Online Optimization Notes

Stephen J. Barr
ISOM
Foster School of Business
University of Washington
email: stevejb@uw.edu

January 26, 2013

This Version: January 26, 2013
Initial Version: January 7, 2012

1 Online Bipartite Matching

1.1 Bipartite Matching

INPUT: Bipartite Graph $G = (V = (L, R), E)$ and w.l.o.g. $|L| = |R| = n$. $E \subseteq L \times R$.

OUTPUT: A matching $M \subseteq E$ such that no vertex has > 1 incident edge in the matching M.

OBJECTIVE: Find a matching of maximum size.

Terminology:

- *i is **matched** to j* to denote that $(i, j) \in M$.
- A matching is ***perfect*** if every vertex is part of the matching.

In *online bipartite matching*, the structure of G, specifically the vertices in R, is revealed over time.

1.2 Online BP Matching

DEFINITION:

1. Vertices in R "arrive" 1-by-1
2. A vertex is matched immediately upon arrival
3. Matches, once made, cannot be reversed

An "instance" I of the O.B.M. consists of G and an ordering of arrival of vertices in R.
Consider an algorithm `ALG`.

- $\texttt{ALG}(I) \triangleq$ number of matches picked by `ALG` on instance I.

- $\texttt{OPT}(I) \triangleq$ size of the max. matching in G.

$\texttt{ALG}$ has a competitive ratio (CR) of α if $\forall I \texttt{ALG}(I) \geq \alpha * \texttt{OPT}(I)$. We say: $\texttt{ALG}$ is α-competitive. Also, we see that, no matter what algorithm we use, we will get an upper bound $a \leq \frac{1}{2}$.

Algorithm 1. *Greedy Algorithm Match (to an arbitrary neighbor) if possible. There is no "intelligence" as to how to break the ties when there are multiple matches.*

- *# of vertices matched by* $\texttt{Greedy} - \texttt{ALG} \geq$ *# of vertices matched by* $\texttt{OPT}$
- *# of edges matched by* $\texttt{Greedy} - \texttt{ALG} = \frac{1}{2}$ *vertices matched by* $\texttt{ALG}$.
- $\texttt{Greedy} - \texttt{ALG}(I) \geq \frac{1}{2}\texttt{OPT}(I)$

Now, consider a random algorithm $\texttt{Rand} - \texttt{ALG}$ that picks matchings by flipping a coin. In the context of Fig. 2, we see that with probablity (w.p.) $\frac{1}{2}$, $\texttt{Rand} - \texttt{ALG}(I) = 2$ and w.p. $\frac{1}{2}$, $\texttt{Rand} - \texttt{ALG}(I) = 1$. Thus, $E[\texttt{Rand} - \texttt{ALG}(I)] = 3/2$. Thus, for random algorithms we redefine the competitive ratio as

$$E[\texttt{ALG}(I)] \geq \alpha * \texttt{OPT}(I) \tag{1}$$

For randomized algorithms, the upper bound on the competitive ratio is $1 - \frac{1}{e}$.

Consider Fig. 3, where there is a perfect matching $1 - a$, $2 - b$, $3 - c$, etc. Divide both left and right sides in half, and draw edges from bottom half of L to top half of R. Then, in a purely random matching, most of L_{lower} matches to R_{upper}, and R_{lower} is then impossible to match. So, we need to be smarter than this.

1.3 Fractional Bipartite Matching

Algorithm 2. *Fractional BM* $\forall$ *edge* (i,j) $x_{ij} \in [0,1]$ *amount of edge is picked up in the matching.* $\forall$ *vertex* i $\sum_{j \in i} X_{ij} \leq 1$.

OBJECTIVE:

$$\max_{(i,j)\in E} x_{ij} \tag{2}$$

We see that

$$X_{ij} = \begin{cases} 1 & (i,j) \in M \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = P_1[X_{ij} = 1]$$

$\forall i \sum_{j\tilde{i}} X_{ij} \leq 1 \to E[\sum_{j\tilde{i}} X_{ij}] \leq 1 \Rightarrow \sum_j x_{ij} \leq 1$.

$$\sum_{(i,j)\in E} x_{ij} = E[\sum_{(i,j)\in E} X_{ij}] = E[\texttt{ALG}(I)]$$

From this we see that randomized algorithms can be written as deterministic fractional algorithm (DFA). The upper bound in DFA $\Rightarrow$ upper bound in randomized algorithm.

1.3.1 Upper Bound on Fractional BM

When we get a vertex $j \in R$, we divide it equally among all of the incident $i \in L$ that are not completely matched. Doing this repeatedly, the vertices in L fill up and reach 1. For large n, we see that the first n we match fully and the rest we do not match at all. We see that the ith vertex got

$$\frac{1}{n} + \frac{1}{n-1} + ... + \frac{1}{i} = H_n - H_{i+1}$$

Setting the above $= 1$,

In the situation where we do not divide equally, we become strictly worse off. This proves the upper bound of $1 - \frac{1}{e}$. This means that no fractional algorithm can do better than this, and no randomized algorithm can do better than this

1.3.2 What about algorithms?

Water Level Algorithm. Objective: find a fractional matching. Consider the water level algorithm. The LHS are bins for water, and the RHS are sources of water. All bins have unit capacity. When you are filling bins, fill the lowest one. When you have multiple bins at the same lowest level, divide the water equally.

Terminology:

- X_{ij} = fraction of edge (i, j)
- $y_i = \sum_{j\tilde{i}} x_{ij} \approxeq$ level of water in bin i
- When j arrives, repeate:
  - $\forall$ vertices $i \in \arg\min_{i'}\{y_{i'}\}$ increment x_{ij} by dx
- Until either $\sum_{i\tilde{j}} x_{ij} = 1$ or $\arg\min_{i'}\{y_{i'}\} = 1$.

How do we analyse this algorithm? Consider the greedy algorithm, which matches to any arbitrary neighbor which is not yet matched. The algorithm always gets at least half optimal. Denote by α_i to be the total money (50 cents from a dollar) placed on i by `ALG`. $\beta_j \in L$ is the total money placed on $j \in R$ by `ALG`. Let $g(x) = e^{x-1}$.

- $g(0) = 1/e$
- $g(1) = 1$
- $\int_0^a g(x)dx = g(a) - (1/e)$

In a matching, consider a quantity dx which gets split between i and j. i gets $dx * g(y_i)$ (and α_i gets incremented by that much), and j gets $dx(1 - g(y_i))$ (and β_j) gets incremented by that much.

Considering a_i as a function,

$$\alpha_i = \int_0^{y_i^f} g(y_i)dy_i = g(y_i^f) - (1/e)$$

where y_i^f denotes a "final" value. If $y_i^f = 1$, then $\alpha_i = g(1) - 1/e = 1 - (1/e)$. OTOH, suppose $y_i^f < 1$, then $\sum_{i\tilde{j}} x_{ij} = 1$, meaning j was completely matched before i. It must also be the case that $\beta_j \geq 1 - g(y_i^f)$. All this means that

$$\alpha_i + \beta_j = g(y_i^f - (1/e) + 1 - g(y_i^f) = 1 - (1/e).$$

The question is, where did $g(\cdot)$ come from?

1.3.3 Derivation of $g(\cdot)$

Denote $\int g(x)dx = G(x)$. Then

$$\alpha_i = \int_0^{y_i^f} g(y_i)dy_i = g(y_i^f) - (1/e) = G(y_i^f) - G(0)$$

If y_i^f is 1, then $\alpha_i = 1 - (1/e) = G(1) - G(0) \geq \gamma$.

$$\alpha_i + \beta_j \geq \underbrace{G(y_i^f) - G(0)}_{\alpha_i} + \underbrace{1 - g(y_i^f)}_{\beta_i} \geq \gamma \forall y_i^f \in [0,1]$$

Differentiating the above and setting to 0,

$$g(y_i^f) - \frac{dg}{dy_i^f} = 0 \Rightarrow g(y_i^f) = \frac{dg}{dy_i^f} \forall y_i^f \in [0,1]$$

which we can solve as

$$\begin{aligned}
\int dy &= \int dg/g \\
y &= lng + c \\
e^y &= g * e^c \\
g &= e^{y-c} \\
G = e^{y-c}.1 - G(0) &\geq \gamma \\
1 - e^{-c} &\geq \gamma \\
G(1) - G(0) &\geq \gamma \\
g(1) &\leq 1 \\
e^{1-c} &\leq 1 \Rightarrow \\
c &\cdot 1 \\
c &= 1 \text{is the optimal}
\end{aligned}$$

The above differential equation and boundary condition are what derives the $g(\cdot)$ function.

1.3.4 Fractional Matching LP

$$\max \sum_{i,j \in E} x_{ij} \text{s.t.}$$
$$\forall i \in L \sum_{j\tilde{i}} x_{ij} \leq 1 \quad (\alpha_i)$$
$$\forall j \in R \sum_{i\tilde{j}} x_{ij} \leq 1 \quad (\beta_i)$$
$$\forall (i,j) \in E x_{ij} \geq 0$$

The dual of this is

$$\min \sum_i \alpha_i + \sum_j \beta_j$$
$$\forall (ij \in E) \alpha_i + \beta_j \geq 1$$
$$\alpha_i \beta_j \geq 0.$$

Weak duality says that feasible primal $\leq$ OPT $\leq$ feasible dual. We will show that feasible primal and feasible dual are within γ of each other. We can think of this as that each i makes an offer, and j picks the best offer.

1.3.5 $b-$matching (KP-2000)

This is an integral matching. But, every vertex $i \in L$ can be matched b times. Every $j \in R$ can be matched only once. They give a $1 - (1/e)$ competitive algorithm. The assignment is to derive a proof of $1 - (1/e)$ based on the proof in class. Can prove to analysis to KP 2000.

2 Adwords (Budgeted Allocation) Problem

- Given a set of advertisers
- Each $i \in L$ has a budget B_i
- Set of "queries", R
- $\forall j \in R$, $\forall i \in L$, there is a bid b_{ij}.
- Each query can be matched to ≤ 1 advertiser
- **Objective:** Maximize for each advertiser

$$\sum_j \min\{\sum_{j:j\rightarrow i} b_{ij}, B_i\}$$

In the adwords problem, we assume that $b_{ij}/B_i << 1$, meaning each individual bid is well under the total budget.

2.1 Fractional Budget Allocation

Assume fractional matching: x_{ij} is the fraction of j that is matched to i. Then the matching constraints become $\sum_i x_{ij} \leq 1$. The new objective is:

$$\max \sum_{i,j} b_{ij} x_{ij}$$

Note that is the bids get smaller and smaller, the Budgeted Allocation problem becomes the fractional budget problem.

2.1.1 Fractional BA LP and Dual

LP:

$$\begin{aligned}
\max \sum_{i,j} b_{ij} x_{ij} & \qquad \text{s.t.} \\
\forall j \sum_i x_{ij} \leq 1 & \qquad (\beta_j) \\
\forall i \sum_j x_{ij} \leq 1 & \qquad (\alpha_i) \\
x_{ij} \geq 0 &
\end{aligned}$$

Dual:

$$\begin{aligned}
&\min \sum_i \alpha_i B_i + \sum_j \beta_j \text{s.t.} \\
&\forall ij \beta_j + \alpha_i b_{ij} \geq b_{ij} \\
&\alpha_i, \beta_j \geq 0
\end{aligned}$$

Suggested exercise: Extend the primal dual analysis from class to the functional BA problem.