6.891 Approximation Algorithms March 17, 2000

Problem Set 2 Solutions

1. Consider the following problem:

MAX k-CUT

- **Input:**
  - An undirected graph $G = (V, E)$
  - Weights $w_{ij} \geq 0$ for all $(i, j) \in E$
- **Goal:** Partition the vertex set V into k parts $V_1, \ldots, V_k$ so as to maximize the weight of all edges whose endpoints are in different parts (i.e. $\max_{(i,j) \in E : i \in V_a, j \in V_b, a \neq b} w_{ij}$).

Give a randomized $\frac{k-1}{k}$-approximation algorithm for the MAX k-CUT problem.

Solution: Consider the following algorithm:

DumbRandom

$V_1 \leftarrow \emptyset$; $V_2 \leftarrow \emptyset$; $\cdots$; $V_k \leftarrow \emptyset$

For $i \leftarrow 1$ to n

Pick x uniformly from $[0, 1]$

If $x \in [j/k, (j+1)/k)$

$V_{j+1} \leftarrow V_{j+1} \cup \{i\}$

Theorem 1 *DumbRandom3 is a $\frac{k-1}{k}$-approximation algorithm.*

Proof: Certainly the algorithm produces a partition into k parts and runs in randomized polynomial time. Let the random variable

$$X_{ij} = \begin{cases} 1 & \text{if for some } a, b, i \in V_a, j \in V_b, a \neq b \\ 0 & \text{otherwise} \end{cases}$$

and let the random variable

$$W = \sum_{(i,j) \in E} w_{ij} X_{ij}.$$

Then

$$\begin{aligned}
E[W] &= \sum_{(i,j)\in E} w_{ij}\Pr[i\in V_a, j\in V_b, \text{for some } a\neq b]\\
&= \sum_{(i,j)\in E} w_{ij}(1-\Pr[i\in V_a, j\in V_a \text{ for some } a])\\
&= \sum_{(i,j)\in E} w_{ij}\left(1-\sum_{a=1}^{k}\Pr[i\in V_a, j\in V_a]\right)\\
&= \sum_{(i,j)\in E} w_{ij}\left(1-\sum_{a=1}^{k}\frac{1}{k^2}\right)\\
&= \sum_{(i,j)\in E} w_{ij}\left(1-\frac{1}{k}\right)\\
&= \frac{k-1}{k}\sum_{(i,j)\in E} w_{ij}\\
&\geq \frac{k-1}{k}\cdot OPT.
\end{aligned}$$

$\square$

2. Consider the linear programming relaxation for the MAX SAT problem that we saw in class:

$$\begin{array}{lll}
\text{Max} & \displaystyle\sum_{j=1}^{m} w_j z_j & \\
& \displaystyle\sum_{i\in P_j} y_i + \sum_{i\in N_j}(1-y_i)\geq z_j & \displaystyle\forall C_j = \bigvee_{i\in P_j} x_i \bigvee_{i\in N_j}\bar{x}_i\\
& 0\leq y_i\leq 1 & 1\leq i\leq n\\
& 0\leq z_j\leq 1 & 1\leq j\leq m.
\end{array}$$

Prove that using randomized rounding with the linear function $f(y_i)=\frac{1}{2}y_i+\frac{1}{4}$ also gives a $\frac{3}{4}$-approximation algorithm for MAX SAT.

Solution: As usual, we first need to analyze the probability that a given clause

C_j is satisfied.

$$
\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i\in P_j}(1-f(y_i^*))\prod_{i\in N_j} f(y_i^*) \\
&= \prod_{i\in P_j}\left(\frac{3}{4}-\frac{1}{2}y_i^*\right)\prod_{i\in N_j}\left(\frac{1}{2}y_i^*+\frac{1}{4}\right) \\
&\le \left[\frac{3}{4}-\frac{1}{2l_j}\left(\sum_{i\in P_j} y_i^* + \sum_{i\in N_j}(1-y_i^*)\right)\right]^{l_j} \qquad (1)\\
&\le \left(\frac{3}{4}-\frac{1}{2l_j}z_j^*\right)^{l_j} \qquad (2)
\end{aligned}
$$

where (1) follows from the arithmetic/geometric mean inequality and (2) follows from the linear programming constraint for the clause. Then

$$
\begin{aligned}
\Pr[C_j \text{ satisfied}] &\ge 1-\left(\frac{3}{4}-\frac{1}{2l_j}z_j^*\right)^{l_j} \qquad (3)\\
&\ge \frac{3}{4}z_j^* \qquad (4)
\end{aligned}
$$

where (4) follows since the function $g(z) = 1-\left(\frac{3}{4}-\frac{1}{2l_j}z\right)^k$ is concave on $[0,1]$, $g(0)=0$, and $g(1)\ge \frac{3}{4}$ for all integer $l_j \ge 1$. Proving the last claim requires a little work: let $h(k) = 1-\left(\frac{3}{4}-\frac{1}{2k}\right)^k$. Clearly $h(1)=h(2)=\frac{3}{4}$, and $h(3)\approx .801$. For $k\ge 3$, observe that

$$h'(k) = -\left(\frac{3}{4}-\frac{1}{2k}\right)^k\left(\frac{1}{1.5k-1}+\ln\left(\frac{3}{4}-\frac{1}{2k}\right)\right).$$

We want to show that $h'(k)>0$ for $k\ge 3$, which will establish our claim. To see this note that $\left(\frac{3}{4}-\frac{1}{2k}\right)^k \ge 0$ for $k\ge 1$ and $\ln\left(\frac{3}{4}-\frac{1}{2k}\right)\le \ln\frac{3}{4}\approx -.287$, while for $k\ge 3$, $\frac{1}{1.5k-1}\le .286$. This proves the claim.

Then

$$
\begin{aligned}
E[W] &= \sum_j w_j \Pr[\text{Clause } j \text{ satisfied}] \\
&\ge \frac{3}{4}\sum_j w_j z_j^* \\
&\ge \frac{3}{4}\cdot OPT,
\end{aligned}
$$

and the algorithm is a $\frac{3}{4}$-approximation algorithm.

3. The *minimum-capacity multicommodity flow problem* is the following:

- **Input** An undirected graph $G = (V, E)$, and k pairs of terminals s_i, t_i.
- **Goal** A multicommodity flow in which one unit of flow is shipped from each s_i to t_i so as to minimize the total flow crossing any edge.

Assume that we can find an optimal vertex solution of the following linear program in polynomial time:

$$\begin{array}{rll} \text{Min} & W & \\ \text{subject to:} & \displaystyle\sum_{P \in \mathcal{P}_i} x_P \geq 1 & \forall i \\ & \displaystyle\sum_{P : e \in P} x_P \leq W & \forall e \in E \\ & x_P \geq 0, & \end{array}$$

where $\mathcal{P}_i$ is the set of paths P from s_i to t_i.

(a) (Almost trivial) Argue that the linear program above is a relaxation of the minimum-capacity multicommodity flow problem.

(b) Let W^* be the optimal value of the linear program. Show that if $W^* \geq \ln n$, then we can obtain in randomized polynomial time a solution of value no more than $W^* + c\sqrt{W^* \ln n}$ with high probability.

Solution:

(a) Suppose we have an optimal solution to the minimum-capacity multicommodity flow problem. Since a feasible solution to the problem sends, for each i, a unit of flow from s_i to t_i, it does so along some path $P \in \mathcal{P}_i$. Set $x_P = 1$ for each such P for each i, and all other $x_P = 0$. This setting of the variables will satisfy the first set of constraints of the LP. If we then set W to be the maximum over all edges of the total flow crossing the edge, the setting of the variables will also satisfy the second set of constraints, and the value of the objective function will be equal to the optimal value. Thus the LP is a relaxation of the problem.

(b) Let (x^*, W^*) be an optimal solution to the LP. Let $c_i = \sum_{P \in \mathcal{P}_i} x^*_P$. Consider the randomized algorithm which for each i, randomly chooses exactly one path P from $\mathcal{P}_i$ with probability x^*_P / c_i and sends a unit flow from s_i to t_i along the path. This certainly gives a solution to the problem. Also observe that at an extreme point to the solution, at most $k + m$ of the x^*_P will be non-zero, so we can make this random choice in polynomial time.

Define random variables X^i_e, where $X^i_e = 1$ if the chosen s_i-t_i path uses edge e, and $X^i_e = 0$ otherwise. For a given $e \in E$, consider $Y_e = \sum_i X^i_e$.

We want to bound $\max_{e \in E} Y_e$, since this gives the total flow crossing any edge. Certainly

$$E[Y_e] = \sum_i \sum_{P \in \mathcal{P}_i : e \in P} \frac{x_P^*}{c_i} \leq \sum_{P : e \in P} x_P^* \leq W^*,$$

since $c_i \geq 1$ for all i and using the LP constraint. Using Chernoff bounds, which we can do since the X_e^i are independent for fixed e, and setting $\delta_e = \sqrt{\frac{3c \ln n}{W^*}}$, we get that the probability that Y_e exceeds $(1+\delta_e)W^*$ is no more than

$$e^{-W^* \delta_e^2 / 3} = e^{-c \ln n} = \frac{1}{n^c}.$$

Note that by assuming $W^* \geq 3c \ln n$ we have $\delta_e \leq 1$. Also note that $(1+\delta_e)W^* = W^* + \sqrt{3cW^* \ln n}$. Then since there can be at most n^2 edges, the probability that $\max Y_e$ exceeds $W^* + \sqrt{3cW^* \ln n}$ is at most $n^2 \cdot \frac{1}{n^c} = \frac{1}{n^{c-2}}$.

4. Using randomized rounding and first fit, give a randomized polynomial-time algorithm for bin packing that uses $\rho \cdot OPT(I) + k$ bins for some $\rho < 2$ and some small constant k.

   Solution: We use a slight variation of the linear programming relaxation for the bin packing problem as it was used for the Karmarkar-Karp algorithm. Rather than grouping pieces into various sizes, and having configurations that say how many of each piece to use, we consider each piece individually, and configurations indicate whether a piece is included or not (note that this is the situation if each piece has a distinct size). The linear program is then:

$$\begin{aligned} \text{Min} \quad & \sum_{j=1}^{N} x_j \\ \text{subject to} \quad & \\ & \sum_{j=1}^{N} a_{ij} x_j \geq 1 \qquad \text{for all } i = 1, \ldots, m \\ & x_j \geq 0, \end{aligned}$$

   where m is the number of pieces, and each variable x_j indicates the number of bins that should be packed in configuration A_j (see Lecture 2 notes for a further explanation). Note that since each a_{ij} is now either 0 or 1 (since either piece i is in configuration A_j or not), in any optimal solution x^*, $0 \leq x_j^* \leq 1$ for each j. This follows since if $x_j^* > 1$, we could reduce it to 1 and all inequalities would still remain feasible.

   Suppose we obtain a near-optimal solution x^* in polynomial time (via the theorem of Karmarkar and Karp given in Lecture 2). We now run the following algorithm: for each non-zero value x_j^* (of which there are at most m), we pack a bin with configuration A_j with probability x_j^* for each configuration A_j independently. This might lead to the same piece being packed multiple times, but that is not a problem. Also it is possible that some pieces are not packed: we pack these remaining pieces into bins using first fit.

We now analyze this algorithm. The expected number of bins used by the randomized procedure is

$$\sum_{j=1}^{N} x_j^* \leq OPT(I) + 1,$$

since by the Karmarkar-Karp theorem, the LP solution x^* is within 1 of the optimal LP solution, and the value of the LP is a lower bound on $OPT(I)$. We now need to analyze the number of bins needed to pack the leftover pieces. Consider the probability that piece i is not packed by the randomized procedure:

$$\begin{aligned}
\Pr[\text{piece } i \text{ not packed}] &= \prod_{j:a_{ij}=1} (1 - x_j^*) \\
&\leq \prod_{j:a_{ij}=1} e^{-x_j^*} \qquad (5) \\
&= e^{-\sum_{j:a_{ij}=1} x_j^*} \\
&= e^{-\sum_{j=1}^{N} a_{ij} x_j^*} \\
&\leq e^{-1}, \qquad (6)
\end{aligned}$$

where (5) follows since $1 - x \leq e^{-x}$ for $x \in [0,1]$, and (6) follows by the linear programming constraint for piece i. Recall that first fit uses a number of bins that can be bounded by twice the total piece size packed plus one bin. Let s_i denote the size of piece i. Thus the expected number of bins used by first fit to pack the pieces remaining after the randomized procedure is

$$\begin{aligned}
2\sum_{i=1}^{m} \Pr[\text{piece } i \text{ not packed by randomized rounding}] s_i + 1 &\leq \frac{2}{e}\sum_{i=1}^{m} s_i + 1 \\
&\leq \frac{2}{e} OPT(I) + 1.
\end{aligned}$$

Thus the expected number of bins used by the algorithm is no more than

$$\left(1 + \frac{2}{e}\right) OPT(I) + 2,$$

where $1 + \frac{2}{e} \approx 1.736$.