Fully Online Bipartite Matching

Zhu Jingcheng

January 19, 2024

1 Greedy Algorithm

1.1 Algorithm

Algorithm 1 GREEDY Algorithm

```
while the events happen do
  if Vertex x releases then
    Construct the bipartite graph by adding x and relative edges.
  else if Vertex x expires then
    if N(x) is NULL then
      x is unmatched forever.
    else
      x matches the first vertex in N(x).
    end if
  end if
end while
```

1.2 Analysis

The obvious greedy algorithm has a matching competitive ratio of 0.5. It can be obtained by the following theorems.

Theorem 1.1. *GREEDY on the full online bipartite matching with adversarial inputs has the competitive ratio at least* 0.5*.*

Proof. For every edge (u, v) in the perfect matching of B, either u or v is present in the matching generated by the algorithm. Otherwise, the matching can be augmented by adding (u, v). So, at least $\frac{n}{2}$ vertices in the matching, i.e. it has a competitive ratio at least 0.5. □

Theorem 1.2. *GREEDY on the full online bipartite matching with adversarial inputs has the competitive ratio at most* 0.5*.*

Proof. Any bipartite graphs with $|U| = n$ and $|V|$, and a maximal matching of size $\frac{n}{2}$ can be used to construct an adversarial input that forces GREEDY produce that particular maximal matching. The order is, first, to group vertices by pairs, based on the $\frac{n}{2}$-sized matching. Any following arrival of vertices won't produce matching because the existing matching is maximal. □

2 Water Filling

In class, we consider the already known vertices L as goods, and the arriving Vertex R as buyers. Each buyer brings a watering can filled with one unit of water and each good corresponds to a tank that holds at most one unit of water.

In this problem, the main difference compared to the above is the lack of a classification of the vertices into buyers and goods a priori. When a vertex arrives, it first acts as a good, passively waits for someone to buy it. Once it reaches the deadline, however, it will act as a buyer, immediately buying the cheapest neighboring vertex which has not yet been matched.

We can extend the Water-Filling approach to the Fully Online Fractional Bipartite Matching Problem: each vertex i acts as a good until its deadline at which point it acts as a buyer. The price of a neighboring good $j \in N(i)$ is then $p\left(w_j\right) := \frac{1}{\sqrt{2}} w_j + 1 - \frac{1}{\sqrt{2}}$ where w_j is how much of j has been matched so far. Accordingly, each buyer will solve the convex program

$$\begin{aligned} \max_{(\Delta x_{i,j})_{j \in N(i)}} & \sum_{j \in N(i)} \left(\Delta x_{i,j} - \int_{w_j}^{w_j + \Delta x_{i,j}} p(w) \mathrm{d}w \right) \\ \text{s.t.} & \sum_{j \in N(i)} \Delta x_{i,j} \leq 1 - w_i, \\ & \Delta x_{i,j} \geq 0 \quad \forall j \in N(i) \end{aligned} \tag{2.1}$$

to maximize their utility and pay $\int_{w_j}^{w_j + \Delta x_{i,j}} p(w) \mathrm{d}w$ to each neighbor $j \in N(i)$.

Algorithm 2 Water-Filling

for each vertex i that expires **do**
  Compute an optimum solution $(\Delta x_{i,j})_{j \in N(i)}$ to the 2.1.
  $x \leftarrow x + \Delta x$
end for

We let g_i be the gain of vertex i, i.e. the sum of its revenue while acting as a good and its utility while acting as a buyer.[1]

Lemma 2.1. *Let $(i, j) \in E$ be arbitrary with i's deadline being earlier. Then at the end of Algorithm 2, we have $g_i + g_j \geq 2 - \sqrt{2}$.*

Proof. Consider the matched levels w_i and w_j right after i 's deadline. Either $w_i = 1$ or $w_j = 1$; otherwise, i could have been matched more to j during its departure.

Note that if $w_j = 1$, since j is always considered as goods, we have $g_j = \int_0^1 p(w) \mathrm{d}w$.

Now assume $w_i = 1$ and $w_j < 1$. Further consider the matched level w_i' before i's deadline, i.e. how much of i was bought while it was acting as a good. Vertex i gains $\int_0^{w_i'} p(w) \mathrm{d}w$ in revenue from those previous matches. In addition, it gains at least $1 - p\left(w_j\right)$ per unit of good that i buys on departure as it could have always bought j instead. Thus

$$g_i \geq \int_0^{w_i'} p(w) \mathrm{d}w + \left(1 - w_i'\right)\left(1 - p\left(w_j\right)\right)$$

Further, $g_j = \int_0^{w_j} p(w) \mathrm{d}w$. Let $p(x) = a(x-1) + 1$, when $a = \frac{1}{\sqrt{2}}$we have

$$g_i + g_j \geq \int_0^{w_i'} p(w) \mathrm{d}w + \left(1 - w_i'\right)\left(1 - p\left(w_j\right)\right) + \int_0^{w_j} p(w) \mathrm{d}w$$

$$= \frac{1}{2\sqrt{2}}(w_i' + w_j - 2 + \sqrt{2})^2 + 2 - \sqrt{2} \geq 2 - \sqrt{2}$$

This time $g_i + g_j \geq g_j = \int_0^1 p(w)\mathrm{d}w = 1 - \frac{\sqrt{2}}{4} \geq 2 - \sqrt{2}$. $\square$

As a corollary we immediately obtain the following theorem.

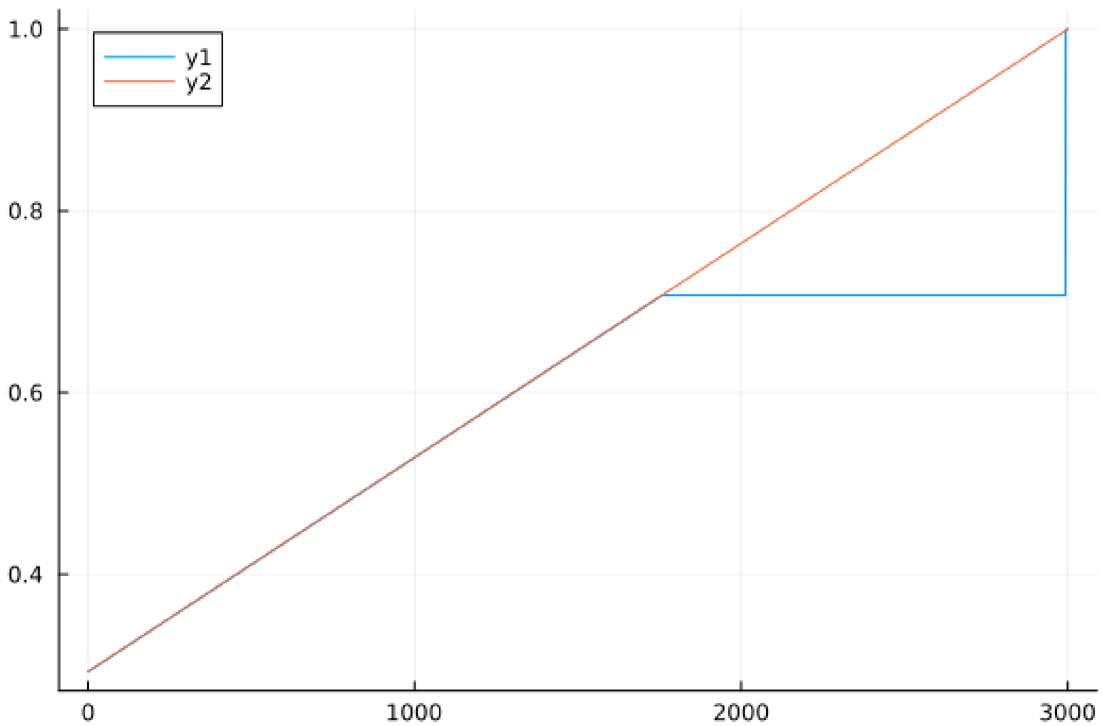
Theorem 2.1. *The competitive ratio of Water-Filling for the Fully Online Fractional Matching Problem is* $2 - \sqrt{2} \approx 0.585$.

We can use a LP solver to achieve this ratio.

$$\begin{aligned}
&p(x) = p_i, x \in [\frac{i-1}{n}, \frac{i}{n}), i \in [n]. \\
&\max r = \min_{w_{i'}, w_j} \left(\int_0^{w_i'} p(w)\mathrm{d}w + (1 - w_i')\left(1 - p\left(w_j\right)\right) + \int_0^{w_j} p(w)\mathrm{d}w \right) \\
\text{s.t. } \forall w_{i'}, w_j, r \leq &\sum_{i=0}^{\lfloor w_{i'} n \rfloor} p_i * \frac{1}{n} + \sum_{i=0}^{\lfloor w_j n \rfloor} p_i * \frac{1}{n} + (1 - w_{i'}) * (1 - p_{\lceil w_j n \rceil}) \\
&p_n = 1 \\
&p_i \leq p_{i+1} \\
&p_i \geq 0
\end{aligned}$$

```
Academic license - for non-commercial use only
Gurobi Optimizer version 9.0.1 build v9.0.1rc0 (win64)
Optimize a model with 9006002 rows, 6003 columns and 36009002 nonzeros
Model fingerprint: 0x16650c29
Coefficient statistics:
  Matrix range     [3e-04, 2e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e-04, 1e+00]

Concurrent LP optimizer: dual simplex and barrier
Showing barrier log only...

Presolve removed 3 rows and 3002 columns (presolve time = 13s) ...
Presolve removed 4 rows and 3003 columns (presolve time = 16s) ...
Presolve removed 4 rows and 3003 columns

Solved with dual simplex
Solved in 371930 iterations and 277.68 seconds
Optimal objective  5.853843871e-01

User-callback calls 150163, time in user-callback 0.03 sec
Objective value:0.5853843870879464
```

Figure 2.1: achieve ratio of 0.58538 when n = 3000



References

[1] Zhiyi Huang Thorben Tröbst. Online matching. URL: `https://ics.uci.edu/~ttrbst/online_matching_chapter.pdf`.