Problem Set II

1 PTAS for MaxCut. *(34 pts)* In this problem, you will design and analyze a *polynomial-time approximation scheme*[1] for the MaxCut problem restricted to graphs that are "dense" and "almost bipartite". Hereafter, let $G = (V, E)$ be an undirected graph on n vertices. We denote by $\textsc{opt}(G) = \max_{U\subseteq V} |E(U, V \setminus U)|$ the maximum size of a cut, and let $Q^* \subseteq V$ be a maximum cut of G, i.e. $|E(Q^*, V \setminus Q^*)| = \textsc{opt}(G)$. For any cut $U \subseteq V$, we use $\mathbb{1}_Q$ to denote the vector in $\{\pm 1\}^V$ with $(\mathbb{1}_U)_u = 1$ if $u \in U$ and $(\mathbb{1}_U)_u = -1$ if $u \notin U$.

1a *(10 pts)* For any assignment $\sigma \in \{\pm 1\}^V$, we define for each $u \in V$ the *bias* of u with respect to σ as $\beta(u; \sigma) = -\sigma_u \sum_{v\in N(u)} \sigma_v$. Furthermore, for a tuple of t vertices $S = (s_1, \ldots, s_t) \in V^t$, we define the *$S$-projected bias* of u with respect to σ as $\hat\beta_S(u; \sigma) = -\sigma_u \sum_{i\in[t]:\, s_i\in N(u)} \sigma_{s_i}$.

Consider sampling S by picking t vertices uniformly at random from V with replacement. Show that for any assignment $\sigma \in \{\pm 1\}^V$ and any vertex $u \in V$ such that $\beta(u; \sigma) > 0$, we have

$$\Pr_S\left[\hat\beta_S(u;\sigma) \le 0\right] \le \exp\left(-\frac{t\cdot(\beta(u;\sigma))^2}{2n^2}\right).$$

Hint: for m independent random variables $Y_1, \ldots, Y_m$ over $[-1, 1]$, and any $\theta > 0$, we have $\Pr[\sum_{i=1}^m (Y_i - \mathbb{E}[Y_i]) \ge \theta] \le \exp(-\theta^2/(2m))$ (a.k.a. Hoeffding's inequality for independent bounded variables).

Solution: First note that

$$\hat\beta_S(u;\sigma) = -\sigma_u \sum_{i\in[t]} \mathbb{1}\{s_i \in N(u)\}\cdot\sigma_{s_i} = \sum_{i\in[t]}\left(-\sigma_u \sum_{v\in N(u)} \mathbb{1}\{s_i = v\}\cdot\sigma_v\right),$$

so by linearity of expectation we have

$$\mathop{\mathbb{E}}_S\left[\hat\beta_S(u;\sigma)\right] = \sum_{i\in[t]}\left(-\sigma_u \sum_{v\in N(u)} \Pr_S[s_i = v]\sigma_v\right) = \frac{1}{n}\sum_{i\in[t]}\left(-\sigma_u\sum_{v\in N(u)}\sigma_v\right) = \frac{t}{n}\beta(u;\sigma).$$

[1] The term "polynomial-time approximation scheme" (PTAS, for short) generically refers to a polynomial-time algorithm which computes a $(1-\epsilon)$-approximate solution (or $1+\epsilon$, for minimization problems) in polynomial time for any constant $\epsilon > 0$.

If we define $X_i = -\sigma_u \sum_{v\in N(u)} \mathbb{1}\{s_i = v\} \cdot \sigma_v$ for each $i \in [t]$, we can write

$$\hat{\beta}_S(u;\sigma) = \sum_{i\in[t]} X_i\,,$$

i.e. $\hat{\beta}_S(u;\sigma)$ is a sum of t independent random variables in the range $[-1,1]$. We then apply Hoeffding's inequality for bounded variables with $m = t$, $Y_i = -X_i$, and $\theta = \frac{t}{n}\beta(u;\sigma)$. Thus, we get

$$\begin{aligned}\Pr_S\left[\hat{\beta}_S(u;\sigma) \le 0\right] &= \Pr\left[\sum_{i\in[t]} Y_i \ge 0\right]\\ &= \Pr\left[\sum_{i\in[t]} Y_i - \mathbb{E}[Y_i] \ge -\sum_{i\in[t]} \mathbb{E}[Y_i]\right]\\ &= \Pr\left[\sum_{i\in[t]} Y_i - \mathbb{E}[Y_i] \ge \theta\right]\\ &\le \exp\left(-\frac{\theta^2}{2t}\right)\\ &\le \exp\left(-\frac{\left(\frac{t}{n}\beta(u;\sigma)\right)^2}{2t}\right)\,,\end{aligned}$$

hence the claim.

1b *(10 pts)* Show that for any cut $Q \subseteq V$, one has

$$\textsc{opt}(G) - |E(Q, V\setminus Q)| \le \sum_{u\in V:\,(\mathbb{1}_{Q^*})_u \ne (\mathbb{1}_Q)_u} \beta(u;\mathbb{1}_{Q^*}) + 2\left(|E| - \textsc{opt}(G)\right)\,.$$

Solution: Let $\sigma = \mathbb{1}_{Q^*}$ and $\rho = \mathbb{1}_Q$, and define the set of mismatched vertices $M = \{u \in V : \rho_u \ne \sigma_u\}$. Then

$$\begin{aligned}\textsc{opt}(G) - |E(Q, V\setminus Q)| &= \sum_{uv\in E} \mathbb{1}\{\sigma_u \ne \sigma_v\} - \mathbb{1}\{\rho_u \ne \rho_v\}\\ &= \sum_{u\in M}\sum_{v\in N(u)\setminus M} \mathbb{1}\{\sigma_u \ne \sigma_v\} - \mathbb{1}\{\rho_u \ne \rho_v\}\\ &= \sum_{u\in M}\sum_{v\in N(u)\setminus M} \mathbb{1}\{\sigma_u \ne \sigma_v\} - \mathbb{1}\{\sigma_u = \sigma_v\}\,.\end{aligned}$$

Now we can add and subtract $\mathbb{1}\{\sigma_u \ne \sigma_v\} - \mathbb{1}\{\sigma_u = \sigma_v\}$ for every $u \in M$ and $v \in$

$N(u) \cap M$. Then, using the fact that $\mathbb{1}\{\sigma_u \neq \sigma_v\} - \mathbb{1}\{\sigma_u = \sigma_v\} = -\sigma_u\sigma_v$ we have

$$\begin{aligned}\textsc{opt}(G) - |E(Q, V \setminus Q)| &= \sum_{u\in M}\sum_{v\in N(u)\setminus M}(-\sigma_u\sigma_v) + \sum_{u\in M}\sum_{v\in N(u)\cap M}(-\sigma_u\sigma_v) - \sum_{u\in M}\sum_{v\in N(u)\cap M}(-\sigma_u\sigma_v) \\ &= \sum_{u\in M}\left(-\sigma_u\sum_{v\in N(u)}\sigma_v\right) + 2\sum_{uv\in E(M)}\sigma_u\sigma_v \\ &= \sum_{u\in M}\beta(u;\sigma) + 2\sum_{uv\in E(M)}\sigma_u\sigma_v\,,\end{aligned}$$

where the last equality followed by definition of β.

We now upper-bound the second term in the right-hand side. Notice that all the positive terms in $\sum_{uv\in E(M)}\sigma_u\sigma_v$ correspond edges that are not cut by Q^*, so

$$2\sum_{uv\in E(M)}\sigma_u\sigma_v \leq 2\left(|E| - \textsc{opt}(G)\right)\,.$$

1c *(14 pts)* Fix a constant $c > 0$. Describe and prove the correctness of a randomized algorithm that, given an accuracy parameter $\epsilon > 0$ and an undirected n-vertex graph $G = (V, E)$ with $|E| \geq cn^2$ (i.e. G is *dense*) and $\textsc{opt}(G) \geq (1-\epsilon)|E|$ (i.e. G is *almost bipartite*), runs in time $2^{O(\epsilon^{-2}\log(1/\epsilon))}\text{poly}(n)$ and outputs a cut $Q \subseteq V$ such that

$$\Pr\left[|E(Q, V\setminus Q)| \geq (1-O(\epsilon))\textsc{opt}(G)\right] \geq \frac{2}{3}\,.$$

Hint: the analysis might require arguing that for every $u \in V$ one has $\beta(u;\mathbb{1}_{Q^*}) \geq 0$ (that is, very $u \in V$ has at least as many neighbors on the opposite side of Q^* as compared to the side where u lies).

Solution: Set $t = C\epsilon^{-2}\log(1/\epsilon)$ for a large enough constant $C > 0$. The algorithm is the following:

1. Sample $S = (s_1, \ldots, s_t)$ by drawing each s_i uniformly from V, and let $\mathcal{S} = \cup_{i\in[t]}\{s_i\}$.
2. For each assignment $z \in \{\pm 1\}^{\mathcal{S}}$, extend z to an assignment ρ for V as follows.
   (a) For each $u \in \mathcal{S}$, copy $\rho_u = z_u$.
   (b) For each $u \in V \setminus \mathcal{S}$, set $\rho_u \in \{\pm 1\}$ such that $-\rho_u\sum_{i\in[t]:s_i\in N(u)} z_{s_i} \geq 0$.
3. Return the best among all the assignments ρ generated as above.

The running time easily follows, so we proceed with analyzing the approximation quality. Fix a maximum cut Q^*, and let $\sigma = \mathbb{1}_{Q^*}$ be its $\{\pm 1\}$ indicator vector. No matter what is the draw of S, the algorithm will consider an assignment z to $\mathcal{S}$ that is the restriction of $\mathbb{1}_{Q^*}$ to the vertices in $\mathcal{S}$. Let ρ be the assignment produced by the algorithm when such z is considered, and denote by Q the corresponding cut. Now observe that $\sigma_u \neq \rho_u$ implies $\hat{\beta}_S(u;\sigma) \leq 0$ for every $u \in V$. Let $\tau = c\epsilon n/100$. We then get

$$\textsc{opt}(G) - |E(Q, V\setminus Q)| \leq \sum_{u\in V:\,\sigma_u\neq\rho_u}\beta(u;\sigma) + 2\epsilon|E|$$

Using the fact that the biases are all non-negative for an optimal cut, we upper-bound the firs term in the right-hand side as

$$\begin{aligned}\sum_{u\in V:\,\sigma_u\neq\rho_u}\beta(u;\sigma) &\leq \sum_{u\in V:\beta(u;\sigma)\leq\tau}\beta(u;\sigma)+\sum_{u\in V:\beta(u;\sigma)>\tau}\mathbb{1}\{\hat{\beta}_S(u;\sigma)\leq 0\}\cdot n\\ &\leq c\frac{\epsilon}{100}n^2+n\cdot\sum_{u\in V:\beta(u;\sigma)>\tau}\mathbb{1}\{\hat{\beta}_S(u;\sigma)\leq 0\}\,.\end{aligned}$$

Note that

$$\mathop{\mathbb{E}}_{S}\left[\sum_{u\in V:\beta(u;\sigma)>\tau}\mathbb{1}\{\hat{\beta}_S(u;\sigma)\leq 0\}\right]\leq c\frac{\epsilon}{100}n$$

by choosing a large enough constant C (as a function of c) when setting t. Hence, by Markov's inequality, we get that

$$\sum_{u\in V:\beta(u;\sigma)>\tau}\mathbb{1}\{\hat{\beta}_S(u;\sigma)\leq 0\}\leq c\frac{\epsilon}{30}n$$

with probability 2/3. Therefore, with probability 2/3 we have

$$|E(Q,V\setminus Q)|\geq \text{OPT}(G)-c\frac{\epsilon}{15}n^2-2\epsilon|E|\,.$$

Finally, we use the fact that the graph G is dense: since $\text{OPT}(G)\geq |E|/2$ and $|E|\geq cn^2$, we get that the error is bounded by $\epsilon/7\cdot\text{OPT}(G)+4\epsilon\cdot\text{OPT}(G)$, i.e.

$$|E(Q,V\setminus Q)|\geq(1-5\epsilon)\text{OPT}(G)\,.$$

2 Knapsack. *(30 pts)* We want to come up with an approximation algorithm for the Knapsack problem. In this problem we are given n items with prices $p_1,\ldots p_n$, weights $w_1,\ldots,w_n$ and a capacity bound W. We want to pack a subset of items $S\subseteq[n]$ that does not exceed the capacity bound and achieves a maximum profit. More formally, we would like to find $\max_{S\subseteq[n]}\sum_{i\in S}p_i$ subject to $\sum_{i\in S}w_i\leq W$. Design and analyze a $(1-\epsilon)$-approximation algorithm for the Knapsack problem that runs in time $n^{O(1/\epsilon)}$ for any positive constant $\epsilon>0$.

Hint: Guess the $\frac{1}{\epsilon}$ most profitable items in the optimal solution and than use a "bang-for-the-buck" greedy strategy.

Solution:

Algorithm Idea Let $\varepsilon>0$ be fixed and define $k=\lceil 1/\varepsilon\rceil$. The main idea is to *guess* the k most profitable items in the optimal solution, then fill the remaining capacity greedily according to profit density.

Algorithm Description

1. Initialize $k = \lceil 1/\varepsilon \rceil$.
2. For every subset $H \subseteq [n]$ with $|H| \leq k$:
   (a) If $\sum_{i \in H} w_i > W$, skip this subset.
   (b) Let the remaining capacity be $W_H = W - \sum_{i \in H} w_i$.
   (c) Let $R = \{i \in [n] \mid p_i \leq \min_{i \in H} p_i\}$ be the remaining items.
   (d) Sort R by non-increasing profit density p_i/w_i.
   (e) Greedily add items from R into the knapsack (in sorted order) until no more items fit into W_H. Denote the set of added items by $G(H)$.
   (f) Let the total profit for this guess be
   $$P(H) = \sum_{i \in H} p_i + \sum_{i \in G(H)} p_i.$$
3. Return the set $H \cup G(H)$ that achieves the maximum $P(H)$.

The algorithm runs in time $O(n^k \cdot \text{poly}(n)) = O(n^{1/\varepsilon})$, since we enumerate all subsets of size at most k.

Approximation Guarantee Let OPT denote an optimal solution and $P^\star = \sum_{i \in \text{OPT}} p_i$ its total profit. Let $H^* \subseteq \text{OPT}$ be the k items in OPT with the largest profits (break ties arbitrarily), and let $p_{\min}$ be the smallest profit among items in H^*.

Since H^* consists of the k most profitable items in OPT, we have

$$p_{\min} \leq \frac{P^\star}{k} \leq \varepsilon P^\star.$$

Now consider the iteration of the algorithm where $H = H^*$. We fill the remaining capacity greedily by profit density. It is well known that the greedy algorithm for fractional knapsack achieves the optimal fractional profit, and that rounding down (removing the last fractional item) loses at most the profit of one item. Thus, the greedy integral packing achieves profit at least

$$\text{Greedy}(H^*) \geq (\text{OPT on remaining items}) - p^{\text{rem}}_{\max},$$

where $p^{\text{rem}}_{\max}$ is the largest profit among the remaining items.

Since all remaining items have profit at most $p_{\min}$, we have $p^{\text{rem}}_{\max} \leq p_{\min}$. Therefore, the total profit for this guess satisfies

$$P(H^*) = \sum_{i \in H^*} p_i + \text{Greedy}(H^*) \geq P^\star - p_{\min} \geq (1 - \varepsilon) P^\star.$$

Hence, the algorithm achieves a $(1 - \varepsilon)$-approximation.

3 Bin Packing. *(36 pts)* In this problem we want to study the bin packing problem. Here, we are given items $U = [n]$ of sizes $\{a_1, \dots a_n\}$ and a capacity bound $W \in \mathbb{N}$. We want to place items in bins of capacity B such that each item is contained in a bin. Moreover, we want to use the minimal number of bins to do so. A solution to this problem corresponds to a partition $B_1 \cup \cdots \cup B_m = U$ of the items such each B_i respects the capacity constraint, $\sum_{j \in b_i} a_j \leq B$ for each $i \in [m]$. You can assume that each item fits in a bin, $a_i \leq W$ for all $i \in U$. This problem is NP-complete so we are interested in an efficient approximation algorithm. An important relaxation for the problem is the *configuration linear program.* A configuration C is a subset of items that respects the capacity bound W, i.e. $\sum_{i \in C} a_i \leq W$. In other words, C is a configuration of a bin we can use for the partition. Let $\mathcal{C} = \{C \subseteq U \mid \sum_{i \in C} a_i \leq W\}$ be the set of all possible configurations. The configuration LP will have one variable x_C for each configuration $C \in \mathcal{C}$ indicating whether we use the configuration C in the partition.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{C \in \mathcal{C}} x_C & & \quad (1) \\
\text{subject to} \quad & \sum_{C \in \mathcal{C}:\, i \in C} x_C \geq 1 & \forall i \in U & \quad (2) \\
& x_C \geq 0 & \forall C \in \mathcal{C}. & \quad (3)
\end{aligned}
$$

Constraint (2) ensures each item a is placed in at least one (possibly fractional) bin. This relaxation is well studied and can be rounded to a solution of cost at most OPT + log(OPT). The standard methods for solving LPs run in exponential time in this case since the LP has exponentially many variables. Therefore, your task is to implement the Hedge strategy to obtain an algorithm that solves the configuration LP approximately in polynomial time. Here, the solution x can be described as a list of non-zero coordinates.

3a *(12 pts)* Show that the oracle for the reduced problem in your Hedge strategy can be solved up to a $(1 - \epsilon)$ multiplicative error in time $n^{O(1/\epsilon)}$.

Hint: Make use of the algorithm designed in Problem 2.

3b *(12 pts)* Prove that the solution you compute is a $(1 + \epsilon)$-approximation to the configuration LP. In other words, you find a solution x such that,

$$\sum_{C \in \mathcal{C}} x_C \leq (1 + \epsilon) \sum_{C \in \mathcal{C}} x^\star_C,$$

where $x^\star$ is an optimal solution to the configuration LP.

3c *(12 pts)* Bound the parameter ρ, i.e. bound the absolute value of the cost vectors in your hedge strategy, in order to achieve the overall runtime bound of $n^{O(1/\epsilon)}$. For example, one can achieve a bound of $\rho \leq n$.

Solution: The hedge algorithm for the configuration linear program works as follows.
Initialization: Set $w_i(1) = 1$ for all constraints $i \in U$.
For $t = 1, 2, \dots, T$:

1. Compute the probability distribution over constraints:

$$p_i^{(t)} = \frac{w_i(t)}{\sum_{j=1}^m w_i(t)} \quad \text{for all } i \in U.$$

2. Query the oracle using the convex combination of constraints given by $p^{(t)}$, and obtain a primal solution $x^{(t)}$.

3. Compute the constraint cost for each $a \in U$:

$$m_i^{(t)} = \sum_{C \in \mathcal{C}\colon i \in C} x_C^{(t)} - 1.$$

   (Note: $m_i^{(t)} > 0$ if constraint i is satisfied, $m_i^{(t)} < 0$ if it is violated.)

4. Update the weights:

$$w_i(t+1) = w_i(t)\, e^{-\varepsilon\, m_i^{(t)}} \quad \text{for all } i \in U.$$

Output: The averaged primal solution $\bar{x} = \frac{1}{T}\sum_{t=1}^T x^{(t)}$ scaled up by $\frac{1}{1-\epsilon}$.

3a We need to show how to implement the oracle queried in the second step. The reduced constraint reads as,

$$1 \leq \sum_{i \in U} p_i \sum_{C \in \mathcal{C}\colon i \in C} x_C^{(t)} = \sum_{C \in \mathcal{C}} \left(\sum_{i \in C} p_i \right) x_C.$$

Minimizing $\sum_{C \in \mathcal{C}} x_C$ subject to this constraint is equivalent to finding the configuration $D \in C$ that maximizes $p(D) = \sum_{i \in D} p_i$. Indeed,

$$1 \leq \sum_{C \in \mathcal{C}} p(C)\; x_C \leq p(D) \cdot \sum_{C \in \mathcal{C}} x_C \quad \Leftrightarrow \quad \sum_{C \in \mathcal{C}} x_C \geq \frac{1}{p(D)}.$$

The bound is achieved setting $x_D = \frac{1}{p(D)}$. Computing the configuration D is the Knapsack problem. We want to find a set $D \subseteq U$ that satisfies a capacity constraint, $\sum_{i \in D} a_i \leq W$ and maximizes the profit $\sum_{i \in D} p_i$. We know how to solve this problem up to a $(1-\epsilon)$ factor. Let O be the optimal solution to this problem. We can find a set D such that $\sum_{i \in D} p_i \geq (1-\epsilon)\sum_{i \in O} p_i$. Thus, returning $x_D = \frac{1}{p(D)}$ and all other coordinates 0 is a is a $(1+2\epsilon)$-approximate solution to the reduced problem,

$$x_D = \frac{1}{p(D)} \leq \frac{1}{(1-\epsilon)p(O)} \leq (1+2\epsilon)\frac{1}{p(O)} \leq (1+2\epsilon) OPT_{LP}.$$

3b By the properties of the oracle we have that $\sum_{C \in \mathcal{C}} x_C^{(t)} \leq (1+2\epsilon) OPT_{LP}$ for all $t \leq T$. This also holds for $\bar{x}$ since it is the average of all $x^{(t)}, t \leq T$. Thus, $\frac{1}{1-2\epsilon}\sum_{C \in \mathcal{C}} \bar{x}_C \leq \frac{(1+2\epsilon)}{1-2\epsilon} OPT_{LP} \leq (1+5\epsilon) OPT_{LP}$. $\frac{\bar{x}}{1-\epsilon}$ is feasible by the guarantees from Lecture 9 for $T = (4\rho^2 \ln n)/\epsilon^2$ where ρ is an upper bound on the width of our cost vectors.

3c Since we choose $T = (4\rho^2 \ln n)/\epsilon^2$ we need to control ρ, the upper bound on the cost $m_i^{(t)}$. Note that p is a probability distribution of items $i \in U$. Therefore, there is an item $i \in U$ such that $p_i \geq \frac{1}{n}$. Moreover, the configuration $C = \{i\}$ has weight $p(C) \geq \frac{1}{n}$. Thus, $p(O) \geq \frac{1}{n}$. For the output of the oracle, $x_D = \frac{1}{p(D)} \leq (1+2\epsilon)\frac{1}{p(O)} \leq (1+2\epsilon)n$. In particular,

$$-1 \leq m_i = \sum_{C \in \mathcal{C}\colon i \in C} x_C - 1 \leq x_D = O(n).$$

Thus, $\rho = O(n)$ and $T = O(n^2 \ln(n))$. Since the oracle can be implemented in time $n^{1/\epsilon}$ we get the runtime from the problem statement.