CSE525: Randomized Algorithms and Probabilistic Analysis

Lecture 6

Lecturer: Anna Karlin *Scribe: Alex Polozov, Daryl Hansen*

1 Online Bipartite Matching

Online Bipartite Matching is a generalization of a well-known Bipartite Matching problem. In a Bipartite Matching, we a given a bipartite graph $G = (L, R, E)$, and we need to find a matching $M \subseteq E$ such that no edges in M have common endpoints. In the online version L is known, but vertices in R are arriving one at a time. When vertex $j \in R$ arrives (with all its edges), we need to make an irreversible decision to match j with one of its neighbors $i \in N(j) \subseteq L$.

Performance of different algorithms A (possible randomized) in comparison to optimal (offline) algorithm is called *competitive ratio*:

Definition 1 (Competitive ratio)**.**

$$\mathsf{c.r.}(A) = \liminf_{I} \frac{\mathbb{E}\left[A(I)\right]}{OPT(I)}$$

where I – input instances (graph + arrival order), and OPT is the optimum offline algorithm.

An obvious deterministic online algorithm is greedy – the one that arbitrarily assigns a node $i \in N(j)$ for every $j \in R$ arrived.

Theorem 2. *The competitive ratio of greedy algorithm is* $1/2$.

Proof Let's assign "profit" to matched nodes, measuring the size of matching partially. When node $j \in R$ arrives, we have a possibility to distribute \$1. Some part of this \$1 we will be able to claim later as part of our total profit (matching size).

If we can match j with some node $i \in N(j)$, let's put 50¢ on every endpoint of the matched edge. The 50¢ put on j becomes part of our profit, but 50¢ put on i is not – it signifies that by matching i with j we might have blocked a possibility for matching i with some other node in future.

If we cannot match j (but offline algorithm can), it means that all nodes in $N(j)$ have already been matched with some other nodes before. In this case they have 50¢ put on them when matched. We can allocate at least one of these 50¢ to j, claiming it as part of our profit.

Thus, for every \$1 that offline algorithm "earns" (for every matched edge), the greedy algorithm "earns" 50¢ allocated on the right endpoint one way or another. Hence, greedy matching size $\geq \frac{1}{2}$ optimum matching size. ■

No deterministic algorithm can achieve a better competitive ratio, because for a given algorithm A an adversary can easily construct a tight graph instance I on which A matches only 1 edge whereas the offline algorithm matches two. This instance is as follows:

$$L = R = \{1, 2\}$$

$$E = \{(1,1),\ (2,1)\} \cup \begin{cases} \{(1,2)\} & \text{if } A \text{ selects } (1,1) \text{ on the first iteration} \\ \{(2,2)\} & \text{if } A \text{ selects } (2,1) \text{ on the first iteration} \end{cases}$$

Therefore we can hope to find a better approximation only with a randomized algorithm. However, as it turns out, there exists a simpler reformulation of the problem that is in some sense equivalent to the randomized version.

2 Fractional Bipartite Matching

A fractional version of the Bipartite Matching problems allows for every node $j \in R$ to be allocated *partially* with nodes $i \in N(j)$ in fractions x_{ij}. All allocated fractions should sum to at most 1 for every graph node.

In other words, both integral and fractional versions of the Bipartite Matching problem can be encoded with the following linear program:

$$\begin{aligned} &\text{maximize} \sum_{i \in L} \sum_{j \in R} x_{ij} \\ &\sum_{j \in N(i)} x_{ij} \leq 1 \qquad \forall i \in L \\ &\sum_{i \in N(j)} x_{ij} \leq 1 \qquad \forall j \in R \end{aligned} \tag{1}$$

With the constraint $x_{ij} \in \{0, 1\}$ it encodes an integral version, with $x_{ij} \in [0, 1]$ it encodes a fractional version.

Claim 3. *Let A be a randomized algorithm for integral Bipartite Matching. There exists a deterministic algorithm D for fractional Bipartite Matching such that*

$$\forall i \quad \sum_{j \in N(i)} x_{ij}^D = \mathbb{E}\left[\sum_{j \in N(i)} x_{ij}^A\right]$$

where x_{ij}^D and x_{ij}^A are the matchings assigned by D and A in the solution of (1), respectively.

Proof Deterministic fractional algorithm D that "simulates" A can be built as follows. When node $j \in R$ arrives, let $x_{ij}^D = \Pr\left(x_{ij}^A = 1\right)$. The latter probability can be calculated from the corresponding subtree of all possible coin tosses of the algorithm A. The required equality follows trivially. The constraints of (1) still hold, because

$$\forall i \quad \sum_{j \in N(i)} x_{ij}^A \leq 1 \quad \Rightarrow \quad \mathbb{E}\left[\sum_{j \in N(i)} x_{ij}^A\right] \leq 1 \quad \Rightarrow \quad \sum_{j \in N(i)} x_{ij}^D \leq 1$$

■

Corollary 4. *Let A be a randomized algorithm for integral Bipartite Matching. Then there exists a deterministic algorithm D for fractional Bipartite Matching such that $\mathsf{c.r.}(D) = \mathsf{c.r.}(A)$. Consequently, any upper bound on the competitive ratio of deterministic fractional algorithms is also an upper bound of the competitive ratio of randomized integral algorithms.*

Proof From the construction in the proof of claim 3 we have

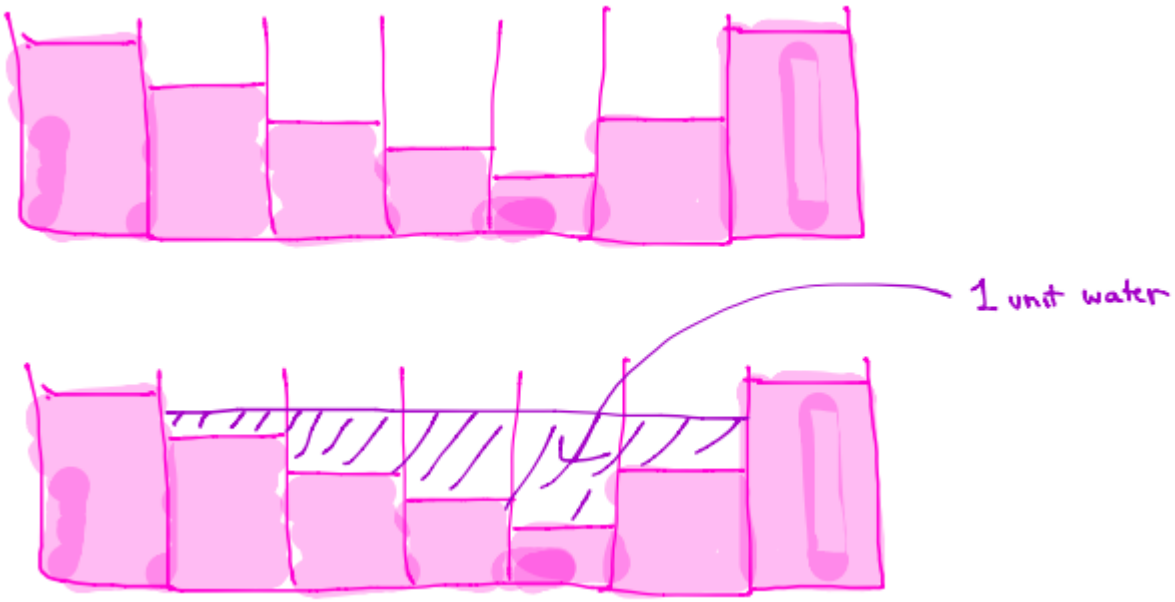
$$\sum_{(i,j) \in E} x_{ij}^D = \mathbb{E}\left[\sum_{(i,j) \in E} x_{ij}^A\right]$$

which is essentially an expectation of performance of A in the definition of competitive ratio. ■

Thus, we need only to construct the "worst" possible instance for deterministic fractional algorithm – an upper bound for their performance on such instance will also be an upper bound for randomized integral algorithms. For that, we introduce the most natural fractional algorithm, present the worst instance for it, and then prove that no other fractional algorithm can outperform it.

3 Water Level Algorithm

The so called "Water Level" algorithm works as follows: when node $j \in R$ arrives, it bring 1 unit of "water". It is distributed between neighbors $i \in N(j)$ in fractions x_{ij} such that the total amount of water in every $i \in N(j)$ is the same after the matching, if possible:



The Water Level algorithm is in some sense the most "cautious" fractional algorithm: it never puts too much water in any specific node $i \in N(j)$, leaving nodes as open as possible for the further arriving $j \in R$. The worst case instance for it is the following graph:

$$L = R = \{1, \dots, n\}$$
$$E = \{(i, j) \mid i \in L,\ j \in R,\ i \geq j\}$$

When node 1 arrives, every node $i \in L$ gets $x_{i1} = 1/n$ water. On the second iteration all the nodes in $N(2) = \{2, \dots, n\}$ get additionally $x_{i2} = 1/(n-1)$ water, etc. This process continues until some i^{th} arriving node overflows the capacity of its neighbors and thereby completes the matching.

The amount of water in the node $i \in L$ is $\frac{1}{n} + \frac{1}{n-1} + \dots + \frac{1}{i} = H_n - H_{i-1} \approx \ln \frac{n}{i}$. It overflows when the amount is equal to 1, i.e. $i = \frac{n}{e}$. The size of the built matching is therefore $n\left(1 - \frac{1}{e}\right)$. The optimal (offline) fractional algorithm allocates every node $j \in R$ in full to the corresponding node $j \in L$, building a matching of size n. Hence, the competitive ratio of the Water Level algorithm is at most $1 - \frac{1}{e}$.

Any other deterministic fractional algorithm does not perform better on this instance, either. If some algorithm D allocates water in any other pattern than equal, the adversary can pick a node i with the minimal water amount put on the first iteration, and move it earlier in the arrival order. This moves more water to the lower part of the graph, comparing with the Water Level algorithm, and therefore overflows even sooner, achieving a smaller competitive ratio.

Corollary 5. *The competitive ratio of any deterministic fractional algorithm is at most* $1 - \frac{1}{e}$.

4 Water Level Analysis

4.1 Cash Placement

This analysis is an analogue of the cash analysis of the Greedy algorithm. For each $i \in L$ we can define α_i to be the amount of money currently placed on i, and similarly for each $j \in R$, we define β_j to be the

amount of money placed on j. As in the greedy algorithm, we want to split our gain between α_i and β_j, but when the water level is high, we want to leave a greater portion of it on i to protect ourselves from making a mistake in the future (that is, being in a situation where we have already filled i and the offline algorithm is still able to place more onto it). To do so, let us use an increasing function $g : [0,1] \to [0,1]$ – the particular function will be pinned down later.

When x_{ij} increases (infinitesmally) by $\mathrm{d}x$, we can thus increase α_i by $g(y_i)\mathrm{d}x$ and β_i by the remainder, $(1 - g(y_i))\mathrm{d}x$, where

$$y_i = \sum_{j \in N(i)} x_{ij}$$

Therefore,

$$\sum_{(i,j)\in E} x_{ij} = \sum_i \alpha_i + \sum_j \beta_j$$

This means that at any moment,

$$\alpha_i = \int_0^{y_i} g(x)\,\mathrm{d}x$$

4.2 c-Approximation

To prove that Water Level has a competitive ratio of at least c, we wish to pick a g with the following property:

$$\forall (i,j) \in E, (\alpha_i + \beta_j) \geq c$$

Let x^*_{ij} be the value that OPT assigns to the edge (i,j). If the above equation holds, then

$$\sum_{(i,j)\in E} x_{ij} = \sum_i \alpha_i + \sum_j \beta_j \geq \sum_i (\alpha_i \sum_{j\in N(i)} x^*_{ij}) + \sum_j (\beta_j \sum_{i \in N(j)} x^*_{ij}) = \sum_{(i,j)\in E} (\alpha_i + \beta_j) x^*_{ij} \geq \sum_{(i,j)\in E} c x^*_{ij} = c\,OPT$$

which proves that Water Level is c-competitive. The task remaining is to find a g that maximizes c, to prove as good of a ratio as possible.

4.3 Selection of g

Fix some edge (i,j).
Let y_i^f be the final water level at node i. There are two possible cases:

1. $y_i^f = 1$:

   $$\alpha_i = \int_0^1 g(x)\,\mathrm{d}x = G(1) - G(0)$$

   where $G'(x) = g(x)$. In this case, it is possible that some neighboring j is completely unfilled, and thus $\beta_j =$, so we need it to be the case that $\alpha_i = G(1) - G(0) = c$.

2. $y_i^f < 1$:
   In this case, j must be fully matched, so $\sum_{\tilde{\imath} \in N(j)} x_{\tilde{\imath} j} = 1$.
   Define $y = \min_{\tilde{\imath} \in N(j)} y_{\tilde{\imath}}$, and notice that $y_i^f \geq y_i \geq y$. $1 - g(x)$ is a decreasing function, so applying a max-length estimate we find that:

   $$\beta_j \geq 1 - g(y) \geq 1 - g(y_i) \geq 1 - g(y_i^f)$$

   $$\alpha_i + \beta_j \geq G(y_i^f) - G(0) + 1 - g(y_i^f)$$

4.4 Differential Equations

From case 2, we want that $\forall y \in [0,1) : G(y) - G(0) + 1 - g(y) = c$.[1] By taking a derivative, we find that $g(y) - g'(y) = 0$, which implies that $g(x) = ke^y$ for some k.
If we substitute $g(x) = G(x) = ke^x$ into our previous equation, what we find is that $ke^y - ke^0 + 1 - ke^y = 1 - k = c$.
From the initial condition in case 1, $G(1) - G(0) = c = \int_0^1 ke^x \, \mathrm{d}x = k(e-1)$.
Therefore $1 - k = k(e-1) \implies k = \frac{1}{e}$, and therefore $c = 1 - \frac{1}{e}$.

4.5 Conclusion

The takeaway is that using the function $g(x) = e^{x-1}$ in the analysis of subsection 4.2 proves that the competetive ratio of Water Level is at least $1 - \frac{1}{e}$. In an earlier section, we have shown that this is precisely the upper bound of the competitive ratio of *any* fractional deterministic algorithm, which leads to two important facts:

1. Water Level has a competitive ratio of exactly $1 - \frac{1}{e}$
2. Water Level is an optimal strategy for fractional deterministic online bipartite matching.

[1] We replaced an inequality with equality here, so as to get the best bound possible