CS 388R: Randomized Algorithms, Fall 2023 October 23, 2023

Lecture 15: Online Bipartite Matching

Prof. Eric Price *Scribe: Bhargav Samineni, Steven Xu*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In the previous lecture we discussed algorithms for the Heavy Hitters problem in streaming models and algorithms for the Perfect Matching problem in the special case of d-regular bipartite graphs.

In this lecture we study the Online Bipartite Matching (OBM) problem. We show that the standard greedy approach gives an algorithm that is $1/2$-competitive and that no deterministic algorithm can do better than this. We also show that the naive randomized algorithm is only $(1/2 + o(1))$-competitive and present an improved randomized algorithm due to Karp, Vazirani, and Vazirani [KVV90] that is $(1 - 1/e)$-competitive.

2 Online Bipartite Matching

Online bipartite matching is the problem of matching n "customers" to n "shops", similar to the offline bipartite matching. However, unlike in offline bipartite matching, the customers are unknown and instead reveal themselves in some order, each needing to be matched before future customers are revealed. Previous matches cannot be changed later. More precisely, the input of the problem is:

- a bipartite graph $G = (U, V, E)$ where U is the set of customers, V is the set of shops, and $n = |U| = |V|$
- an ordering of customers $S : [n] \to U$, where S is a bijection.

And the output of the problem is a matching $M : U \to V \cup \{\text{null}\}$ from U to V. We write $|M|$ to denote the number of nodes matched in M—$|M| = |\{u \in U \mid M(u) \neq \text{null}\}|$, and we also write $(u, v) \in M$ to mean that $M(u) = v$.

In this section, we'll go over two naive algorithms for online bipartite matching and analyze their matching performance. However, we'll first need a measure of how well an algorithm performs.

Definition 1. *The competitive ratio of an online bipartite matching algorithm A is*

$$R(A) = \liminf_{n\to\infty} \min_{I \in I_n} \frac{\mathbb{E}[|A(I)|]}{\text{Opt}(I)},$$

where I_n is the set of inputs of size n, and $\text{Opt}(I)$ is the size of optimal matching for I.

Intuitively, if we define the competitive ratio of a particular matching M to be $|M|/\text{Opt}(I)$, the competitive ratio of an algorithm is the expected worst case performance as the size of the input grows.

Note that the expectation is taken over the randomness given to the algorithm, not the input!

2.1 Deterministic Greedy Algorithm

The first algorithm we'll cover is the greedy deterministic algorithm. The algorithm works as follows:

- Fix some ranking of the merchants.
- When a customer arrives, match it to the highest ranking available merchant in its neighborhood.

Note that the matching produced is a maximal matching—if after the algorithm runs, there exists an unmatched customer u which can be matched to an available shop v, then v must've been available when the algorithm was matching u. Since the algorithm only doesn't match a customer when it has no available shops in its neighborhood to be matched to, this is a contradiction. Thus to lower bound the competitive ratio of our algorithm, we can use the following lemma.

Lemma 2. *The competitive ratio of a maximal matching is at least* $1/2$.

Proof. Let M be a maximal matching of G, let M_{OPT} be an optimal matching of G, and let $\alpha_w = 1$ if w is matched in M and 0 otherwise, where w is *any* node in G. Then for any $(u, v) \in M_{\text{OPT}}$, $\alpha_u + \alpha_v \geq 1$ since otherwise, both u and v are unmatched and we could match u and v in M, violating the maximality of M.

But since $2|M| = \sum_{w \in G} \alpha_w$,

$$|M_{\text{OPT}}| = \sum_{(u,v) \in M_{\text{OPT}}} 1 \leq \sum_{(u,v) \in M_{\text{OPT}}} (\alpha_u + \alpha_v) \leq \sum_{w \in G} \alpha_w = 2|M|,$$

so $|M| \geq |M_{\text{OPT}}|/2$. □

This means that the competitive ratio of our algorithm is at least $1/2$. However, it turns out that this is the best we can do for any deterministic algorithm.

Theorem 3. *For any deterministic online bipartite matching algorithm A and any even n, we can construct an input I of size n s.t the competitive ratio of $A(I)$ is at most* $1/2$.

Proof. We give A customers which can be matched to any merchant until $n/2$ customers have been matched. Then we make all the remaining customers we give to A able to be matched to precisely the merchants taken in the previous batch. In the first batch of customers, $\leq n/2$ customers were matched, and in the second batch, none could be matched, so in total, $\leq n/2$ customers were matched.

However, we can construct an optimal matching of size n as follows. First, match all the customers in the second batch, which is possible since there are $\leq n/2$ customers which are each connected

to $n/2$ merchants. Then match all the customers in the first batch, which is possible since each is connected to every merchant.

Thus the competitive ratio is $\leq (n/2)/n = 1/2$. □

This implies that the competitive ratio of any deterministic algorithm is at most 1/2, so the competitive ratio of our greedy deterministic algorithm must be precisely 1/2.

2.2 Naive Randomized Algorithm

The second algorithm we'll cover is the naive randomized algorithm, which works by assigning a customer uniform randomly to the remaining available merchants in its neighborhood. Unfortunately, the competitive ratio for this algorithm is also at most 1/2.

Theorem 4. *For any deterministic online bipartite matching algorithm A and any even n, we can construct an input I of size n s.t the expected competitive ratio of $A(I)$ is at most $1/2 + o(1)$.*

Proof. Let $u_1, \ldots, u_n$ be the list of customers in order, let $v_1, \ldots, v_n$ be a list of all the merchants, and let $L = \{v_{n/2+1}, \ldots, v_n\}$. To construct the input, we will do the following.

1. For the first half of customers ($i \in [n/2]$), we connect u_i to v_i as well as all merchants in L.
2. For the second half of customers, we connect u_i to v_i only. Note that $v_i \in L$.

Note that the size of the best matching is n, since we can simply match u_i to v_i for all i.

Now we will calculate the expected competitive ratio of $A(I)$. The first half of customers will always be matched to some merchant. However, the second half of customers will only be matched if no customer in the first half chose its corresponding merchant. Thus we will lower bound the expected number of merchants in L which were matched to customers in the first half.

For $i \leq n/2$, let X_i be 1 if the ith customer was matched to a merchant in L and 0 otherwise. Then $X = \sum_i X_i$ is the number of customers in the first half which were matched to merchants in L. We know that

$$\begin{aligned}\Pr[X_i = 1] &= 1 - \frac{\#\text{ of available neighbors not in } L}{\#\text{ of available neighbors}} \\ &\geq 1 - \frac{1}{n/2 - (i-1) + 1}.\end{aligned}$$

Thus

$$\begin{aligned}\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n/2} X_i\right] &= \sum_{i=1}^{n/2} \mathbb{E}[X_i] = \sum_{i=1}^{n/2} \Pr[X_i = 1] \\ &\geq \sum_{i=1}^{n/2} \left(1 - \frac{1}{n/2 - (i-1) + 1}\right) \\ &= \frac{n}{2} - \sum_{i=1}^{n/2} \frac{1}{i+1} = \frac{n}{2} - \Theta(\log n).\end{aligned}$$

Since the number of customers in the second half which are able to be matched is $n/2 - X$, we get that the expected competitive ratio is

$$\begin{aligned}
\frac{1}{n}\,\mathbb{E}\left[\#\text{ of customers matched}\right] &= \frac{1}{n}\,\mathbb{E}\left[\frac{n}{2} + \#\text{ of customers matched in the second half}\right] \\
&= \frac{1}{n}\,\mathbb{E}\left[\frac{n}{2} + \left(\frac{n}{2} - X\right)\right] = 1 - \frac{1}{n}\,\mathbb{E}[X] \\
&= 1 - \frac{1}{n}\left(\frac{n}{2} - \Theta(\log n)\right) = \frac{1}{2} + \Theta\left(\frac{\log n}{n}\right) \\
&= \frac{1}{2} + o(1)
\end{aligned}$$

□

3 The KVV Ranking Algorithm

Instead of choosing an edge at random to match as in the naive randomized algorithm, the Ranking algorithm due to Karp, Vazirani, and Vazirani [KVV90] works by randomly assigning a global "rank" to each offline vertex (i.e., merchant). Once an online vertex (i.e., user) arrives, it then attempts to match with a neighboring offline vertex with minimal rank, if this is feasible. To generate the ranks of the offline vertices, the algorithm simply constructs a random permutation σ of them at the start of the procedure. By essentially incorporating the randomness at the start of the algorithm, we can avoid having the same behavior as the naive randomized algorithm in the pathological bad case we constructed earlier.

Algorithm 1 Ranking

Input: The input graph G with only offline vertices revealed, the arrival order of online vertices π, and the random permutation of offline vertices σ

1: **for** $u \in \pi$ **do**
2: Let $N(u)$ be the set of unmatched offline neighbors of u.
3: If $N(u) \neq \emptyset$, match u to unmatched offline vertex v with minimal rank $\sigma(v)$.

To simplify the analysis of Algorithm 1, let $A(G, \pi, \sigma)$ denote the output of the Ranking algorithm applied on an input graph $G = (U \cup V, E)$ with U the set of online vertices and V the set of offline vertices, an online vertex arrival order π, and random rank permutation of the offline vertices σ. Without loss of generality, we can assume that there exists some perfect matching M in G (by, for example, removing any unmatched vertices in a max cardinality matching in G and working on the new graph G'). For vertices $u \in L$ and $v \in R$, if the edge $(u, v) \in M$, then we use the notation that $M(u) = v$ and $M(v) = u$ in the perfect matching M.

We first make the following simple structural observations of the Ranking algorithm.

Lemma 5. *Consider some offline vertex $x \in V$ and let $H = G \setminus \{x\}$ and π_H and σ_H the orderings of online and offline vertices in H that are induced by π and σ, respectively. Then, $A(H, \pi_H, \sigma_H) = A(G, \pi, \sigma) \oplus P$, where P is some augmenting path starting at x of increasing σ.*

Proof. Let y be the online vertex that matched to x in $A(G, \pi, \sigma)$. Then, under $A(H, \pi_H, \sigma_H)$, the vertex y must have matched to some new offline vertex z with higher rank, i.e., $\sigma(z) > \sigma(x)$.

However, this causes the online vertex y' that was originally matched to z in $A(G, \pi, \sigma)$ to match to some new offline vertex z' with $\sigma(z') > \sigma(z)$ in $A(H, \pi_H, \sigma_H)$. This process continues, creating a chain of kicks until an offline vertex that was unmatched under $A(G, \pi, \sigma)$ is encountered. This chain of kicks creates an augmenting path that alternates between matched edges under $A(G, \pi, \sigma)$ and matched edges under $A(H, \pi_H, \sigma_H)$. Taking the symmetric difference of this augmenting path with $A(G, \pi, \sigma)$ thus gives $A(H, \pi_H, \sigma_H)$. □

Lemma 6. *Consider some online vertex $u \in U$ and let $v = M(u)$, i.e., v is the offline vertex u is matched to in the perfect matching of G. If v is not matched in $A(G, \pi, \sigma)$, then u is matched to some $v' \in V$ with smaller rank (i.e., $\sigma(v') < \sigma(v)$).*

Proof. When u arrives, it has at least one unmatched offline neighbor since v is unmatched, so u must be matched in $A(G, \pi, \sigma)$. However, since v must remain unmatched, u must have matched with some neighbor v' with lower rank. □

We also observe the following crucial lemma, whose proof will be deferred. The competitive ratio of Ranking follows directly from this.

Lemma 7. *Let x_t denote the probability (over the choices of σ) that the offline vertex v of rank t (i.e., $\sigma(v) = t$) is matched under $A(G, \pi, \sigma)$. Then,*

$$1 - x_t \leq \frac{1}{n} \sum_{s \leq t} x_s.$$

Theorem 8. *The Ranking algorithm is $1 - 1/e \approx .63$ competitive.*

Proof. Let $s_t = \sum_{i \leq t} x_i$. By Lemma 7, this gives that

$$1 - (s_t - s_{t-1}) = 1 - x_t \leq \frac{1}{n} \sum_{s \leq t} x_s = \frac{s_t}{n}.$$

Rearranging terms gives the recurrence relation $s_t \geq (1 + s_{t-1})\,(n/n+1)$, which expands into

$$\begin{aligned}
s_t &\geq (1 + s_{t-1}) \left(\frac{n}{n+1}\right) \\
&\geq \left(\frac{n}{n+1}\right) + \left(\frac{n}{n+1}\right)^2 + \ldots + \left(\frac{n}{n+1}\right)^t \\
&= \left(\frac{n}{n+1}\right) \left(\frac{1 - \left(\frac{n}{n+1}\right)^t}{1 - \left(\frac{n}{n+1}\right)}\right) && \text{(Partial geometric series formula)} \\
&= n \left(1 - \left(1 - \frac{1}{n+1}\right)^t\right)
\end{aligned}$$

For the offline vertex v with rank t, let y_t be the indicator random variable that equals 1 if v is matched under $A(G, \pi, \sigma)$ and 0 otherwise. Then $\mathbb{E}[y_t] = x_t$, which gives that

$$\mathbb{E}[\text{size of matching}] = \sum_{i \leq n} \mathbb{E}[y_i] = \sum_{i \leq n} x_t = s_n$$

Additionally, by assumption of G having a perfect matching, the size of the optimal matching is n. Hence, the competitive ratio of the algorithm is

$$R = \frac{\mathbb{E}[\text{size of matching}]}{n} = \frac{s_n}{n} \geq \left(1 - \left(1 - \frac{1}{n+1}\right)^n\right) \geq 1 - \frac{1}{e}.$$

□

3.1 An Incorrect Proof of Lemma 7

We now present the original proof of Lemma 7 in the KVV paper [KVV90]. However, there is a subtle error in the proof that went unnoticed for several years that causes it to be **incorrect**.

Lemma 7. *Let x_t denote the probability (over the choices of σ) that the offline vertex v of rank t (i.e., $\sigma(v) = t$) is matched under $A(G, \pi, \sigma)$. Then,*

$$1 - x_t \leq \frac{1}{n} \sum_{s \leq t} x_s.$$

Incorrect Proof of Lemma 7. Let v be the offline vertex with rank t, i.e., $\sigma(v) = t$, and let $u = M(v)$. Define $R_{t-1} \subset U$ as the subset of online vertices matched by $A(G, \pi, \sigma)$ to offline vertices v' with rank $\sigma(v') \leq t - 1$. Then, $\mathbb{E}[|R_{t-1}|] = \sum_{s \leq t-1} x_s$.

Suppose v is unmatched under $A(G, \pi, \sigma)$. By Lemma 6 this implies that u is matched to some offline vertex with smaller rank under $A(G, \pi, \sigma)$, which in turn implies $u \in R_{t-1}$. Hence,

$$\begin{aligned} 1 - x_t = \Pr[v \text{ is unmatched}] &= \Pr[u \in R_{t-1}] \\ &= \frac{\mathbb{E}[|R_{t-1}|]}{n} \\ &= \frac{1}{n} \sum_{s \leq t-1} x_s \\ &\leq \frac{1}{n} \sum_{s \leq t} x_s \end{aligned} \tag{1}$$

where Eq. (1) holds as σ is random, so v is uniformly random, and hence u is also uniformly random. □

The crux of the error lies in Eq. (1), which implicitly assumes that R_{t-1} and u are independent. However, this is in general not true as the choice of u is dependent on v, which is in turn dependent on σ. Since R_{t-1} is inherently dependent on σ, there is some correlation between u and R_{t-1} and hence Eq. (1) does not hold.

3.2 A Correct Proof of Lemma 7

We now present a correct proof of Lemma 7 due to Birnbaum and Mathieu [BM08]. Consider some $u \in U$ and let $v = M(u)$. Let σ' be any permutation of V and let $\sigma'(v) = t$. Let $\sigma^{(i)}$ be the permutation obtained from σ' by removing v and inserting it back such that its rank is i. We use the following lemma.

Lemma 9. *If v is not matched in $A(G, \pi, \sigma')$, then for any choice of i, u is matched under $A(G, \pi, \sigma^{(i)})$ to some offline vertex $v^{(i)}$ whose rank $\sigma^{(i)}(v^{(i)}) \leq t = \sigma'(v)$.*

Proof. By Lemma 6, u must be matched to some offline vertex v' under $A(G, \pi, \sigma')$ such that $\sigma'(v') \leq t - 1$. Since v is unmatched under $A(G, \pi, \sigma')$, if we remove it from the graph and the permutation σ' to get some induced permutation σ'_{-v}, the matching given by the algorithm does not change, i.e., $A(G, \pi, \sigma') = A(G \setminus \{v\}, \pi, \sigma'_{-v})$.

Now, consider the graph G and permutation $\sigma^{(i)}$ and let $v^{(i)}$ be the offline vertex u is matched to under $A(G, \pi, \sigma^{(i)})$. It must be that the permutation induced by removing v from $\sigma^{(i)}$ is exactly σ'_{-v}. Hence, by Lemma 5, we get that

$$A(G, \pi, \sigma') = A(G \setminus \{v\}, \pi, \sigma'_{-v}) = A(G, \pi, \sigma^{(i)}) \oplus P,$$

where P is some augmenting path starting at v of increasing rank $\sigma^{(i)}$. This implies that $\sigma^{(i)}(v^{(i)}) \leq \sigma^{(i)}(v')$. Additionally, by definition of $\sigma^{(i)}$, it must be that $\sigma^{(i)}(v') - \sigma'(v') \leq 1$, which implies $\sigma^{(i)}(v') \leq t$. Hence, we have that $\sigma^{(i)}(v^{(i)}) \leq t$. □

Lemma 7. *Let x_t denote the probability (over the choices of σ) that the offline vertex v of rank t (i.e., $\sigma(v) = t$) is matched under $A(G, \pi, \sigma)$. Then,*

$$1 - x_t \leq \frac{1}{n} \sum_{s \leq t} x_s.$$

Correct Proof of Lemma 7. Given a random permutation σ, pick an offline vertex $v \in V$ uniformly at random and let σ' be the permutation induced by removing v from σ and inserting it back such that it has rank t. Additionally, let $u = M(v)$. By Lemma 9 (where i is chosen appropriately such that $\sigma^{(i)} = \sigma$), if v is unmatched under $A(G, \pi, \sigma')$, then u must be matched to some vertex $\bar{v} \in V$ with $\sigma(\bar{v}) \leq t$.

Recall that R_t is the set of online vertices matched by $A(G, \pi, \sigma)$ to offline vertices with rank $\leq t$. Hence, it must be that $u \in R_t$. We now have that u is dependent only on v, and v is clearly independent of σ. Thus, u is independent of σ and independent of R_t. Therefore,

$$\begin{aligned} 1 - x_t = \Pr[v \text{ unmatched under } \sigma'] &\leq \Pr[u \in R_t] \\ &= \mathop{\mathbb{E}}_{\sigma}[\Pr[u \in R_t \mid \sigma]] \\ &= \frac{\mathbb{E}_{\sigma}[|R_t|]}{n} \\ &= \frac{1}{n} \sum_{s \leq t} x_s \end{aligned}$$

□

References

[BM08] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM SIGACT News*, 39(1):80–87, 2008. URL: https://bbirnbaum.com/assets/publications/sigact08.pdf.

[KVV90] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. of the 22nd Annual ACM Symposium on Theory of Computing (STOC 1990)*, pages 352–358, 1990. URL: https://people.eecs.berkeley.edu/~vazirani/pubs/online.pdf.