

CS 170

# Efficient Algorithms and Intractable Problems

Lecture 26

Online Algorithms 2 and  
Beyond this course

Nika Haghtalab and John Wright

EECS, UC Berkeley

# Announcements

This is the last lecture!

Last graded homework due this Sunday

Final exam on Monday 5/12 (last lecture's slides had an erroneous date!)

We will have an exam review sessions

→ Early RRR week, look out for an Ed post coming soon

We will have reduced OH during RRR week (see calendar for updated hours)

# Online Algorithms

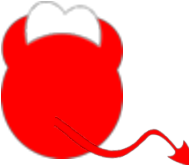
Index ▲ 1.56 ▼ 0.78



# Recall Online Learning Formalism

There are  $n$  “experts”,  $i = 1, \dots, n$  and  $T$  days  $t = 1, \dots, T$

On each day  $t = 1, \dots, T$

- All experts  $i$  give me their *opinion*  $o_i^{(t)}$  (binary, like Yes/No, or Up/Down)
- I make my prediction  $guess^{(t)}$
- Afterwards, I see the real outcome  $real^{(t)}$ , which can be worst-case   
→ Happy if guessed correctly and sad if I made a mistake!

My goal:

$$\underbrace{\sum_{t=1}^T \mathbf{1}(guess^{(t)} \neq real^{(t)})}_{\text{\# of mistakes my Alg makes}} \approx \underbrace{\min_i \sum_{t=1}^T \mathbf{1}(o_i^{(t)} \neq real^{(t)})}_{\text{\# of mistakes the best expert}}$$

# Recall: Weighted Majority Algorithm

## (Deterministic) Weighted Majority with parameter $\epsilon$

Initialize weights  $w_i^{(1)} = 1$  for all  $i \in [n]$ .

For  $t = 1, \dots, T$

Take the weighted majority of the experts:

$$\mathit{guess}^{(t)} = \operatorname{argmax}_y \sum_{i \in [n]} w_i^{(t)} \mathbf{1}(o_i^{(t)} = y)$$

For  $i = 1, \dots, n$

If  $o_i^{(t)} \neq \mathit{real}^{(t)}$  then  $w_i^{(t+1)} \leftarrow w_i^{(t)} (1 - \epsilon)$ , else  $w_i^{(t+1)} \leftarrow w_i^{(t)}$ .

## Theorem: Guarantees of Weighted Majority $\epsilon = 0.5$

For  $M$ : Algorithms # mistakes and  $OPT$ : best expert's # mistakes, the (Deterministic) weighted majority algorithm with  $\epsilon = 0.5$  gets

$$M \leq 2.4(\log_2(n) + OPT).$$

# Today

We will learn more about online algorithms and their performance.

We'll learn the Multiplicative Weights Updates (MWU) Algorithm

One of my all-time favorite algorithms!

We see how they can be used to prove some theorems or design some algorithms we had seen before!

# How much do we regret?

We showed that *Alg's # mistakes*  $\leq 2.4(\log_2(|H|) + OPT)$  is good if *OPT* is small.

→ If best expert is wrong 5% of the time, we are wrong 12% of the time

→ If best expert is wrong 25% of the time, we are wrong half the time!

It would have been nice, if instead *Alg's # mistakes*  $- OPT \leq \text{small}$

→ Ideally, smaller than  $o(T)$ .

→ On average over  $T$  timesteps, we do nearly as well as the best expert.

Idea: Smoothly transition between predicting Yes or No based on the weights.

→ Weighted majority: 49% Yes, 51% No, we predict No

Randomized Weighted majority:

→ If 49% Yes, 51% No, we predict Yes with 0.49 prob and No with 0.51 prob.

→ We can also use less aggressive  $\epsilon$ .

# Randomized Weighted Majority

**Randomized** Weighted Majority Algorithm with parameter  $0 < \epsilon < 1$   
Every time an expert makes a mistake, its weight is multiplied by  $(1 - \epsilon)$

## Randomized Weighted Majority with parameter $\epsilon$

Initialize weights  $w_i^{(1)} = 1$  for all  $i \in [n]$ .

For  $t = 1, \dots, T$

Guess with probability proportional to the weighted majority:

$$\text{guess}^{(t)} \leftarrow y \text{ with prob. } \frac{1}{W^{(t)}} \sum_{i \in [n]} w_i^{(t)} \mathbf{1}(o_i^{(t)} = y)$$

For  $i = 1, \dots, n$

If  $o_i^{(t)} \neq \text{real}^{(t)}$  then  $w_i^{(t+1)} \leftarrow w_i^{(t)} (1 - \epsilon)$ , else  $w_i^{(t+1)} \leftarrow w_i^{(t)}$ .



# Randomized Weighted Majority

## Theorem: Guarantees of Weighted Majority $\epsilon$

For  $M$ : Algorithms # mistakes and  $OPT$ : best expert's # mistakes, the **randomized weighted majority** algorithm with  $\epsilon$  gets

$$\mathbb{E}[M] \leq (1 + \epsilon)OPT + \frac{1}{\epsilon} \log_2(n).$$

For  $\epsilon = \sqrt{\frac{\log_2(n)}{OPT}}$ , get  $\mathbb{E}[M] \leq OPT + 2\sqrt{T \log_2(n)}$ .

# Beyond Binary Guesses and Outcomes

We can extend this to non-binary general outcomes and predictions

We want to take one of  $n$  actions, each one is like an “expert” E.g., each s-t path is one action/expert.

→ Each action  $i$  has some **cost** at time  $t$ , called  $c_i^{(t)} \in [0,1]$  E.g., The traffic of the  $i^{th}$  s-t path at time  $t$ .

→ Alg plays action  $i_t$  at time  $t$ , perhaps randomly

→ We see **cost** of all actions after we take an action

We want the total cost of the algorithm not to be much larger than the cost of the best action, in hindsight.

→ Want small regret

$$\text{REGRET} := \underbrace{\sum_{t=1}^T c_{i_t}^{(t)}}_{\text{Total cost of Alg's choices}} - \underbrace{\min_{i^*} \sum_{t=1}^T c_{i^*}^{(t)}}_{\text{Total cost of the best action}} \leq \textit{small}$$

# Multiplicative Weights Update (MWU)

Multiplicative Weights Update with parameter  $\epsilon$

Initialize weights  $w_i^{(1)} = 1$  for all  $i \in [n]$ .

For  $t = 1, \dots, T$

Play action  $i$  with probability  $\frac{w_i^{(t)}}{W^{(t)}}$

Observe costs  $c_i^{(t)}$  for all  $i = 1, \dots, n$

For  $i = 1, \dots, n$ ,

let  $w_i^{(t+1)} \leftarrow w_i^{(t)} (1 - \epsilon c_i^t)$

**Theorem:** For an appropriate choice of  $\epsilon = \sqrt{\log_2(n) / T}$ , the **MWU Algorithm** has

$$\mathbb{E}[\text{Regret}] \leq O\left(\sqrt{T \log_2(n)}\right).$$

# No-Regret Algorithms

$$\text{REGRET} := \underbrace{\sum_{t=1}^T c_{i_t}^{(t)}}_{\text{Total cost of Alg's choices}} - \underbrace{\min_{i^*} \sum_{t=1}^T c_{i^*}^{(t)}}_{\text{Total cost of the best action}}$$

No-regret algorithms:

→ Algorithms for which **REGRET** (or its expectation) is  $o(T)$

→ E.g. MWU is no-regret because  $\mathbb{E}[\text{Regret of MWU}] \leq O(\sqrt{T \log_2(n)})$ .

→ It doesn't literally mean that you have 0 regret!

→ It means if you play the algorithm long enough ( $T \rightarrow \infty$ ) then your average regret is  $\frac{\text{REGRET}}{T} \rightarrow 0!$

→ Meaning, in hindsight, you do not much regret not having known the best expert a priori! You'll catch up and do nearly as well as the best.

Proving the MinMax Theorem  
using Multiplicative Weights Update

# Revisiting Zero-Sum Games

## **Usage Examples:**

Most two-player board/card games.

Competition between two rival firms, splitting the market share.

Actions are played by self-interested agents in a win-lose game.

Each player takes some actions.

Equilibrium, if neither can improve their position.

# Two player Games

Players: Player **1** and **2**

Strategies: Sets of actions  $X, Y$

Payoffs: When **1** plays  $x$  and **2** plays  $y$ .

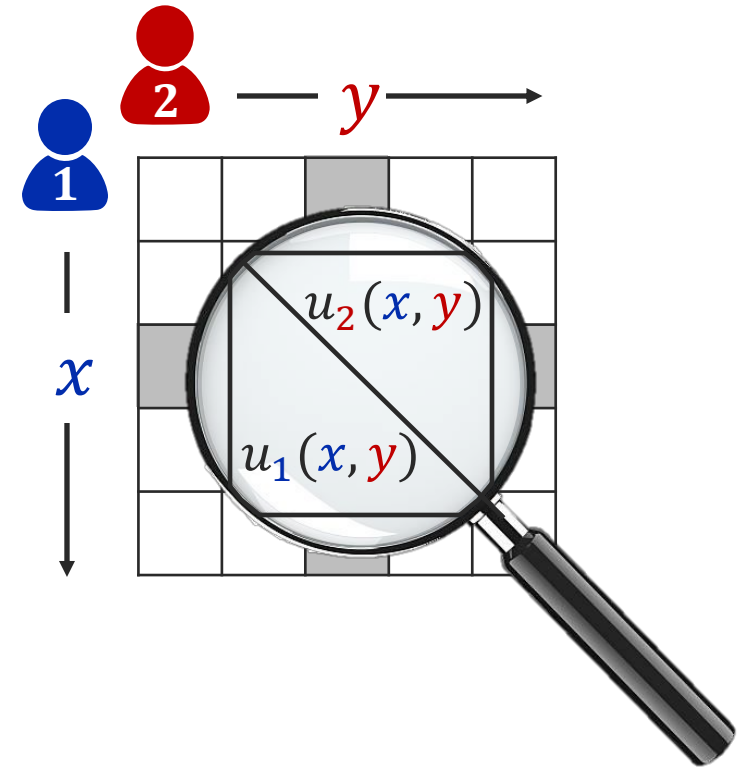
**1**'s payoff :  $u_1(x, y)$       **2**'s payoff :  $u_2(x, y)$

Zero-sum games: focus of this section

$$-u_1(x, y) = u_2(x, y)$$

We'll call one of the loss and one gain/utility

$$\ell(x, y) = -u_1(x, y) \quad (\text{in this section})$$



# MinMax Equilibrium

Mixed Strategies:  1 picks distribution  $p$  over  $X$  and  2 picks distribution  $q$  over  $Y$ .

**MinMax** value

$$\min_p \max_q p^\top L q$$

(player 1 goes first)

**MaxMin** value

$$\max_q \min_p p^\top L q$$

(player 2 goes first)

## Von Neumann's MinMax Theorem

MinMax value = MaxMin value

Under some conditions, e.g.,  $X$  and  $Y$  finite.

**Proof:** It was easy to see that going second is an advantage for either player

$$\min_p \max_q p^\top L q \geq \max_q \min_p p^\top L q$$

The reverse direction was the hard part of this proof.



# Proving the reverse direction of MinMax

**Idea: Online algorithms** and **MinMax** are about interactions with an adversary. So let's use the a no-regret algorithm for one of the players (or both).

Imagine hypothetical interactions over  $t = 1, \dots, T$  days:

- The row player uses Multiplicative Weights Update to choose one row per day

Playing row  $i$  with probability  $p_i^{(t)}$

- Column player “best responds” to the row player

Playing  $q^{(t)} = \operatorname{argmax}_q p^{(t)\top} L q$

- The row player's cost vector of  $(c_1^{(t)}, \dots, c_n^{(t)}) = L q^{(t)}$  is revealed and she suffers  $p^{(t)\top} L q^t$  loss in expectation.

# Proving the reverse direction of MinMax

Playing row  $i$  with probability  $p_i^{(t)}$  using MWU

Playing column  $q^{(t)} = \operatorname{argmax}_q p^{(t),\top} L q$ , Row player cost vector is revealed to be  $(L q^t)$

---

I want to prove that  $\min_p \max_q p^\top L q \leq \max_q \min_p p^\top L q$  using a construction of a pair of strategies that are at minmax equilibrium.

# Algorithm for Max Flow

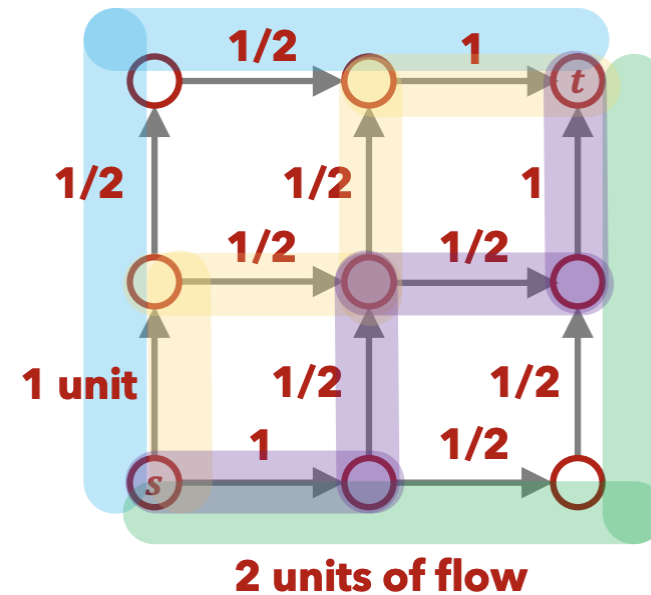
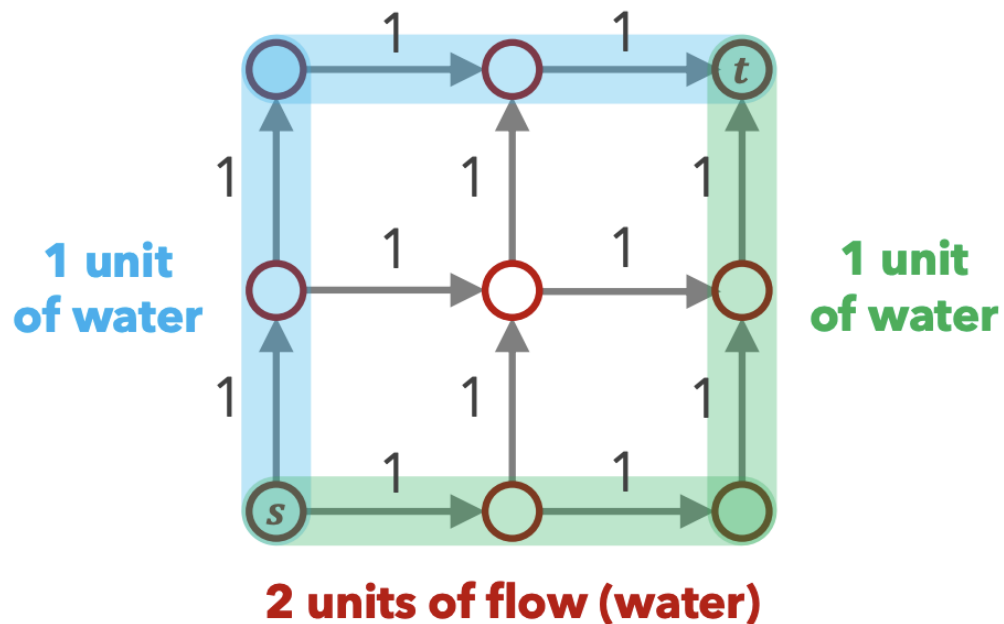
## Using Multiplicative Weights Update

From here on, material covered is not in scope for the final exam!  
Let's just have some fun!

# Revisiting Max Flow

**Input:** A directed graph  $G = (V, E)$ , source vertex  $s$  and sink vertex  $t$ , and edge capacities  $c_e$  for all  $e \in E$ . For ease today assume  $c_e = 1$  for all edges.

**Output:** A maximum valid  $s$ - $t$  flow



# We solved flow problems with an LP before

The primal and dual LPs corresponding to max flow and min cut:

Let  $\mathcal{R}$  be the set of all **s-t paths**,  $f_P$  is the amount of flow on path  $P$  and  $\ell_e$  are dual variable **indicating the cut**.

**Primal: Max Flow**

$$\max \sum_{P \in \mathcal{R}} f_P$$

$$\sum_{P \ni e} f_P \leq 1 \quad \text{for all } e \in E$$

$$f_P \geq 0 \quad \text{for all } P \in \mathcal{R}$$

**Dual: Min Cut**

$$\min \sum_{e \in E} \ell_e$$

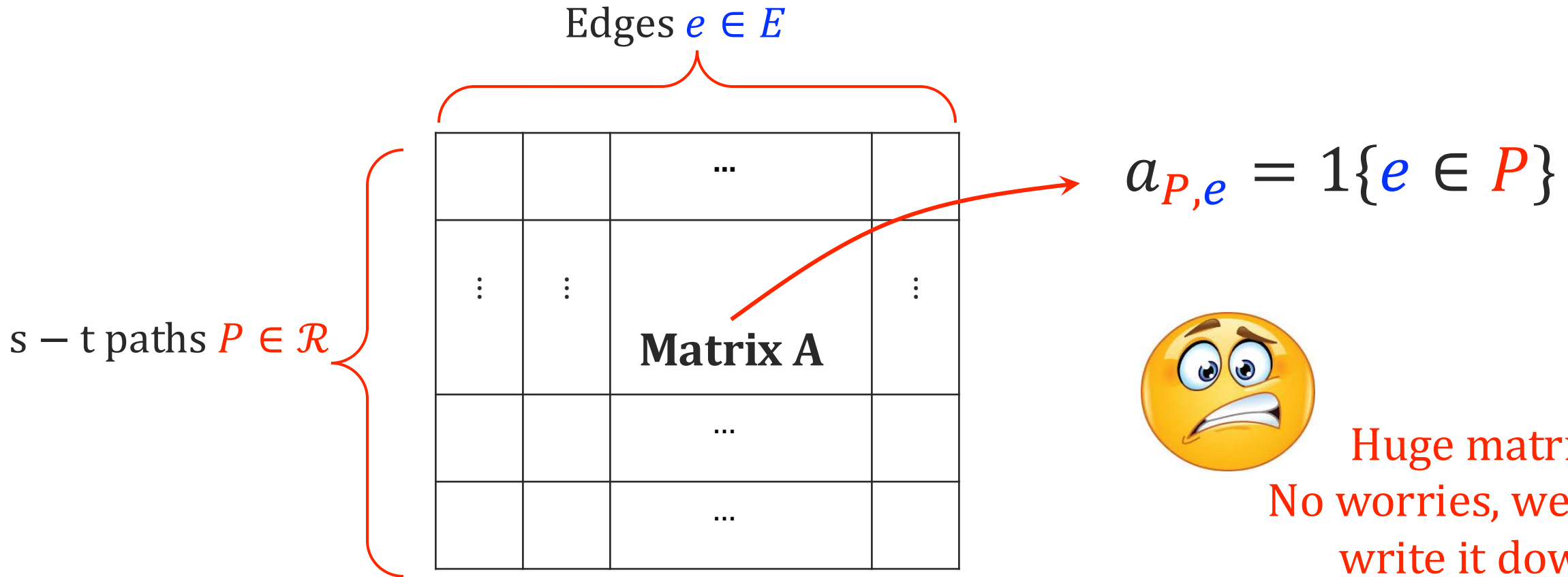
$$\sum_{e \in P} \ell_e \geq 1 \quad \text{for all } P \in \mathcal{R}$$

$$\ell_e \geq 0 \quad \text{for all } e \in E$$

# Min Cut-Max Flow as a MinMax Game

Column player: Choosing the dual variables  $\ell_e$ s.

Row player: Choosing the primal variables  $f_P$



# MinMax Value of the Game

## Claim

Let OPT be the max flow (= min cut). The the minmax value of this game is

$$\min_{y \in [0,1]^{\mathcal{R}}} \max_{x \in [0,1]^E} y^{\top} Ax = \max_{x \in [0,1]^E} \min_{y \in [0,1]^{\mathcal{R}}} y^{\top} Ax = \frac{1}{\text{OPT}}$$

**Proof idea:** construct strategies from the primal and dual solutions. E.g., scaled dual variables  $\frac{\ell_e}{\text{OPT}}$  are mixed strategy: Col. Player puts a uniform distribution over its cut.

## Primal: Max Flow

$$\max \sum_{P \in \mathcal{R}} f_P$$

$$\sum_{P \ni e} f_P \leq 1 \quad \text{for all } e \in E$$

$$f_P \geq 0 \quad \text{for all } P \in \mathcal{R}$$

## Dual: Min Cut

$$\min \sum_{e \in E} \ell_e$$

$$\sum_{e \in P} \ell_e \geq 1 \quad \text{for all } P \in \mathcal{R}$$

$$\ell_e \geq 0 \quad \text{for all } e \in E$$

# Min Cut-Max Flow as a MinMax Game

# columns is small enough: We will run MWU on them

# rows is large: Can we efficiently compute "best response"?

s-t paths  $P \in \mathcal{R}$

Edges  $e \in E$

		...	
⋮	⋮	<b>Matrix A</b>	⋮
		...	
		...	

$$a_{P,e} = 1\{e \in P\}$$



Huge matrix!



# Solving Max Flow with Multiplicative Weights

Actions for the column player are edges  $e \in E$ .

For  $t = 1, \dots, T$

- Use the MWU algorithm to generate a probability distribution  $x^{(t)} \in [0,1]^E$  over the edges (actions)
- Let  $P^{(t)}$  be row player's "best response"

$$\text{Path } P^{(t)} \leftarrow \operatorname{argmin}_{P \in \mathcal{R}} \sum_{e \in P} x_e^{(t)}$$

- Create rewards  $r_e^{(t)} = 1\{e \in P^{(t)}\}$  for all edges and feed them as reward (negative loss) to MWU.

Let  $\bar{f}$  put flow  $\frac{OPT}{T} \times (\#\text{time } P^{(t)} = P)$  on each path  $P \in \mathcal{R}$ .

## Theorem:

When  $T \geq 8 OPT^2 \cdot \frac{\ln(|E|)}{\epsilon^2}$   
then  $\bar{f}(1 - \epsilon)$  is an  $(1 - \epsilon)$ -  
approximately optimal flow!

# How to implement efficiently?

$O\left(\frac{\text{OPT}^2 \ln(|E|)}{\epsilon^2}\right)$  rounds

Actions for the row player are edges  $e \in E$ .

For  $t = 1, \dots, T$

- Use the MWU algorithm to generate a probability distribution  $x^{(t)} \in [0,1]^E$  over the edges (actions)
- Let

Path  $P^{(t)} \leftarrow \operatorname{argmin}_{P \in \mathcal{R}} \sum_{e \in P} x_e^{(t)}$

Shortest path with  $x_e^{(t)}$  as edge lengths:  $O(|E| \log |V|)$

Each step  $O(|E|)$

- Create rewards  $r_e^{(t)} = 1\{e \in P^{(t)}\}$  for all edges and and feed them as reward (negative loss) to MWU.

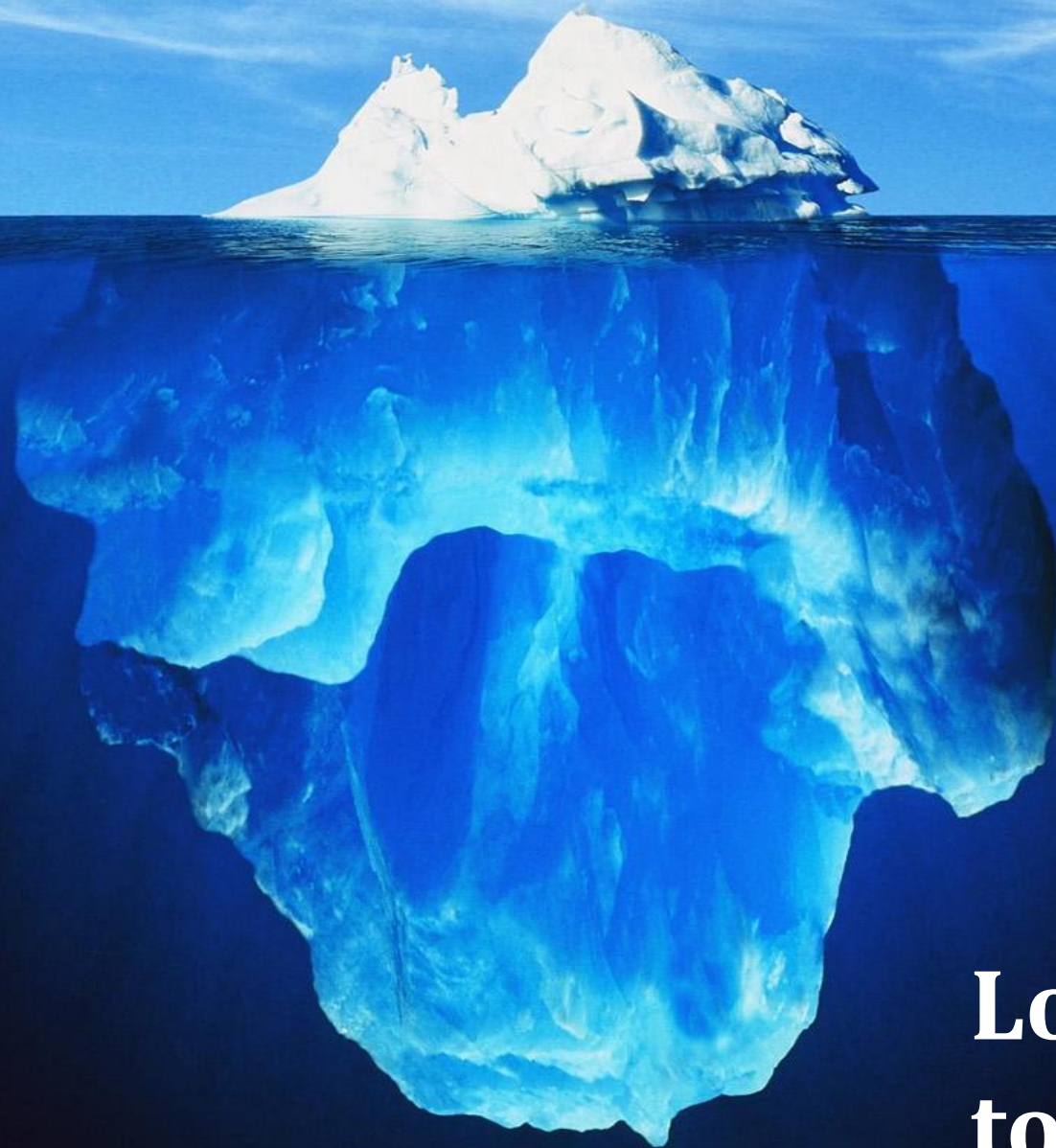
Let  $\bar{f}$  put flow  $\frac{\text{OPT}}{T} \times (\# \text{time } P^{(t)} = P)$  on each path  $P \in \mathcal{R}$ .

Comparable to other algorithms so far. But importantly: It actually extends to many other flow type problems for which Ford-Fulkerson doesn't extend!

**MWU** is the **MVP!**

Even for offline problems, online learning algorithms can be very helpful!

**CS170** →



**Lots and lots  
to learn!**

# Continue to learn about Theory of CS !

To see more ...

- Take more courses
- Come to Theory lunch! Wednesdays at around noon
- Go to the Simons Institute for Theory of Computing on campus.
- Stay in touch with us!

What's Next? #438



Jonathan Pei **STAFF**

3 weeks ago in **Random**



UNPIN



STAR



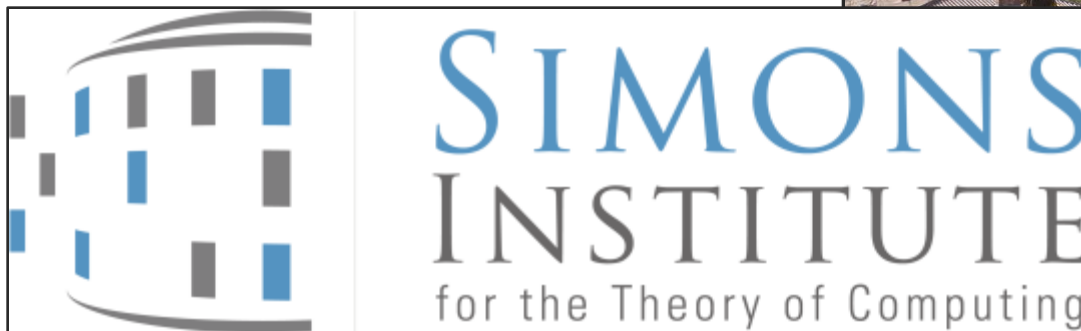
WATCH

138

VIEWS



CS 170 staff (both present and past) have compiled some thoughts about potential courses to take after CS 170. If you have any follow-up questions or questions about other related courses, feel free to ask below! Please note: this



John and Nika want to say a huge thanks to our staff!



Jonny



Carolyn



David W



Eric



Meghal



Andrew



Jessica L



Bill



Ryan



Diana



Xavier



Aaryan



Ajay



Alex



Anushka



David Y



Divya



George



Jeffery



Jessica H



Richik



Shu



Thomas



Yamuna



Will Y

Thank you!