Lecture 10

Lecturer: Aleksander Mądry *Scribe: Zhenyu Liao*

1 Overview

Last time, we introduced the learning from expert advice framework and designed the multiplicative weight update method for that setting. Today, we will use these concepts to approximate solving of linear programs. Specifically, we will show how to use the multiplicative weights update method as a general tool for turning fast algorithms for certain "feasible on average" variant of the LP feasibility problem into fast algorithms that solve the canonical "worst case" variant of that problem.

2 Recap of the Last Lecture

Recall that the learning from expert advice framework corresponds to a certain T-round game that is described as Algorithm 1. Roughly speaking, in this game we are repeatedly "betting" on one of the n "experts" and aim to minimize the resulting loss to make it comparable to the overall loss of the single, best in hindsight, expert.

Algorithm 1 Learning from expert advice framework

for each round $t = 1 \dots T$ **do**
  Choose a *convex combination* $p^t := (p_1^t, \dots, p_n^t) \in \Delta_n$ of experts
  Once we have made our choice of p^t, a "loss" $l_i^t \in [-1, 1]$ is revealed for each expert i
  Our resulting loss in round t is $l^t := \sum_i p_i^t \, l_i^t$
end for

In order to tackle this task, we designed the multiplicative weights update method presented as Algorithm 2.

Algorithm 2 Multiplicative weights update method

Set $w_i^0 \leftarrow 1$, for each expert i
for $t = 1 \dots T$ **do**
  Choose $p_i^t := \frac{w_i^{t-1}}{W^{t-1}}$, for each expert i, i.e., each expert is taken proportionally to his/her weight.
  Once the losses l_i^t for all the experts i are revealed, set

$$w_i^t := \left(1 - \eta l_i^t\right) w_i^{t-1},$$

  where $0 < \eta \leq \frac{1}{2}$ is a parameter of the algorithm
end for

In the above, Δ_n denotes an n-dimensional simplex, i.e., a set of all non-negative n-dimensional vectors whose coordinates sum up to 1.

The performance of this algorithm is described by the following theorem.

Theorem 1 *For any $0 < \eta \leq 1/2$ and any expert i, the total loss $L(T)$ of the multiplicative weights update method (Algorithm 2) after T rounds is*

$$L(T) \leq L_i(T) + \eta \sum_{t=1}^{T} |l_i^t| + \frac{\ln n}{\eta},$$

where $L_i(T) = \sum_{t=1}^{T} l_i^t$ is the total loss of expert i after T rounds.

3 Approximate Solving of Linear Programs

The main goal of the learning from the expert advice framework we developed last time is to tackle the problem of optimal decision making under total uncertainty. However, this framework turns out to be extremely versatile and has applications to many other, seemingly unrelated, tasks.

Today, we will explore one of such surprising applications: approximate LP solving. Specifically, we will be interested in solving the following feasibility question. Given the following set of constraints $\mathcal{P}$

$$\begin{aligned} \mathcal{P} : Ax &\leq b \\ x &\in \mathcal{K}, \end{aligned}$$

where A is an m-by-n matrix, $b \in \mathbb{R}^m$, and $\mathcal{K} \subset \mathbb{R}^n$ is a certain convex set, as well as an error parameter $\varepsilon > 0$, we want to:

(i) find a point $\widehat{x} \in \mathcal{K}$ such that $A\widehat{x} \leq b + \varepsilon\vec{1}$, if $\mathcal{P}$ is feasible, i.e., if there exists $x^* \in \mathcal{K}$ with $Ax^* \leq b$;

(ii) or, return $\perp$ or a point $\widehat{x}$ as above, otherwise, i.e., if $\mathcal{P}$ is not feasible.

In other words, we want to either find a point $\widehat{x}$ in the convex set $\mathcal{K}$ that satisfies all m linear constraints of $\mathcal{P}$ up to an additive error of ε, or conclude that $\mathcal{P}$ is not feasible. (Note that even if $\mathcal{P}$ is not feasible there still might exist such a point $\widehat{x}$.)

3.1 Feasibility vs. Optimization Problems

Observe that even though we want to be able to just check the feasibility of the constraint set $\mathcal{P}$, this task captures solving general linear programs too. After all, every LP can be cast in the form

$$\begin{aligned} &\min c^T x \\ &s.t.\ Ax \leq b. \end{aligned}$$

This is, of course, an optimization problem and not a feasibility question, but we can easily reduce it to the latter. Specifically, if we knew the optimal value C^* of the objective, then solving the above LP would be equivalent to solving the feasibility question:

$$\begin{aligned} c^T x &\leq C^* \\ Ax &\leq b. \end{aligned}$$

In particular, this set of constraints would be feasible and any feasible solution x^* would be an optimal solution to the original LP.

Furthermore, if we do not know the value of C^*, we can just apply a binary search strategy. After all, if C is our current guess on the value of C^* then if we consider the feasibility question

$$\begin{aligned} c^T x &\leq C \\ Ax &\leq b, \end{aligned}$$

we will have that this set of constraints is feasible iff $C^* \leq C$. So, our focus on solving a feasibility question does not really lead to losing any generality.

3.2 (θ, ρ)-Oracles

When solving the feasibility question for $\mathcal{P}$, we do not really intend to do it from a scratch. Instead, we want to assume an existence of an algorithm, so-called oracle, that is capable of solving the feasibility question for $\mathcal{P}$ in certain much weaker sense. Our goal will be to provide a mechanism for utilizing such oracle in a black-box manner to obtain an algorithm that (approximately) solves the feasibility question for $\mathcal{P}$ in the "strong" sense we defined earlier.

To make this precise, let us consider the following definition.

Definition 2 *For any $0 \leq \theta \leq \rho$, an (θ, ρ)-oracle for the constraint set $\mathcal{P}$ is an algorithm that, on a given input vector $p \in \Delta_m$, outputs:*

(i) A point $\widetilde{x} \in \mathcal{K}$ such that $p^T A\widetilde{x} \leq p^T b$ and, furthermore, $A_i\widetilde{x} - b_i \in [-\theta, \rho]$, for all $1 \leq i \leq m$, if $\mathcal{P}$ is feasible;

(ii) $\perp$ or a point $\widetilde{x}$ as above, otherwise, i.e., if $\mathcal{P}$ is not feasible.

Note that the condition $p^T A\widetilde{x} \leq p^T b$ can be rewritten as

$$0 \geq p^T A\widetilde{x} - p^T b = \sum_{i=1}^{m} p_i \left(A_i\widetilde{x} - b_i\right).$$

So, it corresponds to having the point $\widetilde{x}$ satisfy the linear constraints of $\mathcal{P}$ *on average* (or, in *expectation*), with the average/expectation taken wrt the convex combination p. (It is not hard to see that insisting on p being a convex combination is not really needed and allowing any non-negative weights p would not change anything.)

Needless to say, the fact that the point $\widetilde{x}$ satisfies the constraints of $\mathcal{P}$ on average does not tell us much about whether any particular constraint is satisfied. We can have some of these constraints be violated by a large margin or satisfied with a large slack. Consequently, the purpose of the additional requirements that

$$A_i\widetilde{x} - b_i \in [-\theta, \rho],$$

for each $1 \leq i \leq m$, is to provide us with some absolute bounds on how large these margins and slacks can be. Specifically, they ensure that the violation margin is never larger than ρ and the slack is never larger than θ.

The parameter ρ is often referred to as the *width* of the oracle. One can think of it as a measure of "quality" of the oracle. The smaller it is the closer the "feasible on average" solution $\widetilde{x}$ returned by the oracle is to the actual "worst-case" feasibility. In particular, if ρ was equal to 0 then $\widetilde{x}$ would be guaranteed to be feasible. (In principle, there is no reason for us to insist that our bound on the extent of possible slack θ is small. However, as we will see later, it will be convenient to keep it bounded for technical reasons.)

Also, observe that if $x^* \in \mathcal{K}$ is a feasible solution to $\mathcal{P}$ then, obviously, we have that

$$p^T A\widetilde{x} - p^T b = \sum_{i=1}^{m} p_i(A_i x^* - b_i) \leq 0,$$

as $A_i x^* - b_i \leq 0$, for each i. So, any algorithm that simply solves the feasibility question for $\mathcal{P}$ exactly is a trivial oracle with $\rho = 0$. (Although θ might be still potentially quite large.)

Finally, notice that we were so far not too specific about the structure of the set $\mathcal{K}$, apart of insisting that it is convex. The reason is that in this framework we view this set as capturing the "easy" constraints in the constraint set $\mathcal{P}$. In particular, we assume that our oracle is able to navigate this set easily, i.e., all the solutions it returns are automatically in $\mathcal{K}$. Our whole focus is then just on ensuring that the "hard" linear constraints in $\mathcal{P}$ are always (approximately) satisfied.

4 Solving LPs with the Multiplicative Weights Update Method

Once we formally defined our goal: solving the feasibility problem $\mathcal{P}$ up to an additive error ε with the help of the (θ, ρ)-oracle for $\mathcal{P}$ (see Definition 2), we can focus on developing an algorithm for accomplishing it. As already mentioned earlier, we will employ here the learning from expert advice framework and the multiplicative weights update method we developed for that setting.

More precisely, we will use our oracle to setup certain instance of the learning from expert advice framework and then show that running multiplicative weights update method on this instance provides us with the ε-approximate answer to the feasibility problem $\mathcal{P}$ we want to solve.

To this end, let us consider a learning from expert advice framework in which we have m experts, each one of them corresponding to one of the linear constraints in $\mathcal{P}$. (Note that there is a slight notations clash here, as in our previous discussions the number of experts was denoted by n.)

Next, to fully specify an instance of that framework, we need to provide, for each round t of the underlying game (see Algorithm 1) and the convex combination p^t chosen by the algorithm, what the loss l_i^t of each expert i should be. We define these losses with the help of the oracle for $\mathcal{P}$. Namely, given the convex combination $p^t \in \Delta_m$, we run our oracle on this input and consider the solution x^t that this oracle provided. If $x^t = \perp$ then, from the definition of the oracle (Definition 2), we immediately know that $\mathcal{P}$ is infeasible and can just terminate. Otherwise, i.e., if x^t is an "feasible on average" solution to $\mathcal{P}$, we set

$$l_i^t := \frac{1}{\rho}\left(b_i - A_i x^t\right), \tag{1}$$

for each $1 \leq i \leq m$.

Observe that, by definition of the oracle, we have that

$$b_i - A_i x^t = -\left(A_i x^t - b_i\right) \in [-\rho, \theta],$$

for each i. So, normalizing by ρ (and using the fact that $\theta \leq \rho$) ensures that each loss $l_i^t \in [-1, 1]$, as required.

Now, once we specified the instance of the learning from expert advice framework, we just run the multiplicative weights update method (Algorithm 2) on it for some number T of rounds. (The value of T will be specified later.) At the end, provided none of the oracle calls produced $\perp$ – which would mean that we can just output $\perp$ and terminate) – we output

$$\bar{x} := \frac{1}{T}\left(\sum_{t=1}^{T} x^t\right), \tag{2}$$

i.e., the average of all the intermediate solutions x^t computed by the oracle in each round. (Observe that $\bar{x} \in \mathcal{K}$ by the fact that $\mathcal{K}$ is convex.) The resulting algorithm appears as Algorithm 3.

Algorithm 3 LP solving via multiplicative weight update method.

Initially, set $w_i^0 \leftarrow 1$, for all $1 \leq i \leq m$
for $t = 1, 2, ..., T$ **do**
  Define $p_i^t := \frac{w_i^{t-1}}{W^{t-1}}$, for each $1 \leq i \leq m$
  Let x^t be the output of the oracle on the input p^t
  if $x^t = \perp$ **then**
    return $\perp$
  end if
  Otherwise, $w_i^t \leftarrow w_i^{t-1}\left(1 - \frac{\eta}{\rho}\left(b_i - A_i x^t\right)\right)$, for each $1 \leq i \leq m$
end for
return $\bar{x} = \frac{1}{T}\left(\sum_{t=1}^{T} x^t\right)$

To get some intuition on why this algorithm works, notice that by our definition of the losses l_i^t (see (1)), the loss expert i is proportional to the extent to which the constraint i is satisfied. In particular, if this constraints is violated then the expert experiences a gain in that round. On the other hand, if this constraint is satisfied it suffers a positive loss.

At first, this might seem to be counter-intuitive. Why to penalize a constraint that is "doing well"? To understand why this is the right thing to do, observe that this loss setup makes the weights w_i^t of a given expert becomes larger if the constraint is violated and become smaller otherwise. Consequently, over time, the weight is shifted towards constraints that tend to be violated often. This, in turn, ensures that these constraints are more pronounced in the convex combination p^t passed to the oracle and thus making it increasingly harder for the "feasible on average" solution x^t output by the oracle to keep violating them.

The resulting dynamics is that even if most of the constraints get violated by the intermediate solutions x^t at some point, no constraint gets violated too many times. Therefore, once we take the average $\bar{x}$ (see (2)) of all these intermediate solutions, these violations "smoothen out" delivering a solution that violates each constraint only slightly.

The exact bounds that one gets on the performance of Algorithm 3 are provided in the following theorem.

Theorem 3 *Let $0 \leq \varepsilon \leq \frac{1}{2}$ be an error parameter and let us consider an (θ, ρ)-oracle for $\mathcal{P}$ with $\theta \geq \frac{\varepsilon}{2}$. If after at most $T = O\left(\theta\rho\varepsilon^{-2} \ln m\right)$ calls to that oracle the Algorithm 3 does not output $\perp$ then the solution $\bar{x}$ satisfies*

$$A\bar{x} \leq b + \varepsilon\vec{1}.$$

In other words, the above theorem tells us that Algorithm 3 solves the ε-approximate feasibility problem for $\mathcal{P}$. (Note that the condition $\theta \geq \frac{\varepsilon}{2}$ is just a technical requirement, we can always satisfy it by making θ (and, possibly, ρ) large enough.)

Conceptually, the above theorem tells us that solving the feasibility problem for a constraint set like $\mathcal{P}$ to an arbitrary good accuracy can be reduced to solving this constraint set only "on average". The desired accuracy, i.e., ε, and the quality of our "feasible on average" solutions, i.e., the oracle width $\rho \geq \theta$, impact solely the number of iterations that we will need to execute.

This general fact turns out to be very powerful and it is used in a number of areas besides optimization. For instance, it is the key phenomenon behind the boosting technique in machine learning, and the hard-cord set construction in circuit complexity.

Finally, observe that the bound on T in the Theorem 3 already hints on the key trade-off that underlies the whole approach. The trade-off between the quality of the oracle and the number of calls to that oracle made. After all, the larger the width ρ (and θ) is the easier to construct and faster the corresponding oracle. On the other hand, if the value of ρ (and θ) is too large then the resulting large number of oracle calls needed overshadows the gain from having the oracle be fast.

4.1 Proof of Theorem 3

In order to prove Theorem 3 we need to relate the quantities we care about, i.e., feasibility of intermediate solutions x^t and their average $\bar{x}$, to the language of learning from expert advice framework, in which the performance of the multiplicative weights update method is expressed (see Theorem 1).

We start by showing that the "feasibility on average" of all the oracle answers x^t implies that the loss l^t of the multiplicative weight update algorithm in our learning from expert advice instance is non-negative in each round t. To this end, notice that, by definition,

$$\begin{aligned} l^t &= \sum_{i=1}^{m} p_i^t l_i^t \\ &= \sum_{i=1}^{m} \frac{p_i^t}{\rho}(b_i - A_i x^t) \\ &= \frac{1}{\rho}((\rho^t)^T b - (p^t)^T A x^t) \\ &\geq 0, \end{aligned}$$

where the last inequality follows from the definition of the oracle (see Definition 2).

Now, let us fix some i. Our goal is to argue that

$$A_i\bar{x} \leq b_i + \varepsilon, \tag{3}$$

i.e., that the final solution $\bar{x}$ satisfies the constraint i up to an additive error of ε. Clearly, once we prove this, the ε-approximate feasibility of $\bar{x}$ will follow.

Observe that, by (1) and (2),

$$L_i(T) = \sum_{t=1}^{T} l_i^T = \frac{1}{\rho}\sum_{t=1}^{T}(b_i - A_i x^t) = \frac{T}{\rho}(b_i - A_i\bar{x}).$$

So, to get (3) we just need to show that

$$L_i(T) = \frac{T}{\rho}(b_i - A_i\bar{x}) \geq -\frac{\varepsilon T}{\rho},$$

i.e., that the average loss of expert i is at least $-\frac{\varepsilon}{\rho}$ per round.

To get an intuition why we should expect that such a statement is possible to prove, recall that we already know that the loss of the multiplicative weights update method $L(T) = \sum_t l^t$ is non-negative. So, the fact that the multiplicative weight update method asymptotically recovers the loss of the best expert (in particular, that of expert i), we should indeed expect that $L_i(T)$ cannot be too far from being non-negative either. We just need now to provide a quantitative analysis of this reasoning.

To this end, by our guarantee on the performance of the multiplicative weights update method (Theorem 1), we have that

$$\begin{aligned}
0 \quad &\leq \quad L(T) \leq L_i(T) + \eta\sum_{t=1}^{T}|l_i^t| + \frac{\ln m}{\eta} \\
&= \quad \sum_{t=1}^{T} l_i^t + \eta\sum_{t=1}^{T}|l_i^t| + \frac{\ln m}{\eta} \\
&= \quad (1-\eta)\sum_{t=1}^{T} l_i^t + 2\eta\sum_{t:\ l_i^t>0} l_i^t + \frac{\ln m}{\eta} \\
&= \quad \frac{1-\eta}{\rho}\sum_{t=1}^{T}(b_i - A_i x^t) + \frac{2\eta}{\rho}\sum_{t:\ l_i^t>0}(b_i - A_i x^t) + \frac{\ln m}{\eta} \\
&\leq \quad \frac{1-\eta}{\rho}\sum_{t=1}^{T}(b_i - A_i x^t) + \frac{2\eta}{\rho}\theta T + \frac{\ln m}{\eta},
\end{aligned}$$

where we used the fact that, by definition of the oracle (Definition 2), $b_i - A_i x^t \leq \theta$. Setting $\eta = \frac{\varepsilon}{8\theta}$ and multiplying both sides by $\frac{\rho}{(1-\eta)T}$, gives us that

$$\begin{aligned}
0 \quad &\leq \quad \frac{1}{T}\sum_{t=1}^{T}(b_i - A_i x^t) + \frac{\varepsilon}{4(1-\eta)} + \frac{8\theta\rho\ln m}{\varepsilon(1-\eta)T} \\
&\leq \quad b_i - A_i\bar{x} + \frac{\varepsilon}{2} + \frac{16\cdot\theta\rho\ln m}{\varepsilon T},
\end{aligned}$$

where we used the definition (2) of $\bar{x}$ and the fact that $(1-\eta) \geq \frac{1}{2}$ since $\theta \geq \frac{\varepsilon}{2}$.

Finally, taking $T = \frac{32\cdot\theta\rho\ln m}{\varepsilon^2}$, which is $O(\frac{\theta\rho\ln m}{\varepsilon^2})$ as needed, we ensure that

$$\begin{aligned}
0 \quad &\leq \quad b_i - A_i\bar{x} + \frac{\varepsilon}{2} + \frac{16\cdot\theta\rho\ln m}{\varepsilon T} \\
&\leq \quad b_i - A_i\bar{x} + \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \leq b_i - A_i\bar{x} + \varepsilon.
\end{aligned}$$

Thus, indeed, $\bar{x}$ satisfies constraint i up to an additive error of ε, and the theorem follows.

5 Back to the Maximum Flow Problem

Let us now apply the multiplicative weights update method–based approach we developed above to a specific optimization question: the maximum flow problem. Recall that we formulated this problem as

the following ℓ_∞-minimization question.

$$\begin{aligned} &\min \|f\|_\infty \\ &\text{s.t. } Bf = \chi_{s,t}. \end{aligned}$$

First, we need to translate this optimization problem into a feasibility question. Specifically, as discussed in Section 3.1, if we have our guess F on the value F^* of the maximum flow, we want to be able to check if $F \leq F^*$. Observe that answering this question boils down to checking feasibility of the following set of constraints.

$$\begin{aligned} \mathcal{F}(F): \quad F \cdot f &\leq \vec{1} \\ -F \cdot f &\leq \vec{1} \\ f &\in \mathcal{F}_{st}, \end{aligned}$$

where $\mathcal{F}_{st} = \{f \mid Bf = \chi_{st}\}$ is the (convex) set of all the s-t flows of value 1. (Recall that the flow f can have some coordinates f_e be negative, this just corresponds to flowing $|f_e|$ units of flow in the direction opposite to the edge e orientation.) So, relating this constraint set to our template constraint set $\mathcal{P}$ we considered above, we have that $\mathcal{K}$ is $\mathcal{F}_{st}$, the linear constraint matrix A is equal to

$$A = \begin{bmatrix} F \cdot I \\ -F \cdot I \end{bmatrix},$$

and the constraint vector b is simply an $2m$-dimensional all-ones vector $\vec{1}$.

The above formulation, even though a very natural one, is not the most convenient to work with. So, instead, we will use the following alternative, "symmetrized" set of constraints.

$$\begin{aligned} \overline{\mathcal{F}}(F): \quad F \cdot y &\leq \vec{1} \\ y &\in \overline{\mathcal{F}}_{st}, \end{aligned}$$

where $\overline{\mathcal{F}}_{st} := \{\mathsf{abs}(f) \mid Bf = \chi_s t\}$ and $\mathsf{abs}(f)$ is an m-dimensional vector defined as $\mathsf{abs}(f)_e = |f_e|$. In other words, $\overline{\mathcal{F}}_{st}$ is the set of vectors that encode edge flow values (with their directions disregarded) of some s-t flow of value 1.

In the light of this, our linear constraint matrix A is now simply the identity matrix scaled by F, i.e.,

$$A = F \cdot I,$$

and b is an m-dimensional all-ones vector $\vec{1}$. It is not hard to see that the feasibility problem for $\overline{\mathcal{F}}(F)$ is equivalent to the feasibility problem for $\mathcal{F}(F)$.

Observe that, in principle, we need to also argue that $\overline{\mathcal{F}}_{st}$ is a convex set. This can be shown to be indeed the case, but would require a bit of work. Looking at our analysis of Algorithm 3, however, it is not hard to see that what we need here is a slightly weaker property than convexity of $\overline{\mathcal{F}}_{st}$. Namely, all we really need is just that if we have a sequence of flow value vectors y^t with $y^t = \mathsf{abs}(f^t)$ for some unit s-t flows f^t and consider the "average" flow $\bar{f} := \frac{1}{T}\sum_t f^t$ then

$$F \cdot \mathsf{abs}(\bar{f}) \leq F \cdot \frac{1}{T}\sum_t y^t,$$

i.e., the edge flow value vector $\mathsf{abs}(\bar{f})$ of the average flow $\bar{f}$ is upper bounded by the average of edge flow value vectors of the individual flows f^t. This follows easily from the convexity of $|\cdot|$ though. (Note that we implicitly assume here that our oracle always provides the flow f^t that gave rise to the flow value vector y^t, which will indeed be the case.)

5.1 Shortest Path–based Oracle for the Maximum Flow Problem

Once we cast the maximum flow problem as the constraint set $\overline{\mathcal{F}}(F)$, we need to develop the last remaining ingredient needed to apply our multiplicative weight update method–based approach to it: the (θ, ρ)-oracle for $\overline{\mathcal{F}}(F)$.

Recall (see Definition 2) that, ideally, we want this oracle to be fast and, given a convex combination $p \in \Delta_m$ as an input, to either conclude that $\overline{\mathcal{F}}(F)$ is infeasible and return $\perp$, or find a unit s-t flow $f \in \mathcal{F}_{st}$ such that its edge flow value vector $y = \mathsf{abs}(f) \in \overline{\mathcal{F}}_{st}$ is "feasible on average", i.e., that it is the case that

$$0 \geq p^T A y - p^T b = \sum_e p_e \left(F \cdot y_e - 1\right), \tag{4}$$

where p_e is the weight that the input convex combination p puts on the constraint bounding the flow value of edge e. Additionally, we want that the flow f does not violate any individual capacity constraints by too much. That is, that

$$F \cdot y_e - 1 \in [-\theta, \rho], \tag{5}$$

for each edge e and as small θ and ρ, with $\theta \leq \rho$, as possible. In fact, as y_e is always non-negative, we know that we can always take $\theta = 1$. Thus, we can focus on bounding ρ alone. (This observation was one of the key motivations behind working with the formulation $\overline{\mathcal{F}}(F)$ instead of the formulation $\mathcal{F}(F)$. One can check that if we stuck with the formulation $\mathcal{F}(F)$ then the value of θ would be close to the value of ρ.)

It turns out that it is easy to devise an oracle for $\overline{\mathcal{F}}(F)$ using one of the most basic graph problems: the shortest-path computations.

To make this precise, given the input convex combination $p \in \Delta_m$, consider our graph with edge lengths given by p, i.e., the length l_e of the edge being p_e. Let us now compute a shortest s-t path P in this graph and define $\widetilde{f}$ to be the flow that sends one unit of flow along this single path P. Observe that, clearly, $\widetilde{f} \in \mathcal{F}_{st}$ and thus $\widetilde{y} = \mathsf{abs}(\widetilde{f}) \in \overline{\mathcal{F}}_{st}$. Also, we have that

$$\begin{aligned} p^T A \widetilde{y} - p^T b &= \sum_e p_e \left(F \cdot \widetilde{y}_e - 1\right) \\ &= F \sum_{e \in P} p_e \widetilde{y}_e - \sum_e p_e \\ &= F \cdot l(P) - |p|_1, \end{aligned}$$

where $l(P)$ is the length of the path P and $|p|_1 = \sum_e p_e$ is simply the ℓ_1-norm of the vector p.

We want to argue now that if $F \leq F^*$ then $l(P) \leq \frac{|p|_1}{F}$. Observe that, by our derivation above, this would mean that whenever $F \leq F^*$, the edge flow value vector $\widetilde{y}$ output by our shortest-path oracle is guaranteed to satisfy the "feasibility on average" condition (4). (So, in particular, if it turns out that $l(P) > \frac{|p|_1}{F}$, we know that $F > F^*$ and thus can safely return $\perp$.)

To this end, let us assume that indeed $F \leq F^*$ and let f^* be the maximum flow. Note that in our formulation of the maximum flow problem, f^* is an s-t flow of value 1 such that $\|f^*\|_\infty = \frac{1}{F^*}$. Therefore, the total length $l(f^*)$ of the flow f^* wrt the lengths given by the weights p is at most

$$l(f^*) \leq \sum_e p_e |f^*_e| \leq \frac{|p|_1}{F^*} \leq \frac{|p|_1}{F},$$

where we used the fact that $|f^*_e| \leq \|f^*\|_\infty \leq \frac{1}{F^*}$.

It has to be the case though that $l(P) \leq l(f^*)$, as sending one unit of flow on the shortest path P is the cheapest (wrt the lengths p) way of sending an unit of flow from s to t. Thus, indeed, $l(P) \leq l(f^*) \leq \frac{|p|_1}{F}$, and our shortest path oracle always returns correct answers.

Also, since $\widetilde{y}$ has its coordinates be either 0 or 1, we have that the width ρ of our shortest-path oracle is equal to

$$\rho = F \cdot 1 - 1 = F - 1.$$

Putting this all together, Theorem 3 allows us to conclude that calling our shortest path oracle at most

$$T = O\left(\frac{\theta\rho \ln m}{\varepsilon^{-2}}\right) = O\left(\frac{F \ln m}{\varepsilon^{-2}}\right)$$

times provides us with an ε-approximate solution to the maximum flow problem. As each oracle call corresponds to solving a shortest path question wrt non-negative edge lengths $l_e = p_e$, it can each be performed in only $\widetilde{O}(m)$ time, resulting in an ε-approximate maximum flow algorithm that runs in time

$$\widetilde{O}\left(mF\varepsilon^{-2}\right).$$

The fact that the running time of our algorithm depends on the value F of the flow we want to route is hardly desirable. In fact, technically, this algorithm is not even polynomial time! (By performing a different and more careful analysis, one could actually bound this algorithm's running time by $\widetilde{O}(m^2\varepsilon^{-2})$, even if capacities were non-uniform.)

The key takeaway message here, however, is that the multiplicative weight update framework gave us a simple and principled way of reducing a difficult graph problem, that is the maximum flow problem, to a simple graph problem, that is the shortest path computations.[1]

Finally, one should keep in mind that the most crucial aspect that we need to tackle when applying this multiplicative weights update method–based approach is the trade off between the running time of our oracle and the quality of answers it delivers, i.e., its width ρ. In our maximum flow scenario above, we had a very simple and fast oracle, but the width it delivered was poor. So, to get a better maximum flow algorithm we will need to construct a different, stronger oracle that, hopefully, will not be too costly to run.

[1] In the context of the maximum flow problem, this kind of connection was already unearthed by the augmenting path framework. But, here, we derived it in a natural way via fairly general considerations.