1 Recap

Recall from the previous lecture that we have

Theorem 14.1. *For all $0 \le \varepsilon \le 1/2$, for every sequence of gain vectors $g^1, \ldots, g^T \in [-\rho, \rho]^N$, the algorithm Hedge($\varepsilon$) produces probability vectors $p^1, \ldots, p^T \in \Delta_N$ such that if $T \ge \frac{4\rho^2 \ln N}{\varepsilon^2}$, then $\frac{1}{T}\sum_{t=1}^{T} \langle g^t, p^t \rangle \ge \frac{1}{T}\sum_{t=1}^{T} g_i^t - \varepsilon$ for every $i \in [N]$.*

Here, the Hedge algorithm initializes uniform probabilities $p^1 = (1/N, \ldots, 1/N)$ and sets the probabilities per round by $p_i^t \leftarrow \frac{w_i^t}{\sum_{j=1}^{N} w_j^t}$, where $w_i^{t+1} \leftarrow w_i^t e^{\varepsilon g_i^t/\rho}$.

2 Solving LPs

In this lecture, we use the above theorem to solve LPs approximately. We are given an LP with n variables and m constraints (excluding non-negativity constraints):

$$\begin{aligned} \max \quad & c \cdot x \\ & Ax \le b \\ & x \ge 0 \end{aligned}$$

Denote by OPT the optimal value of the LP. Let $K \subseteq \mathbb{R}^n$ be the polyhedron defined by the easy constraints, i.e. $K = \{x \in \mathbb{R}^n : x \ge 0, c \cdot x = OPT\}$, where OPT is found by binary search over possible objective values. We will find $x \in K$ such that $\langle a_i, x \rangle \le b_i + \varepsilon$ for all $i \in [m]$.

2.1 The Oracle

First, we make a simple observation.

Proposition 14.2. *Given a single constraint $\alpha \cdot x \ge \beta$, there is a fast oracle that finds $x \in K$ such that $\alpha \cdot x \ge \beta$ if such an x exists, or else outputs that no such x exists.*

Proof. We give the proof only for the case where $c_i \ge 0$ for all i; the general case is left as an exercise. Find $j^* = \text{argmax}_j \frac{\alpha_j OPT}{c_j}$. Set $x^* = \frac{OPT}{c_{j^*}} e_{j^*}$. Then x^* is a vector that maximizes $\alpha \cdot x$ subject to $x \in K$. Now check whether $\alpha \cdot x^* \ge \beta$ and output x^* if the inequality is satisfied; otherwise output no such x exists.

Note that the runtime of this oracle procedure is polynomial in n, the dimension of the problem. (As we'll see later in the lecture, when we consider max-flows, that even if the dimension is exponential we might be able to do something better for special cases.) □

To solve the LP, we will call the above oracle while using Hedge by repeatedly asking the oracle to produce a solution to a convex combination of the m constraints and updating the weight/importance of each constraint based on how badly the constraint was violated by the current solution. To this end, we will set our gain vectors to be the amount of violation of the corresponding constraint.

An obvious upper bound on the maximum possible violation is the *width* ρ of the LP, which is defined by

$$\rho := \max_{x \in K, i \in [m]} \{|a_i \cdot x - b_i|\}.) \tag{14.1}$$

We will assume that $\rho \geq 1$.

2.2 The Algorithm

Below we give the formal algorithm to approximately solve the LP.

1. $p^1 \leftarrow (1/m, \ldots, 1/m)$
2. Call oracle to give x^t such that $\sum_{i=1}^m \langle p_i^t a_i, x \rangle \leq \sum_{i=1}^m p_i^t b_i$. If oracle says no, it means that the original LP is infeasible, as any feasible solution to the original LP would satisfy a convex combination of the LP's constraints.
3. Define gain vector by $g_i^t = \langle a_i, x^t \rangle - b_i$.
4. Update p^{t+1} using Hedge(ε).
5. Run steps 1 to 4 for $T := \Theta(\rho^2 \ln m/\varepsilon^2)$ rounds.
6. $\hat{x} \leftarrow (x_1 + \cdots + x_T)/T$.
7. Return $\hat{x}$.

2.3 The Analysis

Theorem 14.3. *For every $0 \leq \varepsilon \leq 1/4$, the above algorithm returns $\hat{x} \in K$ such that $\langle a_i, x \rangle \leq b_i + \varepsilon$ for all $i \in [m]$ and calls the oracle $O(\rho^2 \ln m/\varepsilon^2)$ times.*

Proof. Let $i \in [m]$. Define $\alpha^t = \sum_{i=1}^m p_i^t a_i$ and $\beta^t = \sum_{i=1}^m p_i^t b_i$. Then

$$\begin{aligned} \langle p^t, g^t \rangle &= \langle p^t, Ax^t - b \rangle \\ &= \langle p^t, Ax^t \rangle - \langle p^t, b \rangle \\ &= \langle \alpha^t, x^t \rangle - \beta^t \\ &\leq 0. \end{aligned}$$

So the left hand side in Theorem 14.1

$$\frac{1}{T} \sum_{t=1}^T \langle p^t, g^t \rangle \leq 0.$$

Second, the right hand side in Theorem 14.1

$$\frac{1}{T} \sum_{t=1}^T \langle a_i, x^t \rangle - b_i - \varepsilon = \langle a_i, \hat{x} \rangle - b_i - \varepsilon.$$

By Theorem 14.1, we have

$$0 \geq \frac{1}{T} \sum_{t=1}^T \langle p^t, g^t \rangle \geq \frac{1}{T} \sum_{t=1}^T \langle a_i, x^t \rangle - b_i - \varepsilon \geq \langle a_i, \hat{x} \rangle - b_i - \varepsilon.$$

Hence, for each constraint i, we have that $\langle a_i, \hat{x} \rangle \leq b_i + \varepsilon$. □

2.4 A Small Extension: Approximate Oracles

Recall the definition of the problem width from (14.1). Two comments:

- In the above analysis, we do not care about the maximum value of $|a_i \cdot x - b_i|$ over all points $x \in K$, we only care about the largest this expression can get over all the points that can be potentially returned by the oracle. While this seems a pedantic point, it will be useful — if there are many solutions we can return for $\alpha \cdot x \leq \beta$, we can try to return the one with least width. But we can do more, as the next point outlines.
- We can also relax the oracle to satisfy $\alpha \cdot x \leq \beta + \delta$ for some small $\delta > 0$ instead. Define the *width* of the LP with respect the relaxed oracle to be

$$\rho_{\text{relaxed}} := \max_{i \in [m], x \text{ returned by relaxed oracle}} \{|a_i \cdot x - b_i|\}. \tag{14.2}$$

  The new range of g^t using the relaxed oracle will be $[-\rho_{\text{relaxed}}, \rho_{\text{relaxed}}]^N$. So running the same algorithm with the relaxed oracle will give us that $a_i \cdot \hat{x} \leq b_i + \epsilon + \delta$ for all $i \in [m]$ and the number of calls to the relaxed oracle is $O(\rho^2_{\text{relaxed}} \ln m/\epsilon^2)$.

3 Application to Max Flows

Here, we begin with a result similar to Theorem 14.1 with slightly worse bound but requiring fewer rounds. We will then apply it to the max flow problem.

Theorem 14.4. *For all $0 \leq \varepsilon \leq 1/2$, for every sequence of gain vectors $g^1, \ldots, g^T \in [0, \rho]^N$, the multiplicative weights (MW) algorithm with updates*

$$w_i^{t+1} \leftarrow w_i^t(1 + \varepsilon g_i^t/\rho)$$

produces probability vectors $p^1, \ldots, p^T \in \Delta_N$ such that if $T = \Omega(\frac{\rho \ln N}{\varepsilon^2})$, then

$$\frac{1}{T}\sum_{t=1}^{T} \langle g^t, p^t \rangle \geq \frac{1}{T}\sum_{t=1}^{T} g_i^t(1-\varepsilon) - \varepsilon$$

for every $i \in [N]$.

Again, the process works online: each time the MW algorithm produces the probability vector p^t first (with $p_i^t = w_i^t / \sum_j w_j^t$), then the adversary produces the gain vector g^t, whence we get gain $\langle g^t, p^t \rangle$, we update the weights to get w^{t+1}, and repeat.

3.1 An LP for the Max-Flow Problem

In the max flow problem, we are given an $s - t$ network G possibly having multi-arcs, each arc having capacity 1. Let P be the set of all $s - t$ paths in G. We use a path-based LP formulation:

$$\begin{aligned} \max \quad & \sum_{p \in P} f_p & & \\ & \sum_{p \ni e} f_p \leq 1 & & \forall e \in E \\ & f_p \geq 0 & & \forall p \in P \end{aligned}$$

Let F be the optimal flow value over feasible $s-t$ flows in G. Let $K = \{f : f \geq 0, \sum_{p \in P} f_p = F\}$. Given probabilities $q^t \in \Delta_m$, the convex combination of the LP constraints weighted by q^t is $\sum_{e \in E} q_e^t \sum_{p \ni e} f_p \leq \sum_{e \in E} q_e^t = 1$, equivalently $\sum_{p \in P} f_p \sum_{e \in p} q_e^t \leq 1$. We may think of $\sum_{e \in p} q_e^t$ as the length of path p. Observe that an optimal solution to minimizing $\sum_{p \in P} f_p \sum_{e \in p} q_e^t$ subject to $f \in K$ is to place all F flow on a shortest $s-t$ path according to the lengths defined by q_e^t. So we may assume that we have an oracle that finds $f_p \in K$ such that $\sum_{p \in P} f_p \sum_{e \in p} q_e^t \leq 1$ or tells us that no feasible flow exists.

3.2 The Algorithm

This leads to the following algorithm for finding a max flow that approximately satisfies the capacity constraints:

1. $q^1 \leftarrow (1/m, \ldots, 1/m)$.
2. Call oracle to give $f^t \in K$ such that $\sum_{p \in P} f_p \sum_{e \in p} q_e^t \leq 1$. (If oracle says no, it means that the original LP is infeasible, as any feasible solution to the original LP would satisfy a convex combination of the LP's constraints.)
3. Define gain vector by $g_e^t = F$ for all $e \in p^t$ and $g_e^t = 0$ for all $e \notin p^t$, where p^t is the shortest path found by the oracle at round t.
4. Update q^{t+1} using MultiplicativeWeights(ε).
5. Run steps 1 to 4 for $T := \Theta(\rho \ln m/\varepsilon^2)$ rounds, where $\rho := F$.
6. Return $\hat{f} \leftarrow (g_1 + \cdots + g_T)/T$.

Observe that every g^t here is within the range $[0, \rho]^N$. So we can use Theorem 14.4, which requires only $O(F \ln m/\varepsilon^2)$ rounds. Each iteration takes $O(m + n \log n)$ to find the shortest path according to the lengths induced by q^t. So the total runtime is $\tilde{O}(mF)$. Although, this runtime is no better than the Ford Fulkerson algorithm, we will see a way to reduce the runtime via electrical flows next lecture.

3.3 The Analysis

For completeness, we show that every capacity constraint is approximately satisfied using the same analysis as before. Denote the vector with 1 in position e and 0 elsewhere by χ_e. Let $\hat{f}_p = \min_{e \in p} \hat{f}_e$. Let $e \in E$. Assuming that the original LP is feasible, we know that the oracle guarantees our g^t will satisfy $\langle g^t, q^t \rangle = \sum_{p \in P} f_p^t \sum_{e \in p} q_e^t \leq 1$. So we have

$$\begin{aligned}
1 &\geq \frac{1}{T} T \\
&= \frac{1}{T} \sum_{t=1}^{T} \langle g^t, q^t \rangle \\
&\geq \left(\frac{1}{T} \sum_{t=1}^{T} g_e^t \right) (1 - \varepsilon) - \varepsilon && \text{by Theorem 14.4} \\
&\geq (1 - \varepsilon) \langle \hat{f}, \chi_e \rangle - \varepsilon
\end{aligned}$$

So $\sum_{p \ni e} \hat{f}_p \leq \frac{1+\varepsilon}{1-\varepsilon} \leq (1+\varepsilon)^2$.

3.4 An Example

To see how the algorithm corrects the paths it chooses towards better paths, we demonstrate the first few steps of the algorithm on an example. For simplicity of calculations, we ignore the ε in updating the weights, i.e., assume that $w_e^{t+1} \leftarrow w_e^t(1 + g_e^t/\rho)$. The label on edge e in the tth round indicates w_e^t. The green path in the tth round indicates the shortest path the oracle gives according to weights w_e^t.

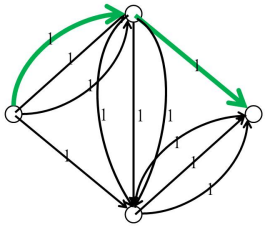

Figure 14.1: Round 1

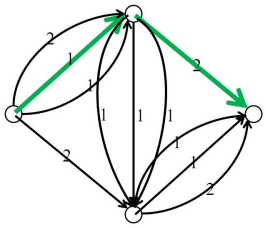

Figure 14.2: Round 2

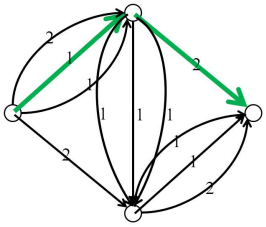

Figure 14.3: Round 3

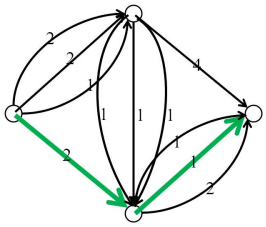

Figure 14.4: Round 4

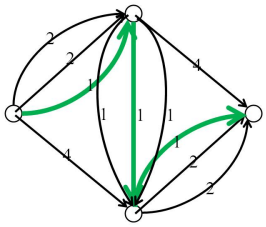

Figure 14.5: Round 5

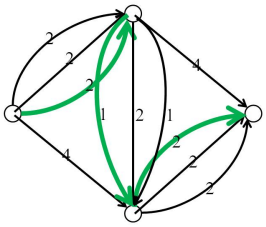

Figure 14.6: Round 6