CSE525: Randomized Algorithms and Probabilistic Analysis April 30, 2013

Lecture 8

Lecturer: Anna Karlin *Scribe: Jonathan Shi, Dimitrios Gklezakos*

1 Online bipartite matching

Recall from last time we introduced the online bipartite matching problem and gave an algorithm to solve a fractional version of the problem with competitive ratio $1 - 1/e$. In the online bipartite matching problem, we are given a bipartite graph $G = (L, R, E)$, where the vertices in R are revealed to us one at a time along with the edges from that vertex to L. Once each vertex in R is revealed, we must immediately select an edge to that vertex to be a part of a matching M or permanently discard all the revealed edges, with the goal of maximizing the size of M. The fractional version of the problem is obtained by taking the linear relaxation of the integer program that corresponds to this problem.

1.1 RANKING algorithm

The RANKING algorithm can be framed in two equivalent ways. In the interpretation that makes for convenient analysis, the algorithm computes the solution to the fractional problem by solving the linear relaxation in Figure 1, and then uses those fractional values as the probability that each edge will be put into our matching. Equivalently, the algorithm imposes a random order Π on the vertices in L and then when v arrives from R, v is matched with its unmatched neighbor that is lowest in Π.

$$
\begin{aligned}
\max \sum_{(i,j)\in E} x_{ij} & \\
\sum_{j\in N(i)} x_{ij} \le 1 \qquad & \forall i \in L \\
\sum_{i\in N(j)} x_{ij} \le 1 \qquad & \forall j \in R \\
x_{ij} \ge 0 \qquad & \forall (i,j) \in E
\end{aligned}
$$

Figure 1: Linear relaxation for online bipartite matching.

$$
\begin{aligned}
\min \sum_{i\in L} \alpha_i + \sum_{j\in R} \beta_j & \\
\alpha_i + \beta_j \ge 1 \qquad & \forall (i,j) \in E \\
\alpha_i \ge 0 \qquad & \forall i \in L \\
\beta_j \ge 0 \qquad & \forall j \in R
\end{aligned}
$$

Figure 2: Dual of the previous linear program.

1.2 Analysis of algorithm

Our analysis of this algorithm will be very similar to the argument for the fractional case. Namely, we will construct a dual solution for each possible outcome of the randomized primal, and show that the primal P and the dual D satisfy the properties:

1. $P \ge cD$ for some c, where P and D are understood to stand in for the values of their objective functions.
2. D is feasible in expectation. That is, for each constraining inequality, in the dual problem, the expected values of the dual variables satisfy that inequality.

Theorem 1. *If $P \ge cD$ and D is feasible in expectation, then $E[P] \ge c \cdot \mathrm{OPT}$.*

Proof If $P \geq cD$ always, then $E[P] \geq c \cdot E[D]$. Also D is feasible in expectation, so that linearity of expectation allows us to conclude that $E[D] \geq \text{OPT}$ since the expectations of the dual variables form a feasible solution. Hence $E[P] \geq c \cdot E[D] \geq c \cdot \text{OPT}$. ■

We construct the dual solution by first randomly constructing the primal solution and then setting dual variables $\hat{\alpha}_i$ and $\hat{\beta}_j$ as follows, using some fixed increasing function $g : [0,1] \to [0,1]$ with $g(1) = 1$:

- Set $\hat{\alpha}_i = \frac{1}{c} g(Y_i)$ and $\hat{\beta}_j = \frac{1}{c}(1 - g(Y_i))$, if $x_{ij} = 1$ (equivalently, if i is matched to j in the primal).
- Set $\hat{\alpha}_i = 0$, if $x_{ij} = 0$ for all j (equivalently, if i is unmatched).
- Set $\hat{\beta}_j = 0$, if $x_{ij} = 0$ for all i (equivalently, if j is unmatched).

This satisfies the property that $P \geq cD$ since, for every (i,j) that is matched, x_{ij} (and hence P) increases by 1, while $\hat{\alpha}_i + \hat{\beta}_j$ (and hence D) increases by $1/c$.

To show that D is also feasible in expectation, we need it to satisfy the only nontrivial constraint, given by:

$$(\forall (i,j) \in E) \quad E[\hat{\alpha}_i + \hat{\beta}_j] \geq 1.$$

So consider a fixed edge (i,j), and fix a value for Y_i. Let G_{-i} denote the problem instance after the vertex i and all incident edges have been removed, and consider running the algorithm on G_{-i}. Then the vertex j will either be matched to some $i' \in L$ or it won't be matched. Let $y^{-i} = Y_{i'}$ when j is matched, or $y^{-i} = 1$ when j is not matched. Let also β_j^{-i} be the value of β_j when the algorithm is run on G_{-i}; so that $\beta_j^{-i} = \frac{1}{c}(1 - g(y^{-i}))$. Finally, let $U(G, j')$ denote the set of unmatched elements of L in G when $j' \in R$ is up to be matched.

Then we can show the following:

Lemma 1. *In parallel executions of the algorithm on G and G_{-i}, for each vertex $j_k \in R$, $U(G, j_k) = U(G_{-i}, j_k) \cup \{i'\}$ for some $i' \in L$. Also, either the execution on G matches j_k to a vertex of equal or lesser Y-value than the execution on G_{-i}, or the execution on G_{-i} doesn't match j_k at all.*

Proof In the base case, j_1 arrives before any of L has been matched and we have $U(G, j_1) = U(G_{-i}, j_1) \cup \{i\}$.

In the inductive case, some j_k arrives with $U(G, j_k) = U(G_{-i}, j_k) \cup \{i'\}$. Then the execution on G could match j_k either to i' or to some $i'' \in U(G_{-i}, j_k)$. If the execution on G matched j_k to some $i'' \neq i'$, then it must be that $Y_{i''}$ is the lowest Y-value in $U(G_{-i}, j_k) \cap N(k)$, so that the execution on G_{-i} also matched j_k to i'', and hence $U(G, j_{k+1}) = U(G_{-i}, j_{k+1}) \cup \{i'\}$. On the other hand, if the execution on G matched j_k to i', then i' must've had a lower $Y_{i'}$ value than anything in $U(G_{-i}, j_k)$, so that the execution on G_{-i} matched j_k to some i'' with $Y_{i''} > Y_{i'}$, and $U(G, j_{k+1}) = U(G_{-i}, j_{k+1}) \cup \{i''\}$. ■

Lemma 2. *If $Y_i < y^{-i}$, then the vertex i will be matched in the original problem.*

Proof Consider what happens when j arrives to be matched in the parallel executions on G and G_{-i}. Assume that i is not matched yet in G when this happens. This means that the executions on G and G_{-i} have been identical up to this point. Let i' be the vertex that j is matched to in G_{-i}. Then $y^{-i} = Y_{i'}$. If $Y_i < y^{-i}$, then $Y_i < Y_{i'}$ so that j gets matched to i instead in the execution on G. ■

Lemma 3. $\hat{\beta}_j \geq \beta_j^{-i}$.

Proof Refer to Lemma 1 to conclude that when j is being matched in the G and G_{-i} instances, either the G_{-i} execution does not match j at all, so that $y^{-i} = 1$ and $\hat{\beta}_j \geq 0 = \frac{1}{c}(1 - g(y^{-i})) = \beta_j^{-i}$, or the G_{-i} execution matches j to an i' and the G execution matches j to an i'' such that $Y_{i''} \leq Y_{i'}$ and:

$$\hat{\beta}_j = \frac{1 - g(Y_{i''})}{c} \geq \frac{1 - g(Y_{i'})}{c} = \beta_j^{-i},$$

where we used the fact that g is an increasing function. ■

These lemmas show that D is feasible in expectation. First, $\hat{\alpha}_i = \frac{1}{c}g(Y_i)$ if i is matched, and by Lemma 2, i is matched whenever $Y_i < y^{-i}$, so that $E[\hat{\alpha}_i] \geq \int_0^{y^{-i}} \frac{1}{c}g(y)\,\mathrm{d}y$. Second, by Lemma 3 we have $\hat{\beta}_j \geq \beta_j^{-i} = \frac{1}{c}(1 - g(y^{-i}))$. If we now set $c = 1 - \frac{1}{e}$ and $g(y) = e^{y-1}$, meeting all the criteria for g and satisfying $\int g(y)\,\mathrm{d}y = g(y) + C$, we find that:

$$E_{Y_i}[\hat{\alpha}_i + \hat{\beta}_j] \geq \frac{g(y^{-i}) + C - g(0) - C + 1 - g(y^{-i})}{c} = \frac{1 - g(0)}{c} = \frac{1 - \frac{1}{e}}{1 - \frac{1}{e}} = 1,$$

for all possible choices of i, j, and Y_i.

1.3 Significance

The analysis of the RANKING algorithm presented here was first described by Devanur, Jain, and Kleinberg in 2013. It significance stems from the fact that, not only is it simpler than any previous proof of correctness for this algorithm, it also introduced the novel approach of bounding the approximation with a dual solution is only feasible in expectation. This allowed us to bound the integer randomized rounding solution using an dual linear program, unifying the fractional and integral analyses of this integer programming problem in a completely new way.

A related problem which does not yet have a similar analysis is as follows: A list of integers $\{B_i\}$ are taken to represent the daily budgets of some advertisers, where advertiser i bids b_{ij} to show their advertisement on a particular search j. The problem is then to determine which bids to take in an online algorithm in order to maximize our income for the day. This problem currently has a 1-(1/e)-approximation achieved in the fractional setting, but nothing better than a 1/2-approximation for the integral one.

2 Semi-definite programming

Semi-definite programming is linear programming where variables are entries in a symmetric psd matrix. A psd matrix is the analogue of a positive number for matrices. The following statements are equivalent:

1. X is a psd matrix (denoted by $X \succeq 0$)
2. $\forall y \in \mathcal{R}^n, y^T X y \geq 0$
3. X has non-negative eigenvalues
4. $\exists V \ : \ V^T V = X$. This is called the Cholesky decomposition of a matrix and is similar to taking its root.
5. $X = \sum_{i=1}^n \lambda_i w_i w_i^T$, where w's are orthonormal vectors and λ's are non-negative.

SDPs are a special case of convex programming. We can solve SDPs to arbitrary additive error ϵ in time $poly(size(input), log(\frac{1}{\epsilon}))$. Formulation (for maximization problems):

$$maximize \sum_{i,j} c_{ij} x_{ij}$$

$$s.t.$$

$$\forall k, \sum_{i,j} a_{ijk} x_{ij} = b_k$$

$$\forall i, j, x_{ij} = x_{ji}$$

$$X \succeq 0$$

or vector formulation:

$$maximize \sum_{i,j} c_{ij}(v_i \cdot v_j)$$

$$s.t.$$
$$\forall k, \sum_{i,j} a_{ijk}(v_i \cdot v_j) = b_k$$
$$\forall i, v_i \in \mathcal{R}^n$$

2.1 MAXCUT revisited (Goemmans-Williamson)

Let $z_{ij} = 1$ if edge (i,j) is cut. Let $x_i \in \{0,1\}$ denote the side of i in the cut. Then $z_{ij} = 1$ only if $x_i \neq x_j$. Consider the following integer program:

$$maximize \sum_{(i,j)\in E} w_{ij} z_{ij}$$
$$s.t.$$
$$\forall (i,j) \in E, z_{ij} \leq x_i + x_j$$
$$\forall (i,j) \in E, z_{ij} \leq 2 - (x_i + x_j)$$
$$\forall (i,j) \in E, z_{ij} \in \{0,1\}$$
$$\forall i \in V, x_i \in \{0,1\}$$

For every graph, the fractional solution puts $\frac{1}{2}$ on everything! We can formulate MAXCUT via quadratic programming:

$$max(\frac{1}{2} \sum_{(i,j)\in E} w_{ij}[1 - y_i y_j])$$

subject to:

$$y_i \cdot y_j = 1$$
$$y_i \in \{0,1\}$$

Replace y_i by a vector in $\mathcal{R}^n$. Then we get an SDP, where the solution will give vectors on the sphere. This is a relaxation.