CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2024**

Lecture 17

November 7, 2024

Instructor: Sepehr Assadi

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

Topics of this Lecture

1 Multiplicative Weight Update for the Matching Linear Program

In the last lecture, we studied several MWU-type algorithms and saw how to analyze them. We will now use the same set of ideas to design algorithms for solving (some) LPs *approximately*. The approach we use in this section (at least in its first part) is quite general and can be applied to many other LP (and "LP-type") problems. However, to keep things concrete and for providing more intuition, we will focus on the *maximum matching* LP that we have worked on a lot with already in this course.

The LP Relaxation of Matching. For any graph $G = (V, E)$, we can write the following LP as a relaxation for the maximum matching problem in G:

$$\begin{aligned} \max_{x \in \mathbb{R}^E} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{e \ni v} x_e \leqslant 1 \quad \forall v \in V \\ & x_e \geqslant 0 \quad \forall e \in E. \end{aligned} \tag{1}$$

Recall that in Lecture 8, we showed that this LP relaxation for *bipartite* graphs have integrality gap, i.e., we can round any fractional solution to this LP to an integral one without decreasing its size. The LP is well-defined for all graphs (not only bipartite ones), although for general graphs, its integrality gap is 2/3 (think of a triangle). For now, we only focus on solving this LP near-optimally and ignore any rounding aspects; hence, we will just work with general graphs.

1.1 The Setup and the Oracle LP

The idea behind the MWU algorithm for solving any LP is to repeatedly reduce the problem to solving a much simpler LP, often called the *oracle LP*. The oracle LP is obtained from the original LP by partitioning the constraints of the LP into "easy" and "hard" ones, and then focusing only on solving a *convex combination* of the "hard" constraints instead of all of them. This may sound too abstract and vague, so let us try to clarify this further.

Suppose we have some weights $w_v > 0$ on each vertex $v \in V$. Define $W := \sum_{v \in V} w_v$. Consider the following simpler LP called the **oracle LP** with weights $\{w_v\}_{v \in V}$:

$$\begin{aligned} \max_{x \in \mathbb{R}^E} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{v \in V} w_v \sum_{e \ni v} x_e \leqslant W \\ & x_e \geqslant 0 \quad \forall e \in E. \end{aligned} \tag{2}$$

Notice that this LP is simpler than the original LP in that instead of the n "hard" constraints of the original LP—one for each vertex of the graph—we now only have a single constraint for their convex combination.

Observation 1. *For any choice of weights $w_v \geqslant 0$ for $v \in V$, the optimal objective value of LP (2) is at least as large as that of LP (1).*

Proof. Any feasible solution x to LP (1) satisfies the following equation for every $v \in V$:

$$\sum_{e \ni v} x_e \leqslant 1$$

Thus if we multiply each side by $w_v \geqslant 0$, we will still get

$$w_v \cdot \sum_{e \ni v} x_e \leqslant w_v;$$

and, if we further sum up both sides on all vertices $v \in V$, we get

$$\sum_{v} w_v \cdot \sum_{e \ni v} x_e \leqslant \sum_{v} w_v = W.$$

Thus, any feasible solution to LP (1) is also a feasible solution to LP (2). As both LPs are maximization LPs with the same objective, we obtain the result. ☐

Moreover, solving this LP is also much simpler than the original one. For every edge $e = (u, v) \in E$, define the *artificial weight* of the edge e to be

$$aw(e) := w_u + w_v.$$

We have,

$$\sum_{v \in V} w_v \cdot \sum_{e \ni v} x_e := \sum_{e=(u,v) \in E} x_e \cdot (w_u + w_v) = \sum_{e \in E} aw(e) \cdot x_e.$$

Hence, the single main constraint of LP (2) is equivalent to:

$$\sum_{e \in E} aw(e) \cdot x_e \leqslant W. \tag{3}$$

But then it means that to obtain the optimal solution to LP (2), we simply need to find

$$e^* := \arg\min_{e \in E} \; aw(e)$$

and then set

$$x_{e^*} := \frac{W}{aw(e^*)} \qquad \text{and} \qquad x_e = 0 \quad \text{for all other } e \neq e^*. \tag{4}$$

This clearly maximizes $\sum_{e \in E} x_e$ (because "moving" any value from x_{e^*} on any other x_e can only make the constraint violated without helping with the objective value). We summarize this as follows.

Observation 2. *The optimal solution to LP (2) is given by Eq (4).*

1.2 The Algorithm

So now how can we use this oracle LP for solving the original LP? Notice that if we put a *large* weight on a vertex, we will "move" the optimum solution of the oracle LP away from that vertex; thus, by adjusting the weights of the constraints, we can make them more important or reduce their importance, hence getting the oracle LP to "focus" on different parts of the graph. Then, by taking the *average* of all the solutions we found, we will hopefully get a solution which is handling every constraint of LP (1) (approximately). We show how we can update the weights using MWU to achieve this goal.

The algorithm is formally as follows (the algorithm has two parameters $\eta > 0$ and $T \geqslant 1$ that will be determined later).

Algorithm 1. A MWU algorithm for the Matching LP.

1. For every vertex $v \in V$, let $w_v^{(1)} = 1$.
2. For $t = 1$ to T iterations:
   (a) Let $x^{(t)}$ be the optimal solution to the oracle LP with weights $w^{(t)}$ according to Eq (4).
   (b) For every vertex $v \in V$, update:
$$w_v^{(t+1)} = \left(1 + \eta \cdot \sum_{e \ni v} x_e^{(t)}\right) \cdot w_v^{(t)}.$$
3. Return the final solution
$$\bar{x} := \frac{1}{T} \cdot \sum_{t=1}^{T} x^{(t)}.$$

A note that the update rule basically means in each iteration the two vertices u, v incident on the optimal edge of Eq (4) are having their weight increased (somewhat proportional to the "violation" of their matching constraint in the original LP) and all other vertices retain their original weight. However, it will be easier to work with the given formulation both for the current analysis, as well as future improvements.

Let $\varepsilon > 0$ be a parameter and recall that our goal is to obtain a $(1+\varepsilon)$-approximation to LP (1) via Algorithm 1. The choice of ε will play a role in determining the values of η and T. The following is the main lemma for this algorithm.

Lemma 3. *For a proper choice of η and T, the output $\bar{x}$ of Algorithm 1 satisfies the following:*

$$\sum_{e \in E} \bar{x}_e \geqslant \mathsf{OPT} \tag{5}$$

$$\sum_{e \in v} \bar{x}_e \leqslant 1 + \varepsilon \qquad \text{for all } v \in V \tag{6}$$

where OPT is the optimum value of LP (1).

Notice that the statement of this lemma does *not* imply $\bar{x}$ is feasible for LP (1), even though its value is as good as the optimal one. However, we can simply rescale $\bar{x}$ with $(1+\varepsilon)$, i.e., consider $\bar{x}/(1+\varepsilon)$ as our solution which is now feasible and a $(1+\varepsilon)$-approximation to LP (1)

1.3 The Analysis

It thus remains to prove Lemma 3 to finalize the whole proof.

Proof of Eq (5) for all choices of η, T. We start with the easy part first which is to prove Eq (5). By Observation 1, we have that for every $t \geqslant 1$, the optimum solution $x^{(t)}$ satisfies

$$\sum_{e \in E} x_e^{(t)} \geqslant \mathsf{OPT}.$$

Thus, the average solution $\bar{x}$ also satisfies this equation, hence proving Eq (5) no matter what is the choice of η and T.

Proof of Eq (6) for proper choices of η, T. We now prove the main part. We follow a similar *potential function* argument as the one for the expert problem in the previous lecture. Our potential function is again $W^{(t)} := \sum_{v \in V} w_v^{(t)}$, namely, the total weights of the vertices. The first part is to show that the potential does not increase too much.

Claim 4. *For every choice of $T \geqslant 1$, we have that at the end of the last iteration*

$$W^{(T+1)} \leqslant \exp\left(\eta \cdot T + \ln n\right).$$

Proof. For every $t \geqslant 1$,

$$\begin{aligned}
W^{(t+1)} &= \sum_{v \in V} w_v^{(t+1)} && \text{(by the definition of } W^{(t+1)}\text{)} \\
&= \sum_{v \in V} 1 + \eta \cdot \sum_{e \ni v} x_e^{(t)} \cdot w_v^{(t)} && \text{(by the update rule of the algorithm)} \\
&= \sum_{v \in V} w_v^{(t)} + \eta \cdot \left(\sum_{v \in V} w_v^{(t)} \cdot \sum_{e \ni v} x_e^{(t)} \right) && \text{(by rearranging the terms)} \\
&= W^{(t)} + \eta \cdot \left(\sum_{v \in V} w_v^{(t)} \cdot \sum_{e \ni v} x_e^{(t)} \right) && \text{(by the definition of } W^{(t)}\text{)} \\
&\leqslant W^{(t)} + \eta \cdot W^{(t)} && \text{(by the feasibility of } x^{(t)} \text{ in the oracle LP (2))} \\
&= (1+\eta) \cdot W^{(t)}.
\end{aligned}$$

As a result, and since $W^{(1)} = n$, we obtain that after the T-th iteration:

$$W^{(T+1)} \leqslant (1+\eta)^T \cdot n \leqslant \exp\left(\eta \cdot T\right) \cdot n = \exp\left(\eta \cdot T + \ln n\right),$$

where we used the inequality $1 + x \leqslant e^x$ for all $x > 0$. □

The second part is to show that if there is a vertex v that even at the end of the T iterations does not satisfy Eq (6), its weight should have increased by a lot. Specifically, we have a vertex $v \in V$ such that after T iterations, it still does not satisfy Eq (6), i.e.,

$$\sum_{e \ni v} \bar{x}_e > (1 + \varepsilon).$$

This equivalently means that

$$\sum_{t=1}^{T} \sum_{e \ni v} x_e^{(t)} > (1+\varepsilon) \cdot T. \tag{7}$$

Claim 5. *For every choice of $T \geqslant 1$ if a vertex v satisfies Eq (7) at the end of the last iteration, then,*

$$w_v^{(T+1)} \geqslant \exp\left(\eta \cdot (1 + \frac{\varepsilon}{4}) \cdot T\right),$$

as long as $\varepsilon < 1/2$ and $\eta \leqslant \varepsilon / 2\rho_v$ for ρ_v defined as:

$$\rho_v := \max_{t \geqslant 1} \sum_{e \ni v} x_e^{(t)}.$$

Proof. We have,

$$
\begin{aligned}
w_v^{(T+1)} &= \prod_{t=1}^{T}\left(1+\eta\cdot\sum_{e\ni v}x_e^{(t)}\right) && \text{(by the update rule and since } w_v^{(1)}=1\text{)}\\
&\geqslant \prod_{t=1}^{T}\exp\left(\eta\cdot\sum_{e\ni v}x_e^{(t)}-\eta^2\cdot\left(\sum_{e\ni v}x_e^{(t)}\right)^2\right),
\end{aligned}
$$

as long as $\eta < 1/\rho_v$ because for such choice of η, we can apply the inequality $1+y \geqslant e^{y-y^2}$ which holds for $y \in (0,1)$. Continuing the above equations, we get,

$$
\begin{aligned}
w_v^{(T+1)} &\geqslant \exp\left(\eta\cdot\sum_{t=1}^{T}\sum_{e\ni v}x_e^{(t)}-\eta^2\cdot\sum_{t=1}^{T}\left(\sum_{e\ni v}x_e^{(t)}\right)^2\right) && \text{(by using } e^{\alpha}\cdot e^{\beta}=e^{\alpha+\beta}\text{)}\\
&\geqslant \exp\left(\eta\cdot\sum_{t=1}^{T}\sum_{e\ni v}x_e^{(t)}-\eta^2\cdot\rho_v\cdot\sum_{t=1}^{T}\sum_{e\ni v}x_e^{(t)}\right) && \text{(by replacing } \textstyle\sum_{e\ni v}x_e^{(t)}\leqslant\rho_v\text{)}\\
&\geqslant \exp\left(\eta\cdot\sum_{t=1}^{T}\sum_{e\ni v}x_e^{(t)}-\frac{\varepsilon}{2}\cdot\eta\cdot\sum_{t=1}^{T}\sum_{e\ni v}x_e^{(t)}\right) && \text{(by the assumption on the value of } \eta\text{)}\\
&= \exp\left(\eta\cdot(1-\frac{\varepsilon}{2})\cdot\sum_{t=1}^{T}\sum_{e\ni v}x_e^{(t)}\right)\\
&\geqslant \exp\left(\eta\cdot(1-\frac{\varepsilon}{2})\cdot(1+\varepsilon)\cdot T\right) && \text{(by Eq (7))}\\
&\geqslant \exp\left(\eta\cdot(1+\frac{\varepsilon}{4})\cdot T\right), && \text{(as } \varepsilon<1/2\text{)}
\end{aligned}
$$

concluding the proof. □

Since all the weights are positive, for any vertex $v \in V$, as long as we pick $\eta < \varepsilon/2\rho_v$ as prescribed by Claim 5, we obtain that for v to satisfy Eq (7) we need to have,

$$\exp\left(\eta\cdot(1+\frac{\varepsilon}{4})\cdot T\right) \leqslant w_v^{(T+1)} \leqslant W^{(T+1)} \leqslant \exp\left(\eta\cdot T+\ln n\right),$$

where the LHS inequality is by Claim 5 and the RHS inequality is by Claim 4. This implies that

$$\eta\cdot(1+\frac{\varepsilon}{4})\cdot T \leqslant \eta\cdot T+\ln n,$$

which in turn means

$$T \leqslant \frac{4\ln n}{\eta\cdot\varepsilon}.$$

Thus, after these many iterations, all vertices violated Eq (7) and thus in turn satisfy Eq (6).

We state all these—for future reference—in the following lemma (whose proof already appeared above).

Lemma 6. *For any $\varepsilon \in (0,1/2)$, if we run Algorithm 1 for $T > \frac{8\rho\cdot\ln n}{\varepsilon^2}$ iterations for*

$$\rho \geqslant \max_{v\in V}\max_{t\geqslant 1}\sum_{e\ni v}x_e^{(t)},$$

with parameter $\eta = \frac{\varepsilon}{2\rho}$, then the output solution $\bar{x}$ satisfies Eq (6).

Finally, we emphasize that in the proof of Lemma 6, we never once used the fact that each $x^{(t)}$ returned by the algorithm is *optimal* for the oracle LP (2), only that it is *feasible* – this is the exact opposite of the case when proving Eq (5).

1.4 Runtime Analysis

Unfortunately, we are still not done entirely, because it is not clear how to pick the parameter ρ needed in Lemma 6. Notice that ρ is basically the *largest* value we will ever assign to any edge e in any $x_e^{(t)}$ in any of the iterations (technically, it is the largest value of $\sum_{e \ni v} x_e^{(t)}$ but since the algorithm only assigns value to a single edge in each iteration, these two are the same). But how can we bound this quantity?

The key observation is that we do *not* need to solve each oracle LP (2) optimally! We only need to solve it as good the value of the *original* LP (1); this is because, even in that case, we can still prove Eq (5) exactly as it is (and lower bound the value of each $x^{(t)}$ by OPT still, which is what we did anyway), and Eq (6) does not have anything to do with the optimality of $x^{(t)}$, only its feasibility.

But now, we do know that $\mathsf{OPT} \leqslant n$ always because it is optimal solution to a matching LP. This implies that in Line (2a) of Algorithm 1, we can in fact replace $x^{(t)}$ computed in Eq (4) by the following ($e^* := \arg\min_{e \in E} \; aw(e)$ is as before):

$$x_{e^*} := \min\left(n, \frac{W}{aw(e^*)}\right) \qquad \text{and} \qquad x_e = 0 \quad \text{for all other } e \neq e^*,$$

i.e., we are now taking the minimum of the previously chosen value and $n \geqslant \mathsf{OPT}$. This ensures that every $x^{(t)}$ still have value at least OPT while also continue to be feasible. Thus, we obtain both Eq (5) and Eq (6) as before with no other changes to the proof.

The benefit of this is that we now can say $\rho \leqslant n$ because we will never assign a larger value in any iteration of the algorithm. This allows us to pick a choice of $\eta \leqslant \varepsilon/2n$ as well which gives us an actual algorithm (with known parameters throughout).

Finally, the number of iterations of this algorithm is

$$O(\frac{n \ln n}{\varepsilon^2})$$

by Lemma 6. Each iteration also requires us finding the edge with *minimum* artificial weight $aw(e)$ (based on the weight function $w^{(t)}$). It is easy to see that each update only changes the weight of $O(n)$ other edges and thus we can maintain the edges in a min-heap and run each iteration of the algorithm in $O(n \log n)$ time or a Fibonacci heap and $O(n)$ time (in fact, here simpler solutions with $O(n)$ time also exist). Thus, the total runtime of the algorithm is

$$O\left(\frac{n^2}{\varepsilon^2} \cdot \log n\right).$$

Almost everything we discussed in this lecture can be readily generalized to solving other LPs if our goal is to find an approximately feasible solution (although not necessary a feasible one with approximate objective). In the next lecture, we see how using problem-dependent ideas (in our case, for the maximum matching problem) we can further optimize this runtime and also reduce the number of iterations of the MWU algorithm dramatically.

Finally, as a historical note, we mention that the use of MWU for approximating LPs was introduced first in the work of [PST91].

References

[PST91] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 495–504. IEEE Computer Society, 1991. 6