might remind you of the setting of weighting the experts via multiplicative weights update rule discussed in the class. Use these ideas to obtain an algorithm that a) either finds a set of non-negative weights certifying infeasibility of LP in (1) or b) finds a solution x that approximately satisfies all the constraints in (1), i.e., for each $1 \leq j \leq m$, $a_j^\top x - b_j \geq -\epsilon$, and for each $1 \leq i \leq n$, $x_i \geq 0$, and $\sum_{i=1}^n x_i = 1$. Prove that your algorithm terminates after solving $O(\ln(m)/\varepsilon^2)$ LPs of form (2) (you do not need to analyze the remaining runtime).
(Hint: Identify m "experts" - one for each inequality constraint in (1) and maintain a weighting of experts (starting with the uniform weighting of all 1s, say) for times $t = 0, 1, \ldots,$ - these are your progressively improving guesses for the weights. Solve (2) using the weights at time t. If the value of (2) is negative, you are done, otherwise think of the "cost" of the j^{th} expert as $a_j^\top x^{(t)} - b_j$ where $x^{(t)}$ is the solution to the LP (2) at time t and update the weights.)

§2 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 $v \in V$, $\sum_{(u,v)\in E} f((u,v)) = \sum_{(v,u)\in E} f((v,u))$. 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 maximum flow in G. There is a variable $f(u,v)$ for all $(u,v) \in E$. (Hint: below, the inequality is not a typo. You should show that it is w.l.o.g. to replace the equality with inequality, as this will make it easier to reason about later parts.)

$$\begin{aligned} \max & \sum_{u} f(u,t) \\ & \forall e = (u,v) \in E, \ f(u,v) \leq c_e \\ & \forall v \notin \{s,t\}, \ \sum_{u} f(u,v) \geq \sum_{w} f(v,w) \\ & \forall e \in E, \ f(e) \geq 0 \end{aligned} \tag{3}$$

(b) Write the dual for the LP (3). Show that this dual LP computes the minimum *fractional* s-t cut in G (a cut that separates s and t in G and minimizes the sum of the capacities c_e of the edges going across it. You will know what a fractional s-t cut is once you take the dual: every node isn't entirely on the s side or the t side, but rather partially on each). 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: there is a simple rounding scheme that works, but it is not a rounding scheme we have already seen in class.)

§3 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. (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).

§4 (Firehouse location) Suppose we model a city as an m-point finite metric space with

$d(x, y)$ denoting the distance between points x, y. These $\binom{m}{2}$ distances (which satisfy triangle inequality) are given as part of the input. The city has n houses located at points $v_1, v_2, \ldots, v_n$ in this metric space. The city wishes to build k firehouses and asks you to help find the best locations $c_1, c_2, \ldots, c_k$ for them, which can be located at any of the m points in the city. The *happiness* of a town resident with the final locations depends upon his distance from the closest firehouse. So you decide to minimize the cost function $\sum_{i=1}^{n} d(v_i, u_i)$ where $u_i \in \{c_1, c_2, \ldots, c_k\}$ is the firehouse closest to v_i. Describe an LP-rounding-based algorithm that runs in $poly(m)$ time and solves this problem approximately. If OPT is the optimum cost of a solution with k firehouses, your solution is allowed to use $O(k \log n)$ firehouses and have cost at most OPT.[1]

§5 (extra credit) Design an algorithm that uses k firehouses but has cost $O(\text{OPT})$. (Needs a complicated dependent rounding; you can also try other ideas.) Partial credit available for partial progress.

§6 (extra credit) In a *combinatorial auction* there are n bidders and m items. Bidder i has a monotone valuation function $v_i(\cdot)$ where $v_i(S)$ denotes their value for set S of items (and $v_i(S \cup T) \geq v_i(S)$ for all S, T). A *Walrasian Equilibrium* is a price for each item $\vec{p}$ such that:

- Each buyer i selects to purchase a set $B_i \in \arg\max_S \{v_i(S) - \sum_{j \in S} p_j\}$.
- The sets B_i are disjoint, and $\cup_i B_i = [m]$.

Prove that a Walrasian equilibrium exists for $v_1, \ldots, v_n$ if and only if the optimum of the LP relaxation below (called the *configuration LP*) is achieved at an integral point (i.e. where each $x_{i,S} \in \{0, 1\}$). Hint: use strong duality!

$$\begin{aligned} \max \sum_i \sum_S v_i(S) \cdot x_{i,S} \\ \forall i, \ \sum_S x_{i,S} = 1 \\ \forall j, \ \sum_{S \ni j} \sum_i x_{i,S} \leq 1 \end{aligned} \tag{4}$$

Also, come up with an example of two valuation functions v_1, v_2 over two items where a Walrasian equilibrium doesn't exist.

[1] The term for an approximation guarantee like this is *resource augmentation* — the solution is as good as the optimum, but it requires additional firehouses.