CS 521: Linear Programming **Rutgers: Fall 2022**

Lecture 6

October 14, 2022

Instructor: Sepehr Assadi Scribes: Jonathan Garcia, Denson George, Richard Magnotti, Hoang Minh

Topics of this Lecture

1 Packing and Covering Linear Programs

In this lecture, we focus on solving a special family of LPs referred to as **packing** and **covering** LPs.

Definition 1. Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ be vectors of non-negative entries. Then, the following primal-dual LP pairs are called a **packing** LP and a **covering** LP, respectively:

Packing LP:

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{subject to} \quad & Ax \leqslant b \\ & x \geqslant 0 \end{aligned}$$

Covering LP:

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & b^\top y \\ \text{subject to} \quad & A^\top y \geqslant c \\ & y \geqslant 0 \end{aligned}$$

Notice that the only difference of the packing/covering LPs with the general formulation of the problem is that we now require all the constants and variables to be non-negative. This simple change however makes

these problems rather considerably easier than solving general LPs. For instance, any packing LP always has the all-zero vector $0 \in \mathbb{R}^n$ as a feasible solution, and any covering LP always admit a feasible solution by picking all coordinates of $y \in \mathbb{R}^m$ sufficiently large with respect to the numbers in c and A.

A few examples of the packing problems include previously seen problems such as maximum matching, knapsack, and maximum flow problem. Examples of covering problems include are vertex cover, set cover, and minimum cut. Notice that dual of packing problems are covering problems and vice versa.

Remark. The algorithmic technique, namely, the multiplicative weight update method, that we study in this lecture applies "best" to packing and covering programs, hence, our choice of focusing on these LPs in this lecture. However, with more work, one can use these ideas to *any* other linear program as well, and thus there is no need in general to limit oneself to packing/covering LPs when working with this technique.

2 Multiplicative Weight Update (MWU)

In this lecture, we review another simple yet highly useful technique for solving linear programs, namely, the *multiplicative weight update (MWU)* method. Before mentioning this method, we should emphasize that MWU is not a single algorithm, but rather a family of algorithms, and in fact more broadly an algorithmic technique (similar in spirit to, say, greedy or dynamic programming techniques). In the following, we focus on presenting this technique in its simplest form and along the way mention several alternative design decisions that one can make when using this technique.

We focus on solving the following packing LP presented in the following form[1]:

$$\begin{aligned} \max \quad & c^\top x \\ \text{subject to} \quad & Ax \leqslant b \\ & x \geqslant 0. \end{aligned}$$

In the rest of this section, we go over several concepts related to MWU.

2.1 Feasibility LP

MWU is actually an algorithm for solving the *feasibility* problem and not the optimization one. Thus, using our standard binary search technique, we first reduce the problem to checking the feasibility of the following LP instead for any given parameter $F \in \mathbb{R}$:

$$\begin{aligned} &\text{Polyhedron (P):} \\ c^\top x &\geqslant F \\ Ax &\leqslant b \\ x &\geqslant 0. \end{aligned}$$

Note that at this point, (P) is no longer a packing LP strictly speaking. As we will see in the next part, MWU solves this feasibility problem "approximately".

2.2 Easy and Hard Constraints

The next step is to partition the constraints into *"easy"* constraints and *"hard"* constraints: the easy constraints are the ones that we think we can easily satisfy "on our own", while the hard constraints are the ones we need the help of the MWU to satisfy[2]. This is the first example of a design question: which constraints to consider easy? For now, we pick one of the simplest choices for this:

[1] The same idea, almost as is, also applies to covering LPs.

[2] This is quite informal and vague on purpose and will become clear later in this lecture.

Easy constraints (E):

$$c^\top x \geqslant F$$
$$x \geqslant 0.$$

By this, we mean that we can easily find an $x \in E$: of course, this is trivial at this point, but in the next step, we should be able to satisfy this constraint plus just one more constraint.

Error term. MWU is an approximation algorithm and thus returns a solution which is "approximately feasible" for (P). More specifically, given a parameter $\varepsilon > 0$, it either return a vector $\tilde{x}$ such that

$$\tilde{x} \in E$$
$$A\tilde{x} \leqslant b + \varepsilon \cdot I_m,$$

where I_m is the identy $m \times m$ matrix. In other words, the final solution satisfies that for every constraint $i \in [m]$, $\langle a_i \, , \, x \rangle \leqslant b_i + \varepsilon$, so the hard constraints are satisfied approximately. In this case, we say that $\tilde{x}$ is *ε-approximately* in (P).

Remark. When working with packing and covering LPs, one can "translate" a point $\tilde{x}$ which is ε-approximately feasible for the LP, into a point which is feasible for the LP but has a weaker objective value to a factor which is determined by ε. The general idea is to simply use the non-negativity of all the constants in the program, to "scale" the vector $\tilde{x}$ by an appropriate amount to satisfy all the constraints, while weakening the objective by the same factor. This is one of the reasons that makes MWU quite suitable for solving packing and covering LPs. We will see this in more details in Section 4.

2.3 MWU Oracle

The next and main step is to solve the following *oracle* problem: Given *non-negative* weights $w_1, \ldots, w_m$ for the m hard constraints $a_i^\top x \leqslant b_i$ that remain, find an $x \in E$ such that x satisfies the *average* hard constraint (averaged by weights w). I.e., find an x in the following polyhedron:

Oracle polyhedron (O) with non-negative weights $(w_1, \ldots, w_m)$:

$$\sum_{i=1}^{m} w_i \cdot (a_i^\top x) \leqslant \sum_{i=1}^{m} w_i \cdot b_i$$
$$x \in E.$$

Notice that the oracle polyhedron (O) only has *one* constraint over the "trivially feasible" polyhedron (E), so we hope finding a feasible point for it would be much simpler than the original problem. The following claim further formalizes this:

Claim 2. *Any feasible $x \in P$ is also feasible for the oracle polyhedron (O) with any given non-negative weights $(w_1, \ldots, w_m)$.*

Proof. The solution x clearly satisfies $c^\top x \geqslant F$ and $x \geqslant 0$, thus $x \in E$. Moreover, it satisfies that for every $i \in [m]$, we have $a_i^\top x \leqslant b_i$. Given that all weights $(w_1, \ldots, w_m)$ are non-negative, this implies that

$$\sum_{i=1}^{m} w_i \cdot (a_i^\top x) \leqslant \sum_{i=1}^{m} w_i \cdot b_i;$$

thus, x is also feasible for the oracle polyhedron. □

Width parameters of the oracle. For any feasible solution x in (O), we define:

$$\rho(x) := \max_{i\in[m]} (a_i^\top x) - b_i,$$
$$\ell(x) := \min_{i\in[m]} (a_i^\top x) - b_i.$$

Thus, we have for every $i \in [m]$, $(a_i^\top x) - b_i \in [\ell(x), \rho(x)]$. In words, $\rho(x)$ is the "largest violation" we get on a constraint when we plug x inside our original LP and $\ell(x)$ is the "lowest underutilization" of a constraint in the original LP under x. We refer to $\rho(x)$ and $\ell(x)$ as the **width parameters** collectively. The role of these parameters will become quite clear once we define the MWU algorithm formally, but for now, we only mention that our goal is almost always to *minimize* the absolute value of $\rho(x)$ and $\ell(x)$.

The *main* design question when working with MWU is to design an algorithm for finding a point x in the oracle polyhedron (O) with small values of $|\rho(x)|, |\ell(x)|$. Again, let us show one of the simplest choices.

Claim 3. *Suppose (P) is feasible. Then, there exists some $j \in [n]$, such that the solution*

$$y := [\underbrace{0, \ldots, 0}_{j-1}, \frac{F}{c_j}, \underbrace{0, \ldots, 0}_{n-j}]$$

is feasible for (O). In words, one of the n "singleton" tight vectors in (E) is also feasible for (O).

Proof. Let $x \in P$ and by Claim 2, we also have $x \in O$. In particular,

$$\sum_{i=1}^{m} w_i \cdot (a_i^\top x) = \sum_{j=1}^{n} x_j \cdot \sum_{i=1}^{m} a_{ij} \cdot w_i \leqslant \sum_{i=1}^{m} w_i \cdot b_i := B.$$

Define $d_j := \sum_{i=1}^{m} a_{ij} \cdot w_i$ for every $j \in [n]$. Consider the optimization problem (Π):

$$\begin{aligned} \min_{y\in\mathbb{R}^n} \quad & \sum_{j=1}^{n} y_j \cdot d_j \\ \text{subject to} \quad & \sum_{j=1}^{n} y_j \cdot c_j \geqslant F \\ & y \geqslant 0. \end{aligned}$$

Let $j^* := \arg\max_{j\in[n]}(d_j/c_j)$. We claim that the vector y where $y_j = F/c_j$ for $j = j^*$ and $y_j = 0$ for $j \neq j^*$ is an optimum solution (Π). This will conclude the claim since given that the vector x was a feasible solution to (Π), we will obtain that $\sum_{j=1}^{n} y_j \cdot d_j \leqslant B$, implying that y is a feasible solution in (O) as well, as desired.

We now prove the optimality of y for (Π). While this can be proven using a simple greedy algorithm and an exchange argument, let us prove this using a duality argument in the spirit of this course. Firstly, y is trivially a feasible solution to (Π), thus we only need to prove its optimality. As (Π) itself is a covering LP, we can write its dual with only a single variable as the following packing LP:

$$\begin{aligned} \max_{z\in\mathbb{R}} \quad & F \cdot z \\ \text{subject to} \quad & c_j \cdot z \leqslant d_j \qquad \forall j \in [n] \\ & z \geqslant 0. \end{aligned}$$

The optimal solution of this single-variate LP happens on the point $z = \min_{j\in[n]} d_j/c_j$, resulting on the value of the LP becoming $F \cdot d_j/c_j$. Given that this value is equal to the objective value of y in (Π), by weak duality, we obtain that y is also optimal, concluding the proof. □

While Claim 3 gives us a way of picking a feasible point in the oracle polyhedron, this approach is typically quite inefficient given the large width parameters it will induce (more on this in Section 4).

2.4 The Algorithm

We are now ready to present the MWU algorithm.

The MWU Algorithm. Given a feasibility LP (P) and parameter $\varepsilon > 0$, returns a point $\tilde{x}$ which is ε-approximately in polyhedron (P).

The algorithm uses two parameters $\eta \in [0, 1/2)$ and $T \in \mathbb{N}$ to be determined later.

(*i*) Let $w_i^{(1)} = 1$ for every $i \in [m]$[a] (here and throughout the algorithm, the subscript goes over constraints of (P) and the superscript denotes the iteration).

(*ii*) For $t = 1$ to T iterations:

(a) Solve the oracle LP (O) with weights $(w_1^{(t)}, \ldots, w_m^{(t)})$ to obtain a vector $x^{(t)} \in \mathbb{R}^n$. If the oracle LP is infeasible, output (P) is also infeasible.

(b) For $i \in [m]$, let $w_i^{(t+1)} = w_i^{(t)} \cdot \left(1 + \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right)$, for $\rho\ell(x^{(t)}) := \max\left(|\rho(x^{(t)})|, |\ell(x^{(t)})|\right)$.

(*iii*) Return $\tilde{x} := \frac{1}{T} \cdot \sum_{t=1}^{T} x^{(t)}$.

[a] This is yet another design choice since in certain cases, it also makes sense to start with non-uniform weights.

Let us parse the MWU algorithm now. It starts by putting uniform weights over the constraints, namely, "treating them equally", and obtain a solution $x^{(1)}$ based on that, namely, a solution that satisfies an "average" hard constraint (averaged by the weights in the iteration). The algorithm then adjusts the weights over the constraints: if a constraint $i \in [m]$ is violated by $x^{(1)}$, it increases its weight and if the constraint is underutilized, it decreases its weight. This has the following natural effect; in the next iteration that we are running the oracle LP, the constraints previously violated will be given "more priority", in that the oracle now has to pay more attention to satisfying them; the constraint which are underutilized get less weight and thus "less priority" and so the oracle will be paying less attention to satisfying them.

This process then continues by focusing on different constraints in each iteration through these reweighing. At the end, the algorithm returns the *average* of all solutions found which makes sense: in different iterations, the algorithm attempts to satisfy different constraints and thus "on average", we hope all the constraints to be satisfied. The idea behind the analysis is then pretty simple[3] (and quite similar to that of the Clarkson's algorithm in Lecture 5 actually). Basically, by making sure the oracle satisfies the average constraint in each iteration, we can make sure that the total weights of the constraints does not increase throughout the algorithm. On the other hand, if at the end we have a constraint that we have not satisfied on average even approximately, we would have that the weight of this constraint should grow quite a lot during the algorithm (by our weight update rule). By picking the number of iterations sufficiently large, we will end up having increased the weight of this constraint so much that it would become more than our original bound on the total weights; combined with the previous part this leads to a contradiction, which concludes the proof.

We now formalize this intuition and specify the last remaining details in the algorithm.

Theorem 4. *Consider any feasibility LP (P) and the MWU algorithm with a given parameter $\varepsilon \in (0, 1)$ for solving it. Let*

$$\rho \geqslant \max_{t \in [T]} |\rho(x^{(t)})| \qquad \ell \geqslant \max_{t \in [T]} |\ell(x^{(t)})|;$$

that is, upper bounds width parameters that we encounter throughout the algorithm. Furthermore, assume $1 \leqslant \ell \leqslant \rho$ and let

$$\eta := \frac{\varepsilon}{4 \cdot \ell} \qquad T = \frac{8 \cdot \ell \cdot \rho \cdot \ln m}{\varepsilon^2}.$$

[3] Although the technical details are rather messy on the surface. After all, the devil is in the detail.

Then, the MWU algorithm either correctly outputs that (P) is infeasible or outputs a vector $\tilde{x}$ which is ε-approximately in (P).

We prove this theorem in the next section. Before that, we have the following remark.

Remark. Given any feasibility packing LP with entries bounded by $U \geqslant 1$, we can use Claim 3 to obtain a trivial algorithm for the oracle LP with width parameters $\rho = U \cdot F$ and $\ell = U$ that runs in $O(m \cdot n)$ time. Plugging in this in Theorem 4, we see that the MWU algorithm requires at most $O(\varepsilon^{-2} \cdot U^2 \cdot F \ln m)$ iterations to compute a vector ε-approximately in the LP. Considering the book keeping and everything else, each iterations of MWU can be implemented in $O(m \cdot n)$, leading to an algorithm with $O(\varepsilon^{-2} \cdot mn \cdot U^2 \cdot F \cdot \ln m)$ time for solving any feasibility packing LP.

By using MWU more carefully and exploiting further properties of the underlying packing LP we would like to solve, we can dramatically improve the runtime of this algorithm. This the content of Section 4 at the end of this lecture.

3 Proof of Correctness of MWU (Theorem 4)

We now prove Theorem 4. We will use the following set of inequalities throughout the proof.

Fact 5. *For $\eta < 1/2$, we have,*

$$\begin{array}{lcll} \ln(1+\eta) & \geqslant & \eta - \eta^2 & \\ \ln(1-\eta) & \geqslant & -\eta - \eta^2 & \\ 1+\eta x & \geqslant & (1+\eta)^x & \forall x \in [0,1] \\ 1+\eta x & \geqslant & (1-\eta)^{-x} & \forall x \in [-1,0]. \end{array} \tag{1}$$

In proving Theorem 4, the fact that the MWU algorithm outputs (P) is infeasible, only if it is truly infeasible follows from Claim 2 – as long as (P) is feasible, the oracle LP (O) will also be feasible no matter the choice of weights. The main part is in proving that if the algorithm does not terminate early, it outputs an ε-approximate solution.

We first show that the total weight of the constraints is non-increasing during the algorithm, simply because we are attempting to solve the "average" constraint in the oracle LP, and thus on average, the weights should not increase.

Lemma 6. *For any iteration $t \geqslant 0$, define $W^{(t)} := \sum_{i=1}^{m} w_i^{(t)}$. Then, $W^{(t+1)} \leqslant W^{(t)}$.*

Proof. We have,

$$\begin{aligned} W^{(t+1)} &= \sum_{i=1}^{m} w_i^{(t)} \cdot \left(1 + \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right) && \text{(by the update rule of the MWU algorithm)} \\ &= W^{(t)} + \frac{\eta}{\rho \ell(x^{(t)})} \cdot \left(\sum_{i=1}^{m} w_i^{(t)} \cdot (a_i^\top x^{(t)}) - \sum_{i=1}^{m} w_i^{(t)} \cdot b_i\right) && \text{(by the definition of } W^{(t)}) \\ &\leqslant W^{(t)}, \end{aligned}$$

since $x^{(t)} \in O$ and thus the second term of the RHS is non-positive. □

We now argue that in order for $\tilde{x}$ to violate a constraint "badly" at the end, the weight of that constraint should have grown quite a lot in the algorithm, to the extend that it becomes larger than $W^{(1)} = m$, contradicting Lemma 6. To do so, we first argue that a badly violated constraints results in many "positive updates" to the weight.

Claim 7. *For any $i \in [m]$, if $a_i^\top \cdot \tilde{x} > b_i + \varepsilon$, then $\sum_{t=1}^{T}(a_i^\top x^{(t)} - b_i) \geqslant \varepsilon \cdot T$.*

Proof. By the definition of $\tilde{x}$, we have,

$$a_i^\top \cdot \tilde{x} - b_i - \varepsilon = \frac{1}{T} \cdot (\sum_{t=1}^{T} a_i^\top x^{(t)} - b_i - \varepsilon).$$

Since the LHS is at least 0 by the lemma statement, we have,

$$\sum_{t=1}^{T}(a_i^\top x^{(t)} - b_i) \geqslant \varepsilon \cdot T,$$

concluding the proof. □

Remark. Before we continue, let us show a *wrong* but intuitive way of concluding the proof. Towards a contradiction, fix any $i \in [m]$ such that $a_i^\top \cdot \tilde{x} > b_i + \varepsilon$. By the update rule of the MWU and Claim 7,

$$w_i^{(T+1)} = \prod_{t=1}^{T}\left(1 + \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right) \approx \exp\left(\sum_{t=1}^{T} \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right) \geqslant \exp\left(\eta \cdot \frac{\varepsilon \cdot T}{\rho}\right)$$

which is larger than m for the choice of η and T in Theorem 4. This then contradicts Lemma 6 and concludes the proof.

The problem of course is in the "approximation" step we used for approximating $(1 + z) \approx \exp(z)$, which is in the *wrong direction* in this scenario. As a result, we have to work with the *second order* approximation of these parameters that makes the proof more challenging (and somewhat tedious)[a], even though the idea is exactly the same still.

[a] But also leading to the right dependence for T as opposed to what is implied by the wrong calculation above.

We are now going to prove a lower bound on the final weight of any (sufficiently) violating constraint.

Lemma 8. *For any $i \in [m]$, if $a_i^\top \cdot \tilde{x} > b_i + \varepsilon$, then $w^{(T+1)} \geqslant m$.*

Proof. Let $N \subseteq [T]$ be the iterations wherein $a_i^\top \cdot \tilde{x} < b_i$ and $\bar{N} := [T] \setminus N$ be the remaining iterations. Then,

$$\begin{aligned}
w_i^{(T+1)} &= \prod_{t=1}^{T}\left(1 + \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right) && \text{(by the update rule of the algorithm and since } w_i^{(1)} = 1) \\
&= \prod_{t \in N}\left(1 + \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right) \cdot \prod_{t \in \bar{N}}\left(1 + \eta \cdot \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}\right) \\
&\geqslant \prod_{t \in N}(1 - \eta)^{\frac{-(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}} \cdot \prod_{t \in \bar{N}}(1 + \eta)^{\frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}} \\
& && \text{(by the second two inequalities of Fact 5)} \\
&= (1 - \eta)^{\sum_{t \in N} \frac{-(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}} \cdot (1 + \eta)^{\sum_{t \in \bar{N}} \frac{(a_i^\top x^{(t)} - b_i)}{\rho \ell(x^{(t)})}}.
\end{aligned}$$

Let us now take $\ln(\cdot)$ of both sides of this equation to have,

$$\begin{aligned}
\ln\left(W^{(T+1)}\right) &\geqslant \sum_{t\in N} \frac{-(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \cdot \ln(1-\eta) + \sum_{t\in \bar{N}} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \cdot \ln(1+\eta) \\
&\geqslant \sum_{t\in N} \frac{-(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \cdot (-\eta-\eta^2) + \sum_{t\in \bar{N}} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \cdot (\eta-\eta^2) \\
&\qquad \text{(by the first two inequalities of Fact 5)} \\
&= \sum_{t\in N} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \cdot (\eta+\eta^2) + \sum_{t\in \bar{N}} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \cdot (\eta-\eta^2) \\
&\qquad \text{(just canceling the negatives in the first term)} \\
&= \eta \cdot \sum_{t=1}^{T} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} - \eta^2 \cdot \left(\sum_{t\in \bar{N}} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} - \sum_{t\in N} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \right) \\
&\qquad \text{(by reorganizing the terms)} \\
&= \eta \cdot \sum_{t=1}^{T} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} - \eta^2 \cdot \sum_{t=1}^{T} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} + 2\eta^2 \sum_{t\in N} \frac{(a_i^\top x^{(t)} - b_i)}{\rho\ell(x^{(t)})} \\
&\qquad \text{(by collecting all the terms in the second part, and then canceling the negative terms again)} \\
&\geqslant \frac{\eta}{2} \cdot \sum_{t=1}^{T} \frac{(a_i^\top x^{(t)} - b_i)}{\max(\rho,\ell)} + 2\eta^2 \cdot \sum_{t\in N} \frac{(a_i^\top x^{(t)} - b_i)}{\max(\rho,\ell)} \qquad \text{(as } \eta \leqslant 1/2 \text{ and by the definition of } \rho,\ell) \\
&\geqslant \frac{\eta}{2} \cdot \sum_{t=1}^{T} \frac{(a_i^\top x^{(t)} - b_i)}{\max(\rho,\ell)} - 2\eta^2 \cdot \sum_{t\in N} \frac{\ell}{\max(\rho,\ell)} \\
&\qquad \text{(by the definition of } \ell \text{ and since the second sum is only on } N) \\
&\geqslant \frac{\eta}{2} \cdot \frac{\varepsilon \cdot T}{\max(\rho,\ell)} - 2\eta^2 \cdot \frac{\ell \cdot T}{\max(\rho,\ell)} \\
&\qquad \text{(by Claim 7 for the first term and trivially for the second one)} \\
&= \frac{\varepsilon^2 \cdot T}{4 \cdot \ell \cdot \max(\rho,\ell)} - \frac{\varepsilon^2}{8\cdot\ell^2} \cdot \frac{\ell \cdot T}{\max(\rho,\ell)} \qquad \text{(by the choice of } \eta = \tfrac{\varepsilon}{4\cdot\rho} \text{ in Theorem 4)} \\
&= \frac{\varepsilon^2 \cdot T}{8 \cdot \ell \cdot \max(\rho,\ell)} \qquad \text{(by direct calculation)} \\
&= \frac{\varepsilon^2}{8 \cdot \ell \cdot \max(\rho,\ell)} \cdot \frac{8 \cdot \ell \cdot \rho \cdot \ln m}{\varepsilon^2} \qquad \text{(by the choice of } T = \tfrac{8\cdot\ell\cdot\rho\cdot\ln m}{\varepsilon^2} \text{ in Theorem 4)} \\
&= \ln m. \qquad \text{(as we assumed } \ell \leqslant \rho \text{ and thus } \max(\ell,\rho) = \rho)
\end{aligned}$$

This implies that $w^{(T+1)} \geqslant m$, as desired. □

After this (long) proof, we can conclude the proof of Theorem 4 easily.

Proof of Theorem 4. By Lemma 6, we have that at the end of the last iteration, $W^{(T+1)} \leqslant W^{(1)} = m$. On the other hand, if there is any constraint $i \in [m]$ which is not even ε-approximated, then by Lemma 8, $w_i^{(T+1)} \geqslant m$. Since $\eta < 1/2$ and thus all the weights are strictly larger than zero, we have that $W^{(T+1)} > w^{(T+1)} > m$, a contradiction. Thus, the final point $\tilde{x}$ is ε-approximately in (P), concluding the proof. □

4 Maximum Flow via MWU

We now consider an application of the MWU algorithm to solving the maximum flow problem on undirected unit-capacity graphs. Recall that in this problem, we are given an undirected graph $G = (V, E)$ and two

vertices $s, t \in V$, and the goal is to find the maximum flow from s to t without sending more than one unit of flow over each edge. Let $\mathcal{P}_{s,t}$ denote the set of all s-t paths in G. We formulate the unit-capacity maximum flow problem as the following packing LP.

$$\begin{aligned} \max_{f \in \mathbb{R}^{\mathcal{P}_{s,t}}} \quad & \sum_{p \in \mathcal{P}_{s,t}} f_p \\ \text{subject to} \quad & \sum_{p \ni e} f_p \leqslant 1 \qquad \forall e \in E \\ & f_p \geqslant 0 \qquad \forall p \in \mathcal{P}_{s,t}. \end{aligned}$$

We have seen a different formulation of the maximum flow problem in this course, but it is easy also to verify that the above formulation works.

To solve this problem using the MWU algorithm, we follow the steps in Section 2.

Step 1: Feasibility LP. We first turn this problem into a feasibility LP such that given any value $F \geqslant 0$, we want to test if the following LP is feasible:

Polytope (P) in $\mathbb{R}^{\mathcal{P}_{s,t}}$:

$$\begin{aligned} & \sum_{p \in \mathcal{P}_{s,t}} f_p \geqslant F \\ & \sum_{p \ni e} f_p \leqslant 1 \qquad \forall e \in E \\ & f_p \geqslant 0 \qquad \forall p \in \mathcal{P}_{s,t}. \end{aligned}$$

Having solved this LP (approximately), we can use binary search on F to solve the original problem.

Step 2: Easy and hard constraints. We let E be the first constraint of (P) on the total flow as well as the non-negativity constraints on each path. We are thus left with $m := |E|$ hard constraints corresponding to the capacity constraints on the edges.

Step 3: Oracle LP. Let $\{w_e \mid e \in E\}$ be the set of weights on the hard constraints. The oracle LP is now:

Oracle LP (O) in $\mathbb{R}^{\mathcal{P}_{s,t}}$:

$$\begin{aligned} & \sum_{e \in E} w_e \cdot \sum_{p \ni e} f_p \leqslant \sum_{e \in E} w_e \\ & \sum_{p \in \mathcal{P}_{s,t}} f_p \geqslant F \\ & f_p \geqslant 0 \qquad \forall p \in \mathcal{P}_{s,t}. \end{aligned}$$

As stated earlier, the main step of using MWU is to design a proper algorithm for solving the oracle LP. This is the content of the next two subsections wherein we see two different algorithms for this task.

4.1 A Basic Oracle for Maximum Flow

We use the same strategy as in Claim 3 for designing this oracle. Let us rewrite the weighted constraint of the oracle LP as:

$$\sum_{e \in E} w_e \cdot \sum_{p \ni e} f_p = \sum_{p \in \mathcal{P}_{s,t}} f_p \sum_{e \in p} w_e \leqslant \sum_{e \in E} w_e = W$$

Notice that the second term from left basically "charges" each path $p \in \mathcal{P}_{s,t}$ proportional to the total weight of the path. Thus, to satisfy the oracle LP, we can simply find the *s-t (weighted) shortest path* in G using the weights w_e on the edges $e \in E$ of G. Let p be this path. We then set $f_p = F$ and $f_{p'} = 0$ for $p' \neq p \in \mathcal{P}_{s,t}$. This is a feasible solution for the oracle LP as long as the oracle LP has a feasible solution (which is the case if the original LP (P) is feasible; see Claim 2). The reason for this to be true is exactly the same as Claim 3, because if a combination of F paths with different weights can have a total weight no more than W, then picking the shortest path F times cannot increase the total weight, and thus leads to an oracle solution.

The width parameters of this approach are

$$\rho = |\max_e \sum_{p \ni e} f_p - 1| = |F - 1| = F - 1;$$
$$\ell = |\min_e \sum_{p \ni e} f_p - 1| = |-1| = 1.$$

By Theorem 4, the MWU algorithm outputs a point $f \in \mathbb{R}^{\mathcal{P}_{s,t}}$ that is ε-approximately in (P) in $O(\varepsilon^{-2} F \ln m)$ iterations. Computing the shortest path in each step can be done using Dijkstra's algorithm in $O(m \log n)$ time. The bookkeeping in each iteration also involves updating the weights which can be done in $O(m)$ time. Thus, overall, this leads to an algorithm with $O(\varepsilon^{-2} \cdot mF \cdot \log^2 n)$ time. Thus, at the end, we obtain a flow $f \in \mathbb{R}^{\mathcal{P}_{s,t}}$ satisfying:

$$\begin{aligned} &\sum_{p \in \mathcal{P}_{s,t}} f_p = F \\ &\sum_{p \ni e} f_p \leqslant 1 + \varepsilon \qquad \forall e \in E \\ &f_p \geqslant 0 \qquad \forall p \in \mathcal{P}_{s,t}, \end{aligned}$$

namely, a flow which is ε-approximately in (P).

Finally, if we define $f' \in \mathbb{R}^{\mathcal{P}_{s,t}}$ to be $f/(1+\varepsilon)$ on every coordinate, we will have that

$$\sum_{p \ni e} f'_p = \frac{1}{1+\varepsilon} \cdot \sum_{p \ni e} f_p \leqslant \frac{1}{1+\varepsilon} \cdot (1+\varepsilon) = 1$$
$$\sum_{p \in \mathcal{P}_{s,t}} f'_p = \frac{1}{1+\varepsilon} \cdot \sum_{p \in \mathcal{P}_{s,t}} f_p = \frac{F}{1+\varepsilon} \geqslant (1-\varepsilon) \cdot F,$$

for small $\varepsilon > 0$. This implies that we can obtain an $(1-\varepsilon)$-approximation to max flow in $O(\varepsilon^{-2} \cdot mF \cdot \log^2 n)$ time. We remark that given the Ford-Fulkerson already achieves $O(mF)$ time for finding the maximum flow exactly even in directed graphs, the runtime we obtained in this step is not that interesting and only serves the purpose of giving a very simple application of the MWU method and setting up the stage for the next oracle algorithm.

4.2 A (Much) Stronger Oracle for Maximum Flow

We now present a stronger oracle for the maximum flow problem, based on the breakthrough work of Christiano, Kelner, Madry, Spielman, and Teng [1][4]. Consider the basic oracle of the previous subsection. It did not achieve a good runtime because the number of iterations of MWU was high due to having width parameter $\rho = \Omega(F)$. Our goal now is to obtain an oracle with a smaller width.

For any flow $f \in \mathbb{R}^{\mathcal{P}_{s,t}}$ and any edge $e \in E$, we define $f_e := \sum_{p \ni e} f_p$, i.e., the total flow going through the edge e. Moreover, for any weights $\{w_e \mid e \in E\}$, let $W = \sum_e w_e$ and define the set $\{q_e = w_e/W \mid e \in E\}$

[4] We only provide the simplified version with weaker guarantees; see [1] for the main result.

to be the normalized weights. With this notation, the oracle problem is:

$$\sum_{e \in E} q_e \cdot f_e \leqslant 1$$
$$\sum_{p \in \mathcal{P}_{s,t}} f_p = F$$
$$f_p \geqslant 0 \qquad \forall p \in \mathcal{P}_{s,t}.$$

The width parameters for any solution $f \in \mathbb{R}^{\mathcal{P}_{s,t}}$ to this oracle will also be

$$\rho = \max_e (f_e - 1) \qquad \ell = -1.$$

The basic approach results in a large width for this oracle because it routes all the flow through just a single path, leading to $f_e = F$ on the edges of the path. Thus, our goal will be to find a flow of value F which is "well spread", namely, does not increase the value of f_e on any edge too much. Of course, we can let f be the maximum s-t flow in G, which results in width parameters $\rho = \ell = 1$ (recall that we need $1 \leqslant \ell \leqslant \rho$), which is quite efficient. But obviously, if we could solve maximum flow already we did not need this oracle to begin with! But, we can exploit the fact that in the oracle LP, we do not need to preserve the capacity constraint exactly, rather, we only try not too violate it "too badly" so that the width remains small. We do this using the notion of *electrical flows* which we define next.

4.2.1 Detour: Electrical Flows

Let $G = (V, E)$ be any undirected graph with given function $r : E \to [0, 1]$ over the edges. Interpret G as an electrical network where an edge e is replaced by a **resistor** of resistance $r(e)$ ohm.

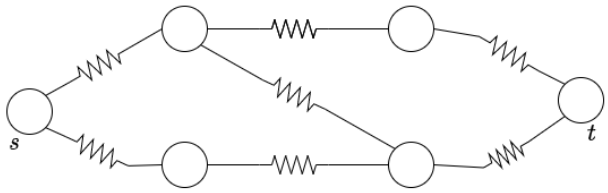

Figure 1: An electrical network interpretation of a graph.

Now suppose we attach a battery to vertices s and t (with the positive voltage at s) to induce a **current** $i : E \to \mathbb{R}$ from s to t and a **voltage potential** $\phi : V \to \mathbb{R}$ on the vertices. By **Ohm's law** from physics, we have that for every edge $e = (u, v)$,

$$i(e) = \frac{\phi(u) - \phi(v)}{r(e)}.$$

This allows us to define electrical flows from a physical point of view.

Definition 9. An **electrical flow** $f_e \in \mathbb{R}^E$ of value F on an undirected graph $G = (V, E)$ with resistors $r : E \to [0, 1]$ is the unique s-t flow corresponding to the current $i : E \to \mathbb{R}$, i.e., $f_e = i(e)$, that sends F units of current from s to t in the electrical network represented by G.

Remark. Let us take an even further detour for readers with some familiarity with spectral graph theory[a]. Suppose G has a unit resistor on every edge. Then, the Ohm law can be written as $i = B \cdot \phi$ and $B^\top i = F \cdot (\chi(s) - \chi(t))$ where B is the signed edge-incidence matrix of the graph and $\chi(s)$ and $\chi(t)$ are characteristic vectors of s and t in $\mathbb{R}^V$. Combining these together implies that $F \cdot (\chi(s) - \chi(t)) = B^\top B \cdot \phi = L \cdot \phi$ where L is Laplacian of G. Thus by solving this Laplacian system we can obtain the vector of vertex potentials and then use Ohm's law to obtain the electrical flow as well.

[a]You can safely ignore this entire discussion as it will not play any role in the rest of this lecture.

While the above interpretation is perhaps more natural from a physical point of view (and is the key toward obtaining fast algorithms for electrical flows), we will use an alternative definition of electrical flows as a purely optimization problem. We need this equivalent interpretation of electrical flows for this lecture. We will not prove the equivalence of these two definitions in this course and instead refer the interested reader to the monograph by Vishnoi [3].

Definition 10. Given any undirected graph $G = (V, E)$ with resistors $r : E \to [0, 1]$, the **energy** of a s-t flow $f : E \to \mathbb{R}$ (that satisfies preservation of flow but not necessarily capacity constraint) is

$$\text{Energy}(f) = \sum_{e \in E} r(e) \cdot f_e^2.$$

An **electrical flow** of value F is the unique flow f with value F from s to t that *minimizes* $\text{Energy}(f)$.

For us, the important fact is that (approximate) electrical flows (or rather approximate energy-minimizing flows) can be found efficiently using the seminal result of Spielman and Teng [2] on Laplacian solvers.

Proposition 11 (cf. [1, 2])**.** *There exists an algorithm that given any undirected graph $G = (V, E)$ with resistors $r : E \to [0, 1]$ and parameter $\delta > 0$, outputs a s-t flow f such that*

$$\text{Energy}(f) \leqslant (1 + \delta) \cdot \text{Energy}(\tilde{f}),$$

where $\tilde{f}$ is the energy-minimizing s-t flow in G (i.e., $\tilde{f}$ is the electrical flow). The algorithm runs in $\widetilde{O}(\frac{m \log R}{\delta})$ time where R is the ratio of the largest to smallest resistance.

4.2.2 Back to the Oracle

We are going to use Proposition 11 as our oracle. In particular, given the weights $\{w_e \mid e \in E\}$, we are going to define the resistors

$$r(e) = q_e + \frac{\varepsilon}{m},$$

for each edge $e \in E$ where $q_e = w_e/W$ is the normalized weights defined earlier. The reason for adding the extra additive term is to ensure that none of the resistances become too small (we clarify this further on). We then compute an approximate energy-minimizing flow f with parameter $\delta = \varepsilon$ on this network using Proposition 11. We first argue that this flow *approximately* preserves the oracle condition (but not exactly – we will show later this is not at all problematic).

Lemma 12. *As long as (P) is feasible, the returned flow f by the oracle satisfies*

$$\sum_{e \in E} q_e \cdot f_e \leqslant (1 + 5\varepsilon).$$

Proof. The first step is to bound the energy of f using its approximate energy-minimizing property.

Claim 13. $\text{Energy}(f) \leqslant (1+3\varepsilon)$.

Proof of Claim 13. Let $\tilde{f}$ be the electrical flow and f^* be the optimal s-t flow with unit-capacity in G. If (P) is feasible, we have that value of f^* is at least F, thus f^* is also a feasible solution for the energy-minimization flow problem of $\tilde{f}$, and as such $\text{Energy}(\tilde{f}) \leqslant \text{Energy}(f^*)$. In f^*, the flow over each edge is at most 1 as it satisfies capacity constraints, thus,

$$\text{Energy}(f^*) = \sum_{e\in E} r(e)\cdot f_e^{*2} \leqslant \sum_{e\in E} r(e) = \sum_{e\in E}(q_e + \frac{\varepsilon}{m}) = (\sum_{e\in E} q_e) + \varepsilon = 1+\varepsilon,$$

where we used the fact that $\sum_{e\in E} q_e = 1$ as we normalized the weights to get $\{q_e\}$. Thus, we have,

$$\text{Energy}(f) \leqslant (1+\varepsilon)\cdot\text{Energy}(\tilde{f}) \leqslant (1+\varepsilon)\cdot\text{Energy}(f^*) \leqslant (1+\varepsilon)\cdot(1+\varepsilon) \leqslant (1+3\varepsilon),$$

concluding the proof. □

We continue with the proof of Lemma 12. Notice that by Claim 13 we managed to bound a (weighted) variant of ℓ_2-norm of f and we now would like to use that to bound its ℓ_1-norm (with the same weights) to prove the lemma. We have,

$$\begin{aligned}
\sum_{e\in E} q_e\cdot f_e &\leqslant \sum_{e\in E} r(e)\cdot f_e = \sum_{e\in E}\sqrt{r(e)}\cdot\sqrt{r(e)\cdot f_e^2} \\
&\leqslant \sqrt{\sum_{e\in E}\sqrt{r(e)}^2}\cdot\sqrt{\sum_{e\in E}\sqrt{r(e)\cdot f_e^2}^2} && \text{(by Cauchy-Schwartz inequality)} \\
&= \sqrt{\sum_{e\in E} q_e + \frac{\varepsilon}{m}}\cdot\sqrt{\sum_{e\in E} r(e)\cdot f_e^2} && \text{(by definition } r(e) = q_e + \varepsilon/m\text{)} \\
&= \sqrt{1+\varepsilon}\cdot\sqrt{\text{Energy}(f)} && \text{(as } \textstyle\sum_{e\in E} q_e = 1 \text{ and by the definition of Energy}(f)\text{)} \\
&\leqslant \sqrt{(1+\varepsilon)\cdot(1+3\varepsilon)} \leqslant (1+5\varepsilon), && \text{(by Claim 13)}
\end{aligned}$$

finalizing the proof of Lemma 12. □

We now bound the width of the resulting (approximate) oracle solution, crucially using the fact that we ensured each resistance is at least ε/m and the bound on the energy of f.

Lemma 14. *The returned flow f by the oracle satisfies that for all $e \in E$,*

$$f_e \leqslant \sqrt{2m/\varepsilon}.$$

Proof. Suppose towards a contradiction that there exists an edge $e' \in E$ with $f_{e'} > \sqrt{2m/\varepsilon}$. Then, we have,

$$\text{Energy}(f) = \sum_{e\in E} r(e)\cdot f_e^2 \geqslant r(e')\cdot f_{e'}^2 \geqslant \frac{\varepsilon}{m}\cdot\frac{2m}{\varepsilon} = 2.$$

This contradicts Claim 13 that bounds $\text{Energy}(f) \leqslant (1+5\varepsilon)$ (when $\varepsilon < 1/5$, which we assume w.l.o.g.). □

Consider the following approximate oracle LP (O') instead of (O):

$$\begin{aligned}
&\sum_{e\in E} q_e\cdot f_e \leqslant 1 \\
&\sum_{p\in\mathcal{P}_{s,t}} f_p \geqslant F/(1+5\varepsilon) \\
&f_p \geqslant 0 \qquad\qquad \forall p\in\mathcal{P}_{s,t}.
\end{aligned}$$

We can compute f as an approximate electrical flow by Proposition 11 and scale it down by $(1+5\varepsilon)$ to obtain a feasible solution for the approximate oracle LP (O') by Lemma 12. Moreover, by Lemma 14, the width parameters of this oracle are $\rho = O(\sqrt{m/\varepsilon})$ and $\ell = 1$. Thus, by Theorem 4, if (P) is feasible, in $O(\varepsilon^{-1} \cdot \sqrt{m/\varepsilon} \cdot \ln m)$ iterations, we obtain a flow f satisfying:

$$\begin{aligned}
&\sum_{p \in \mathcal{P}_{s,t}} f_p \geqslant F/(1+5\varepsilon) \\
&\sum_{p \ni e} f_p \leqslant 1+\varepsilon && \forall e \in E \\
&f_p \geqslant 0 && \forall p \in \mathcal{P}_{s,t},
\end{aligned}$$

Scaling this flow down further by another $(1+5\varepsilon)$ factor, leads to a flow of value $(1-7\varepsilon) \cdot F$ which satisfies the capacity constraints and preservation of flow constraints. The runtime for each oracle is also $\widetilde{O}(m/\varepsilon)$ by Proposition 11. Finally, we by scaling $\varepsilon \leftarrow \varepsilon/7$ and running a binary search, we can use this to obtain an approximate maximum flow algorithm, proving the following theorem.

Theorem 15 (Weak version of [1])**.** *There is an algorithm that given any undirected unit-capacity graph $G = (V, E)$ and vertices s and t, finds a $(1-\varepsilon)$-approximate maximum s-t flow in G in $\widetilde{O}(\varepsilon^{-2.5} \cdot m^{1.5})$ time.*

The main result of [1] then improves this bound via an elegant *width reduction* technique to achieve an $\widetilde{O}(m^{4/3} \cdot \text{poly}(1/\varepsilon))$ time algorithm still purely based on MWU and further down to $\widetilde{O}(\varepsilon^{-11/3} \cdot m \cdot n^{1/3})$ using further combinatorial and randomized algorithms from prior work.

We conclude this lecture with the following important remark about MWU.

Remark. In the context of LPs and many other optimization problems, the MWU method can also be seen alternatively as a **boosting framework**: one can obtain a good approximation for a problem by designing a "much weaker approximation" for a more general variant of the problem (weighted by the MWU weight), where this weak approximation notion can be interpreted quite broadly[a] – this weak approximation ratio then translates to the number of iterations of the MWU method.

This is somewhat of a different view than what we started this lecture with which reduces solving the original problem to a *simpler* oracle problem (if we entirely neglects the role of the width parameter, which results in getting an MWU algorithm with large number of iterations).

[a] E.g., in the context of maximum flow, this was a flow that did not violate capacity constraints by more than $O(\sqrt{m})$.

References

[1] Paul F. Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282. ACM, 2011. 10, 12, 14

[2] Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Anal. Appl.*, 35(3):835–885, 2014. 12

[3] Nisheeth K. Vishnoi. Lx = b. *Found. Trends Theor. Comput. Sci.*, 8(1-2):1–141, 2013. 12