CS 388R: Randomized Algorithms, Fall 2019 October 15th, 2019

Lecture 14: Online Bipartite Matching

Prof. Eric Price *Scribe: William Kretschmer, Justin Yirka*

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

1 Overview

In the last lecture, we introduced the *online bipartite matching* problem: We're given a bipartite graph with n left and n right vertices which we think of as corresponding to "customers" and "merchants" respectively, and edges exist between merchants and the customers they would like to advertise to. The merchants are known in advance while the customers come in one at a time (the corresponding edges are only revealed after a customer arrives).

When a customer comes in, we have to decide then which merchant to match them to, and we can't go back and make changes: once a customer is assigned to a merchant, neither vertex can be reassigned to a different edge later. Our goal is to match as many vertices as possible — i.e. we want our online algorithm to perform as close to an offline algorithm, with full information, as possible

Last time, we saw that a naive greedy algorithm gives a guaranteed $\frac{1}{2}$-approximation to a maximum matching and that no deterministic algorithm can provide a better guarantee. Today, we'll see how to achieve an expected approximation ratio of $1-\frac{1}{e} \approx 0.632$ using a randomized approach described by Karp, Vazirani, and Vazirani [KVV90], for which a corrected analysis was given by Goel and Mehta [GM08].

2 Deterministic Algorithms

One naive greedy algorithm is to fix some ranking on the merchants and whenever a vertex v arrives, match v to the highest priority unmatched merchant in $N(v)$.

First, we lower bound the performance of this algorithm.

Proposition 1. *This algorithm is guaranteed to achieve at least a $\frac{1}{2}$-approximation.*

Proof. Because the algorithm will always add edges to the matching if possible, the output is a maximal matching, and it is known that maximal matchings are always at least $\frac{1}{2}$ the size of a maximum matching (see any graph theory textbook). □

Next, we can show the following which upper bounds the performance of not just this algorithm but *any* deterministic algorithm.

Proposition 2. *No deterministic algorithm can guarantee better than a $\frac{1}{2}$-approximation in the worst case.*

Proof. We will refer to the customers by the order in which they arrive.

Consider any deterministic algorithm; we know its ranking on the merchants ahead of time. Suppose the input is a graph in which the first $\frac{n}{2}$ customers are fully connected to all merchants and the final $\frac{n}{2}$ customers are fully connected only to the $\frac{n}{2}$ highest ranked merchants. Then, the first $\frac{n}{2}$ customers will match with the highest ranked merchants and the final $\frac{n}{2}$ customers will have no one left to match with.

There exists a perfect matching, with the first $\frac{n}{2}$ customers matched to the lower ranked merchants and the final $\frac{n}{2}$ customers matched with the higher ranked merchants, but this algorithm only achieved a matching of size $n/2$. □

3 A (Bad) Randomized Algorithm

Before we begin to discuss randomness, we make an assumption about the online input which we didn't explicitly state in class:

Assumption 3. *We assume the merchants' preferences and the order of the customers is determined before runtime, so the input may be adversarial but it is non-adaptive. Importantly, this means the input cannot adapt to our random choices; adversaries can only use our deterministic choices against us.*

In fact, [KVV90] proves that even if there exists a perfect matching, no randomized algorithm can achieve a matching of size more than $\frac{n}{2} + O(\log n)$ in expectation if the input is from an adaptive adversary (so the ratio approaches $\frac{1}{2}$ for large n).

In the previous section, we introduced a deterministic greedy algorithm which chose a fixed global ranking on the merchants. What if instead, whenever a customer arrives, we randomly pick a merchant in the customer's neighborhood? This can be viewed as randomly assigning each customer its own ranking of the merchants.

Unfortunately, this algorithm achieves an expected approximation ratio in the worst case approaching $\frac{1}{2}$ as n grows large, so it doesn't improve on the deterministic case by much.

Proposition 4. *This randomized algorithm achieves an expected approximation ratio of at most* $\frac{\frac{n}{2}+O(\log n)}{n}$ *in the worst case.*

Proof. Label the customers by c_i for i in the order they appear and the merchants by m_j in any order, for $1 \leq j \leq n$. Suppose that all customers c_i are connected to the corresponding merchant m_i, and that the first $\frac{n}{2}$ customers are fully connected to the second half of merchants $m_{n/2+1}, \ldots, m_n$.

Intuitively, we expect the first $\frac{n}{2}$ customers to match half with the first $\frac{n}{2}$ merchants and half with the second $\frac{n}{2}$ of merchants, leaving half of the second $\frac{n}{2}$ customers unmatched. The analysis is actually a little subtler, so we'll give the formal version.

Observe that for the first $\frac{n}{2}$ customers, the probability a customer c_i doesn't match with its corresponding neighbor in the first $\frac{n}{2}$ merchants and instead matches with one of the second $\frac{n}{2}$ merchants decreases when more of the second half of merchants have previously been matched: the probability

is smallest when all previous $i-1$ customers matched with the second half of merchants, giving a probability of $1 - \frac{1}{\frac{n}{2}-i+2}$. Therefore, we expect

$$\begin{aligned}
\mathbb{E}[\#\text{ of } c_i \text{ not matched with } m_i] &= \sum_{i=1}^{n/2} 1 - \mathbb{P}[c_i \text{ matches with } m_i] \\
&= \frac{n}{2} - \sum_{i=1}^{n/2} \mathbb{P}[c_i \text{ matches with } m_i] \\
&\geq \frac{n}{2} - \sum_{i=1}^{n/2} \frac{1}{\frac{n}{2}-i+2} \\
&\geq \frac{n}{2} - \log(\frac{n}{2}+1)
\end{aligned}$$

,

so at most $\log(\frac{n}{2}+1)$ of the second $\frac{n}{2}$ customers will have their corresponding merchant still available to match with.

Clearly, there exists a perfect matching of size n: just use the edges $\{c_i, m_i\}$, but here we expect to match all of the first $\frac{n}{2}$ customers and only up to $O(\log \frac{n}{2} + 1)$ of the second half, yielding the claim. □

4 The (Good) Randomized Algorithm

The previous (bad) algorithm had each customer pick a random ranking of the merchants. The idea of this algorithm [KVV90] is to pick a global random ranking of the merchant (right) vertices and to assign each new customer (left) vertex to the highest ranked right vertex that's available.

Formally, suppose we have a graph G with left vertices U and right vertices V. Randomly choose a ranking σ defined by a bijection from $V \to \{1, \ldots, n\}$. We'll follow the convention that lower values correspond to higher rank. When a left vertex u gets added, u gets matched to the unmatched right vertex v with the smallest $\sigma(v)$.

Theorem 5 ([KVV90, GM08])**.** *This algorithm outputs at least a $\left(1 - \frac{1}{e}\right)$-approximation to the maximum matching in expectation (where the expectation is over the random choice of σ).*

Before proving Theorem 5, we state three Lemmas, proving some now and some later.

Lemma 6. *Given an instance of this algorithm, let σ denote the random ranking of right vertices and let π denote the order in which left vertices arrive. Let M be the output matching. For any right vertex v, let $H = G - x$. Let M' be the output matching when we run the algorithm on H with the same ranking and order σ, π, with v removed.*

If M and M' are not identical, then $|M'| \leq |M|$ and they differ by an augmenting/alternating path starting at x.

Proof. We will refer to some edges as directed in order to emphasize the alternating path.

By deleting a vertex from the right, a left-to-right edge may be deleted. The left-endpoint of that deleted edge is some customer who, when they arrive, will now look to match with a new merchant. If one is available, this creates a new left-to-right edge in the matching. Now, the process repeats, as that added edge in M' may have displaced an edge in M, so we have another deleted left-to-right edge, which forces another customer to look for a new left-to-right edge, and so on.

Note that this means M' will not include any matched vertices on the left that were not matched in M, since the alternating path only visits customers which were matched in M. So, $|M'| \leq |M|$. $\square$

Now, suppose some graph G does not have a perfect matching, but it has a maximum matching M of size k. Then, apply the reasoning from the previous proof repeatedly to remove the vertices on the right in G which are not in M and create a graph H on $2k$ vertices for which there exists a perfect matching M' of size k. If we claim this algorithm achieves an ϵ-approximation on perfect graphs, then running it on H will give a matching of size at least ϵk. By the above Lemma, running the algorithm on G will not produce a worse matching than on H, so it will still output a matching of size at least ϵk, which is an ϵ-approximation in G. Therefore, it will suffice to only consider graphs which we assume to have perfect matchings.

Henceforth, assume G contains a perfect matching $M : U \leftrightarrow V$.

Lemma 7. *Let $u \in U$ and $v = M(u)$. If v is unmatched after running this algorithm, then u was matched to some v' with $\sigma(v') < \sigma(v)$.*

Proof. Since v is unmatched, we know that when customer u arrives, it could have matched u with v. The algorithm would only not match u to an available vertex v if it instead chose to match u to some v', which would require $\sigma(v') < \sigma(v)$. $\square$

Lemma 8. *Let X_t be the probability (over the choice of σ) that the rank-t vertex is matched, i.e.*

$$X_t := \mathbb{P}\left[v \text{ such that } \sigma(v) = t \text{ is matched}\right].$$

Then,

$$1 - X_t \leq \frac{1}{n} \sum_{s \leq t} X_s.$$

Lemma 8 tells us that higher ranked vertices (when the number of $s \leq t$ is small) are very likely to be matched, and even the lower ranked vertices have decent probability of being matched.

Example: Clearly, $X_1 = 1$. So by Lemma 8, $1 - X_2 \leq \frac{1}{n} + \frac{X_2}{n}$, implying $X_2 \geq (1 - \frac{1}{n})(\frac{n}{n-1}) = 1 - O(\frac{1}{n})$.

We will prove Lemma 8 later. Assuming it for now, we are ready to prove Theorem 5.

Proof of Theorem 5 (assuming Lemma 8). Let the random variable $S_t := \sum_{s \leq t} X_s$ denote the expected number of right vertices of rank at most t which will be matched. Applying Lemma 8 to S_t

and S_{t-1} and combining implies

$$\begin{aligned}
\frac{1}{n} S_t &\geq 1 - (S_t - S_{t-1}) \\
S_t \cdot \frac{n+1}{n} &\geq 1 + S_{t-1} \\
S_t &\geq \frac{n}{n+1} + \left(\frac{n}{n+1}\right)^2 + \ldots + \left(\frac{n}{n+1}\right)^t \\
S_t &\geq \frac{n}{n+1} \cdot \frac{1 - (\frac{n}{n+1})^t}{1 - \frac{n}{n+1}} = n\left(1 - \left(\frac{n}{n+1}\right)^t\right).
\end{aligned}$$

We want to know the expected ratio of the size of the matching to the perfect matching n, which is $\frac{S_n}{n}$. By the above inequality, this ratio is at least

$$\frac{S_n}{n} \geq 1 - \left(1 - \frac{1}{n+1}\right)^n \approx 1 - \frac{1}{e},$$

which was the claim. So, once we prove Lemma 8, we are done! □

5 Original Proof of Lemma 8

In this section, we give the proof of Lemma 8 as given by [KVV90]. If something is confusing, skip to the note after the proof...

Original Proof of Lemma 8. Note that the input G is fixed, so the ordering of customers and the perfect matching M are henceforth fixed. For any rank t, let v be the right vertex such that $\sigma(v) = t$ and set the left vertex $u = M(v)$ for M the perfect matching. Because σ is random, v is uniformly distributed over $\{1, \ldots, n\}$, so u is also uniform.

Define the set of left vertices matched with vertices of rank at most $t-1$ to be

$$R_{t-1} = \{u' \in U : u' \text{ is served by } v' \text{ such that } \sigma(v') \leq t-1\}.$$

Notice that $\mathop{\mathbb{E}}_{\sigma}[|R_{t-1}|] = \sum_{i=1}^{t-1} X_i$.

By Lemma 7, if v is unmatched, it implies that u is served by some v' with $\sigma(v') \leq t-1$, which implies that $u \in R_{t-1}$. But we know u is uniformly distributed! So, the probability of this is

$$\mathop{\mathbb{P}}_{u}[u \in R_{t-1}] = \frac{|R_{t-1}|}{n}. \tag{1}$$

Finally, since X_t is over the choice of σ, we use the expected size of R_{t-1} over the choice of σ and have

$$1 - X_t = \mathbb{P}[v \text{ unmatched}] \leq \mathbb{P}[u \in R_{t-1}] \leq \frac{\mathop{\mathbb{E}}_{\sigma}[|R_{t-1}|]}{n} = \frac{1}{n} \sum_{s \leq t-1} X_s, \tag{2}$$

as desired. □

This proof is **wrong** as stated!

The above proof is similar to what was given in [KVV90], and it took 17 years for a bug to be discovered by Erik Krohn and Kasturi Varadarajan in 2007 (cited by Goel and Mehta [GM08]).

The problem is subtle: while the choice u is uniformly random, it is *not* independent of the choice of σ or any choices depending on σ. In particular, in Equation (1), we did not explicitly state the conditional probability $\mathop{\mathbb{P}}_{u}[u \in R_{t-1} \mid \sigma]$, which was not yet a problem since the probability was only over u, so the choices were still independent. But in Equation (2), where X_t is defined over the choice of σ, we have probabilities over both u and σ, and u is dependent on σ, so they are no longer independent, and the relationship from Equation (1) no longer holds.

Fortunately, a fix was soon given by [GM08]. In fact, our statement of Lemma 8 is the corrected one from [GM08], but the original claim by [KVV90] was only different by a constant (t vs $t-1$). Really, the problem was only in the analysis, not the result.

6 Fixed Proof of Lemma 8

Intuitively, there shouldn't be much correlation between the choice of u according to σ and the choice of R_{t-1} over σ since fixing u should only shift the previous matched vertices by some alternating path. Also, note that the our statement of Lemma 8 differs from the result concluded in Equation (2) by an index of t, rather than $t-1$, in the summation. In fact, this discrepancy will ultimately give us the room needed to fix this proof, as we will see below.

The idea to fix this proof is to choose u based on a modified permutation σ' so that u and σ are independent. Set σ' equal to σ but with $v = M(u)$ moved to rank t, so that everything between rank t and v gets shifted over by one position between σ and σ'.

Lemma 9. *Let σ' be a permutation, and let $\sigma^{(i)}$ be σ' but with the rank-t vertex moved to rank-i and all other vertices shifted by one to accommodate.*

If v is not matched by the algorithm when using the ranking σ', then for all i, when the algorithm uses the ranking $\sigma^{(i)}$, u is matched to some $v^{(i)}$ where $\sigma^{(i)}(v^{(i)}) \le \sigma'(v)$.

Proof. Under σ', v is unmatched, so under σ' if we think of v as being removed, nothing changes. Therefore, the matching produced under $\sigma^{(i)}$ is the same as under σ' expect perhaps it differs by a single augmenting path (Lemma 6). Moreover, the vertices in the path are in order of increasing priority.

So how does the ranking of the vertex to which u matches change? We can think about it as removing the old position of v and reinserting, where removing does nothing, and inserting can grow the position of u's match by 1. Overall: $\sigma^{(i)}(\text{matched to } u) \le \sigma'(\text{matched to } u) + 1$. □

Now, let's use this to fix our proof of Lemma 8:

Fixed proof of Lemma 8. Pick σ and u independently and modify to σ' as above. Then σ' is still a uniformly random permutation. By Lemma 9, if v is unmatched in σ', then u under σ is matched to some v' with $\sigma(v') \le \sigma'(v) = t$. So $\mathbb{P}[\sigma \text{ unmatched under } \sigma'] = 1 - X_t \le \mathbb{P}[\sigma(v') \le t]$.

Similarly to before, define $R_t = \{u' \in U : u' \text{ is served by } v' \text{ such that } \sigma'(v') \leq t\}$. Then:

$$1 - X_t \leq \mathop{\mathbb{P}}_{u,\sigma}[u \in R_t] = \mathop{\mathbb{E}}_{\sigma}[\mathop{\mathbb{P}}_{u}[u \in R_t | \sigma]] = \mathop{\mathbb{E}}_{\sigma}\left[\frac{|R_t|}{n}\right] = \frac{1}{n}\sum_{s \leq t} X_s.$$

□

So the key differences between the broken proof and the fixed proof are that R_{t-1} got replaced by R_t, and u and σ are now independent.

References

[GM08] Gagan Goel and Aranyak Mehta. Online Budgeted Matching in Random Input Models with applications to Adwords. *Proceedings of SODA 2008*, 982–991, 2008.

[KVV90] Richard Karp, Umesh Vazirani, and Vijay Vazirani. An Optimal Algorithm for On-line Bipartite Matching. *Proceedings of STOC 1990*, 352–359, 1990.