Advanced Approximation Algorithms **CMU 15-854B, Spring 2008**

HOMEWORK 4
Due: Thursday, March 20

---

1. Online Set Cover. In the online (unweighted) set cover problem, you are given a set system $(U, \mathcal{S})$ in advance with n elements and m sets. You just don't know which of the elements of U you actually want to cover. An unknown sequence σ of elements $e_1, e_2, \ldots, e_t$ arrives one by one: when an element e_i arrives and is not already covered by the sets picked thus far by us, we have to pick a new set $S_{(i)}$ containing this element. (We are allowed to pick multiple sets $S_{(i,1)}, S_{(i,2)}, \ldots$ instead of just one set, if we deem fit to do so.)

Let $\mathsf{Set}(\sigma) \subseteq U$ denote the set of elements that appear in σ, and let $\mathsf{OPT}(\sigma)$ be the fewest number of sets in $\mathcal{S}$ that cover the elements of $\mathsf{Set}(\sigma)$ — note that OPT is purely an offline quantity. Suppose $\mathcal{A}$ is an online algorithm for Set-Cover; let $\mathcal{A}(\sigma)$ be number of sets picked by the algorithm when given the sequence σ. (One can extend the definition of the problem to the *weighted* case, where each set has a cost $c(S_i)$, and both $\mathcal{A}$ and OPT now minimize the total cost, etc.) The goal is to develop an algorithm to minimize the *competitive ratio*: this is the smallest value ρ such that on every input sequence σ, the ratio

$$\frac{\mathcal{A}(\sigma)}{\mathsf{OPT}(\sigma)} \le \rho.$$

If the algorithm $\mathcal{A}$ is randomized, replace the numerator by $\mathbf{E}[\mathcal{A}(\sigma)]$.

a. **Example A:** Let $n = 2^d$, the universe be $\{0,1\}^d$, and the set system is $\mathcal{S} = \{S_i\}_{i=0}^d$ with $S_0 = \{0^d\}$, and for $1 \le i \le d$, S_i is the set of all strings with 1 in the i^{th} place. Given any deterministic algorithm $\mathcal{A}$, show that the adversary can choose an input sequence σ such that $\mathsf{OPT}(\sigma) = 1$ but $\mathcal{A}(\sigma) = d$.

   This shows that no deterministic algorithm for this problem has a competitive ratio of better than $d = \log_2 n$.

b. **Example B:** Let the universe be $U = [n]$, and the set system be the collection of all subsets of U with $\sqrt{n}$ elements (hence $m = |\mathcal{S}| = \binom{n}{\sqrt{n}}$). Given any deterministic algorithm $\mathcal{A}$, show that the adversary can choose an input sequence σ such that $\mathsf{OPT}(\sigma) = 1$ but $\mathcal{A}(\sigma) = \sqrt{n}$.

   This shows that no deterministic algorithm for this problem has a competitive ratio of better than $\Omega(\frac{\log m}{\log n})$.

c. Given a graph $G = (V, E)$, give a deterministic online algorithm for Vertex-Cover with a competitive ratio of 2. For the weighted case, give a deterministic or randomized 2-competitive algorithm for the problem. (You can use results you proved in the previous homework.)

d. Suppose we want to cover the subset $U' \subseteq U$. Recall the natural LP relaxation of the (unweighted) set cover problem, and its dual.

$$\begin{array}{llll} \min \sum_{S \in \mathcal{S}} x_S & & \max \sum_{e \in U'} y_e & (1) \\ \text{s.t. } \sum_{S: e \in S} x_S \ge 1 \quad \forall e \in U' & & \text{s.t. } \sum_{e \in S} y_e \le 1 \quad \forall S \in \mathcal{S} & (2) \\ x \ge 0 & & y \ge 0 & (3) \end{array}$$

   We will develop a primal-dual online algorithm to get an approximate *fractional* solution to this LP.

   Recall that we don't up front know the elements we want to consider: initially $U' = \emptyset$, and when an element e arrives online, we get a new constraint ($\sum_{S: e \in S} x_S \ge 1$) in the primal, and a new variable y_e gets added to constraints in the dual for each $S \ni e$.

   We will construct (feasible) primal solution x and a (possibly infeasible) dual solution y such that $\sum_S x_S \le 2 \sum_e y_e$, and moreover $y/O(\log m)$ is feasible. Explain why this proves that $\sum_S x_S$ is at most $O(\log m)$ times the optimal LP solution.

e. Here is the procedure. Initialize $x_S = 1/2m$ for all S.

- When element e arrives, set $y_e \leftarrow 0$.
- While $\sum_{S:e\in S} x_S < 1$
  double all the x_S for sets $S \ni e$; also $y_e \leftarrow y_e + 1$.

Show that $\sum_S x_S \le \frac{1}{2} + \sum_e y_e$ at all times. Show that after the first element has arrived, $\sum_S x_S \le \frac{3}{2}\sum_e y_e$.

f. Show that, at any point in time, $y/(\log_2 2m)$ is a feasible dual solution.

g. **[Optional]** We just showed how to obtain a fractional solution to the minimum set cover which has value at most $O(\log m)$ times the optimal LP solution. Using randomized rounding (if an uncovered element e arrives and causes sets $S \ni e$ to be doubled for zero or more times, at the end of the process, independently pick each set $S \ni e$ with probability $O(\log n) \cdot x_S$), show that the expected cost of this process is at most $O(\log n)$ times the online LP solution, and hence at most $O(\log n \log m)$ times the optimal number of sets to cover the elements in the input sequence so far.

2. Max-Cut: Cycles and Gaps. Let C_k denote the cycle graph on k vertices, with each edge having equal weight $1/k$.

a. Determine the Max-Cut in C_k for every $k \ge 3$.

b. Show that C_5 leads to a Max-Cut SDP integrality gap of factor

$$\frac{4/5}{1/2 - (1/2)\cos(4\pi/5)} \approx .884.$$

c. The "triangle inequalities" are the constraints

$$\|v_i - v_j\|^2 + \|v_j - v_k\|^2 \ge \|v_i - v_k\|^2, \qquad (4)$$

for every three vectors v_i, v_j, v_k in the solution. Show that we can express these constraints as linear inequalities on dot products. Do the same thing for the case when we replace v_j by $-v_j$ in (4).

d. Show that it is valid to add all of the triangle inequality constraints (including the ones with $-v_j$) into the Max-Cut SDP (i.e., that it is still a relaxation of Max-Cut).

e. Show that with all triangle inequality constraints included in the SDP, $\mathrm{Sdp}(C_k) = \mathrm{Opt}(C_k)$ for all $k \ge 3$. (Hint: mess around with adding triangle inequalities together.)

f. When the graph is C_7, find an optimal feasible solution to the SDP with triangle inequalities in which each vector is in $\{-\frac{1}{\sqrt{7}}, +\frac{1}{\sqrt{7}}\}^7$.

g. Suppose an algorithm did Goemans-Williamson rounding on the vector solution in (f). Show that

$$\frac{\mathbf{E}[\text{value achieved}]}{\text{optimal value}} = \frac{\frac{7}{5}\arccos(-5/7)}{\pi} \approx .8788.$$

3. 2-Coloring 3-Uniform Hypergraphs ("2C3U"). The Max-2C3U problem is the following: The input is a 3-uniform hypergraph $G = (V, E)$. The task is to output a 2-coloring $f : V \to \{-1, 1\}$. We say that a hyperedge $\{u, v, w\}$ is "legally" colored if it is not monochromatic; i.e., $f(u), f(v), f(w)$ are not all equal. The goal is to maximize the fraction of legally colored hyperedges.

a. Formulate an SDP relaxation for this problem. (Hint: no constraints are necessary beyond $v_i \cdot v_i = 1$.)

b. Show that Goemans-Williamson-style rounding yields a .878-factor approximation algorithm.

c. Show that the same algorithm is actually a 1 vs. $3\arccos(-1/3)/2\pi \approx .912$ approximation algorithm. (Hint: it's possible you'll need calculus to minimize a multivariable expression.)

d. Can your SDP relaxation have value greater than 1? If not, prove it can't. If so, can you add a "valid" constraint which will ensure it can't?

4. Duality. Let G be a weighted graph on n vertices, $G = (V, E, \{w_{ij}\})$.

a. Show that
$$\text{Max-Cut}(G) = \max \frac{n}{4}\, x^\top L x$$
$$\text{subject to } x \in \left\{-\tfrac{1}{\sqrt{n}}, \tfrac{1}{\sqrt{n}}\right\}^n,$$
where L is the "Laplacian" matrix of G, which has $L_{ij} = -w_{ij}$ for $i \neq j$ and $L_{ii} = \sum_k w_{ik}$.

b. Let $\mathcal{U} = \{u \in \mathbb{R}^n : \sum_{i=1}^n u_i = 0\}$. Define $g : \mathcal{U} \to \mathbb{R}$ by
$$g(u) = \frac{n}{4} \max\{x^\top (L + \text{diag}(u))x : \|x\| = 1\},$$
where $\text{diag}(u)$ denotes the matrix with the vector u along the diagonal and 0's elsewhere. Show that $\text{Max-Cut}(G) \leq g(u)$ for all $u \in \mathcal{U}$. Conclude that $\text{Max-Cut}(G) \leq D(G) := \min_{u \in \mathcal{U}} g(u)$.

c. Show that $g(u)$ can be computed in polynomial time. Also, show that g is convex on $\mathcal{U}$. (Hint: eigenvalues.)

Remark: It follows that computing $D(G)$ involves minimizing a polynomial-time-computable convex function over the hyperplane $\mathcal{U}$. There are several methods for doing this in polynomial time; e.g., gradient descent. Hence $D(G)$ is a poly-time-computable upper bound on $D(G)$. In fact, the minimization problem $D(G)$ is the dual of the Goemans-Williamson SDP formulation of Max-Cut, and $D(G) = \text{Sdp}(G)$!

5. A Pairing Lemma and Priority Steiner Tree. Given a tree $T = (V, E)$ and a set of $2k$ points $S \subseteq V$, you can pair up the points in S into k pairs $\{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{2k-1}, v_{2k}\}\}$ such that the (unique) paths between each of these pairs of nodes are mutually edge-disjoint.

Clearly state the greedy algorithm for the Priority Steiner tree problem (sketched in Lecture 12), and give a full proof that it is an $O(\log n)$ approximation algorithm. (The above pairing lemma might be useful.)

6. Tree Embeddings and Group Steiner. Given a metric space $\mathcal{M} = (V, \delta)$ on $|V| = n$ points, let $\mathcal{T}$ be the set of trees $T = (V, E_T)$ on the vertex set V with edge lengths $\ell : E_T \to \mathbb{R}$ such that each edge $e = \{u, v\} \in E_T$ has length $\ell(\{u, v\}) \geq \delta(u, v)$ (at least the distance between its endpoints in the metric space $\mathcal{M}$).

a) Show that for *any pair* of vertices $u, v \in V$ and any tree $T \in \mathcal{T}$, the shortest path distance in T (denoted by $d_T(u, v)$) is at least $\delta(u, v)$.

A probability distribution π on this set of trees $\mathcal{T}$ is said to α-approximate the metric $\mathcal{M}$ if for every $u, v \in V$,
$$\mathbf{E}_{T \leftarrow \pi}[d_T(u, v)] \leq \alpha \cdot \delta(u, v). \tag{5}$$
I.e., for any two points, the expected distance in a random tree (drawn from this distribution) is at most α times what it was in (V, δ).

By the *tree-embedding* results of [Bartal *FOCS 1996*, Fakcharoenphol-Rao-Talwar *STOC 2003*], for any metric $\mathcal{M}$ on n points one can find a distribution $\pi = \pi(\mathcal{M})$ such that $\alpha = \alpha_{FRT} = O(\log n)$. In fact, this embedding is efficient: in poly-time one can output a random tree from this distribution π.

- An instance of the Group Steiner tree problem consists of a graph $G = (V, E)$, a "root" vertex $r \in V$, groups $g_1, g_2, \ldots, g_k$ with each $g_i \subseteq V$; in class we considered instances where the graph G was a tree, and gave an ρ_{GST}-approximation algorithm, where $\rho_{GST} = O(\log n \log |g_{\max}|)$.

  Prove that you can use the above tree-embedding results and the approximation for the group Steiner tree (on trees) seen in class to get a randomized algorithm whose expected cost is $O(\rho_{GST} \cdot \alpha_{FRT})$ times the cost of the optimal tree.

(Hints: apply the tree embedding to the shortest-path metric d_G of the graph G. Also, where in your proof do you use the fact that distances in the random tree T are at least those in d_G?)