

# From external to swap regret 2.0

An Efficient Reduction for Large Action Spaces

Shahar Glam   Daniel Volkov  
Tel-Aviv University

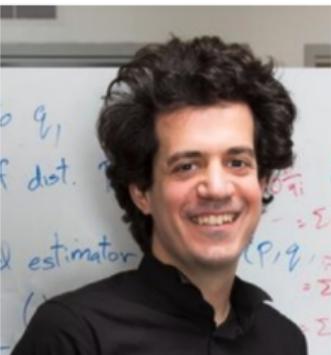
December 9, 2024

# Authors

---



Yuval Dagan



Costis Daskalakis



Maxwell Fishelson



Noah Golowich

# Overview

---

## 1. Introduction

## 2. Preliminaries

- 2.1 Strategic game setting
- 2.2 Types of equilibrium
- 2.3 Online learning setting
- 2.4 Types of regret
- 2.5 No-regret learning

## 3. From external to swap regret

- 3.1 Blum-Mansour algorithm
- 3.2 Tree-swap algorithm
- 3.3 Proof sketch
- 3.4 Pseudo-code

## 4. Results and discussion

## 5. References

# Example - The Prisoner's Dilemma

---

## Example (Prisoner's Dilemma)

- Two bank robbers are arrested and being interrogated.
- Police only has evidence of minor offenses.
- The robbers are interrogated separately and can't communicate with each other.
- Each robber can either cooperate with the police  or remain silent .

# Example - The Prisoner's Dilemma

## Example (Prisoner's Dilemma)

- Two bank robbers are arrested and being interrogated.
- Police only has evidence of minor offenses.
- The robbers are interrogated separately and can't communicate with each other.
- Each robber can either cooperate with the police 🐀 or remain silent 😬.

$A_1 \setminus A_2$	🐀 <sub>2</sub>	😬 <sub>2</sub>
🐀 <sub>1</sub>	(−5, −5)	(0, −20)
😬 <sub>1</sub>	(−20, 0)	(−1, −1)

# Strategic game setting

---

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

# Strategic game setting

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

A [normal form](#) game

$A_1 \setminus A_2$	2	2	2
1	(0, 0)	(-1, 1)	(1, -1)
1	(1, -1)	(0, 0)	(-1, 1)
1	(-1, 1)	(1, -1)	(0, 0)

# Strategic game setting

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

A **normal form** game

$K = 2$

$A_1 \setminus A_2$	2	2	2
1	(0, 0)	(-1, 1)	(1, -1)
1	(1, -1)	(0, 0)	(-1, 1)
1	(-1, 1)	(1, -1)	(0, 0)

# Strategic game setting

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

A **normal form** game

$$K = 2$$

$$A_i = \{\text{Rock}_i, \text{Paper}_i, \text{Scissors}_i\}$$

$A_1 \setminus A_2$	$_2$	$_2$	$_2$
$_1$	(0, 0)	(-1, 1)	(1, -1)
$_1$	(1, -1)	(0, 0)	(-1, 1)
$_1$	(-1, 1)	(1, -1)	(0, 0)

# Strategic game setting

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

A **normal form** game

$A_1 \setminus A_2$	2	2	2
1	(0, 0)	(-1, 1)	(1, -1)
1	(1, -1)	(0, 0)	(-1, 1)
1	(-1, 1)	(1, -1)	(0, 0)

$$\begin{aligned} K &= 2 \\ A_i &= \{\text{Rock}_i, \text{Paper}_i, \text{Scissors}_i\} \\ A &= \{(\text{Rock}, \text{Rock}), (\text{Paper}, \text{Paper}), \\ &\quad (\text{Scissors}, \text{Scissors}), (\text{Rock}, \text{Paper}), (\text{Paper}, \text{Rock}), (\text{Scissors}, \text{Rock}), \\ &\quad (\text{Rock}, \text{Scissors}), (\text{Scissors}, \text{Paper}), \dots\} \end{aligned}$$

# Strategic game setting

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

A **normal form** game

$A_1 \setminus A_2$	2	2	2
1	(0, 0)	(-1, 1)	(1, -1)
1	(1, -1)	(0, 0)	(-1, 1)
1	(-1, 1)	(1, -1)	(0, 0)

$$\begin{aligned} K &= 2 \\ A_i &= \{\text{Rock}_i, \text{Paper}_i, \text{Scissors}_i\} \\ A &= \{(\text{Rock}, \text{Rock}), (\text{Paper}, \text{Paper}), \\ &\quad (\text{Scissors}, \text{Scissors}), (\text{Rock}, \text{Paper}), (\text{Paper}, \text{Rock}), (\text{Scissors}, \text{Rock}), (\text{Rock}, \text{Scissors}), \\ &\quad (\text{Scissors}, \text{Paper}), (\text{Paper}, \text{Scissors})\} \\ f(\text{Rock}, \text{Rock}) &= f(\text{Scissors}, \text{Scissors}) = f(\text{Paper}, \text{Paper}) = (0, 0), \dots \end{aligned}$$

# Strategic game setting

## Definition (Strategic game)

A strategic game is a triplet  $\langle K, A, f \rangle$ , where  $A = \times_{i=1}^K A_i$  is the set of all player's actions, and  $f = (f_1, \dots, f_K) : A \rightarrow \mathbb{R}^K$  is the payoff (or reward) function.

## Definition (Strategy profile)

$s = (s_i; s_{-i}) = (s_1, \dots, s_n) \in A$  is called a strategy profile, where  $s_i \in A_i$  is the strategy (or action) of player  $i$

# Equilibrium

---

## Game Equilibrium

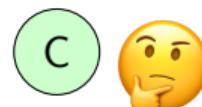
Strategy profile where no player wants to deviate



# Equilibrium

## Game Equilibrium

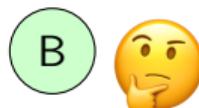
Strategy profile where no player wants to deviate



# Equilibrium

## Game Equilibrium

Strategy profile where no player wants to deviate



# Equilibrium

## Game Equilibrium

Strategy profile where no player wants to deviate



## Coarse Deviations

A	Z
B	Z
C	Z

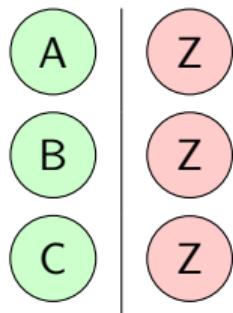
# Equilibrium

## Game Equilibrium

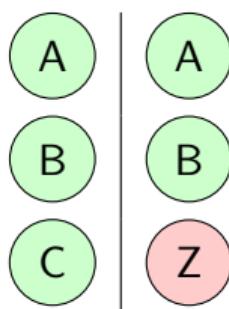
Strategy profile where no player wants to deviate



Coarse Deviations



Swap Deviations



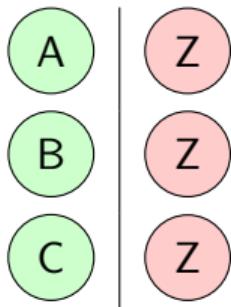
# Equilibrium

## Game Equilibrium

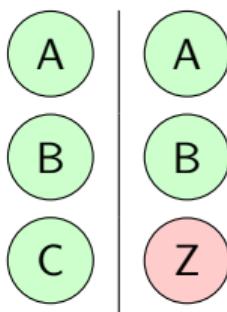
Strategy profile where no player wants to deviate



### Coarse Deviations



### Swap Deviations



**CCE:** No player wants to make a coarse deviation

**CE:** No player wants to make a swap deviation

**NE:** No player wants to make a swap deviation

**AND** profile must be independent

# Nash Equilibrium

---

## Definition (Nash equilibrium)

A strategy profile  $s^*$  in a game  $\langle K, A, f \rangle$  is called a **Nash equilibrium** if for all  $i \in [K]$ :

$$\forall d \in A_i : f_i(s_i^*; s_{-i}^*) \geq f_i(d; s_{-i}^*)$$

# Nash Equilibrium

## Definition (Nash equilibrium)

A strategy profile  $s^*$  in a game  $\langle K, A, f \rangle$  is called a **Nash equilibrium** if for all  $i \in [K]$ :

$$\forall d \in A_i : f_i(s_i^*; s_{-i}^*) \geq f_i(d; s_{-i}^*)$$

$A_1 \setminus A_2$	 2	 2
 1	(2, 1)	(0, 0)
 1	(0, 0)	(1, 2)

# Nash Equilibrium

## Definition (Nash equilibrium)

A strategy profile  $s^*$  in a game  $\langle K, A, f \rangle$  is called a **Nash equilibrium** if for all  $i \in [K]$ :

$$\forall d \in A_i : f_i(s_i^*; s_{-i}^*) \geq f_i(d; s_{-i}^*)$$

$A_1 \setminus A_2$	 2	 2
 1	(2, 1)	(0, 0)
 1	(0, 0)	(1, 2)

# Correlated Equilibrium (CE)

## Definition (Correlated equilibrium)

Let  $Q$  be a **distribution** over  $A$ .  $Q$  is called a Correlated equilibrium, if for all  $i \in [K]$ , and for every function  $\phi : A_i \rightarrow A_i$ :

$$\mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

# Correlated Equilibrium (CE)

## Definition (Correlated equilibrium)

Let  $Q$  be a **distribution** over  $A$ .  $Q$  is called a Correlated equilibrium, if for all  $i \in [K]$ , and for every function  $\phi : A_i \rightarrow A_i$ :

$$\mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

## Definition ( $\varepsilon$ -approximate-CE)

$Q$  is called an  $\varepsilon$ -Correlated equilibrium if for all  $i \in [K]$ , and for every function  $\phi : A_i \rightarrow A_i$ :

$$\varepsilon + \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

# Correlated Equilibrium (CE)

## Definition (Correlated equilibrium)

Let  $Q$  be a **distribution** over  $A$ .  $Q$  is called a Correlated equilibrium, if for all  $i \in [K]$ , and for every function  $\phi : A_i \rightarrow A_i$ :

$$\mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

$A_1 \setminus A_2$	 $_2$	 $_2$
 $_1$	(−100, −100) −1, 0)	(0, −1) −1, −1)
 $_1$		

# Correlated Equilibrium (CE)

## Definition (Correlated equilibrium)

Let  $Q$  be a **distribution** over  $A$ .  $Q$  is called a Correlated equilibrium, if for all  $i \in [K]$ , and for every function  $\phi : A_i \rightarrow A_i$ :

$$\mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

$A_1 \setminus A_2$	 2	 2
 1	(−100, −100) <span style="border: 1px solid red; padding: 2px;">(−1, 0)</span>	(0, −1) <span style="border: 1px solid red; padding: 2px;">(−1, −1)</span>
 1		

# Correlated Equilibrium (CE)

## Definition (Correlated equilibrium)

Let  $Q$  be a **distribution** over  $A$ .  $Q$  is called a Correlated equilibrium, if for all  $i \in [K]$ , and for every function  $\phi : A_i \rightarrow A_i$ :

$$\mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

$A_1 \setminus A_2$	 $_2$	 $_2$
 $_1$	$(-100, -100)$	$(0, -1)$
 $_1$	$(-1, 0)$	$(-1, -1)$

**Correlated Equilibrium:**  
 $p(S, C) = \frac{1}{2}$ ,  $p(C, S) = \frac{1}{2}$ ,  
 $p(C, C) = p(S, S) = 0$

# Coarse Correlated Equilibrium (CCE)

## Definition (Coarse Correlated equilibrium)

Let  $Q$  be a **distribution** over  $A$ .  $Q$  is called a Coarse Correlated equilibrium, if for all  $i \in [K]$ , and for all  $d \in A_i$ :

$$\mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

## Definition ( $\varepsilon$ -CCE)

$Q$  is called an  $\varepsilon$ -Coarse Correlated equilibrium if for all  $i \in [K]$ , and for all  $d \in A_i$ :

$$\varepsilon + \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

# CE VS CCE

---

Definition (Correlated equilibrium)

$$\forall \phi : A_i \rightarrow A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

Definition (Coarse Correlated equilibrium)

$$\forall d \in A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

# CE VS CCE

Definition (Correlated equilibrium)

$$\forall \phi : A_i \rightarrow A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

Definition (Coarse Correlated equilibrium)

$$\forall d \in A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

$A_1 \setminus A_2$	A	B	C
A	(1, -1)	(-1, 1)	$(-\infty, -\infty)$
B	(-1, 1)	(1, -1)	$(-\infty, -\infty)$
C	$(-\infty, -\infty)$	$(-\infty, -\infty)$	(-10, -10)

# CE VS CCE

Definition (Correlated equilibrium)

$$\forall \phi : A_i \rightarrow A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

Definition (Coarse Correlated equilibrium)

$$\forall d \in A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

$A_1 \setminus A_2$	A	B	C
A	(1, -1)	(-1, 1)	$(-\infty, -\infty)$
B	(-1, 1)	(1, -1)	$(-\infty, -\infty)$
C	$(-\infty, -\infty)$	$(-\infty, -\infty)$	<span style="border: 2px solid red;">(-10, -10)</span>

# CE VS CCE

Definition (Correlated equilibrium)

$$\forall \phi : A_i \rightarrow A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

Definition (Coarse Correlated equilibrium)

$$\forall d \in A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

$A_1 \setminus A_2$	A	B	C
A	(1, -1)	(-1, 1)	$(-\infty, -\infty)$
B	(-1, 1)	(1, -1)	$(-\infty, -\infty)$
C	$(-\infty, -\infty)$	$(-\infty, -\infty)$	<span style="border: 2px solid red; padding: 2px;">(-10, -10)</span>

CE

$$\begin{aligned} P(A, B) &= P(B, A) = \\ P(A, A) &= P(B, B) = \frac{1}{8} \\ P(C, C) &= \frac{1}{2} \end{aligned}$$

# CE VS CCE

Definition (Correlated equilibrium)

$$\forall \phi : A_i \rightarrow A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(\phi(s_i); s_{-i})]$$

Definition (Coarse Correlated equilibrium)

$$\forall d \in A_i : \mathbb{E}_{s \sim Q} [f_i(s)] \geq \mathbb{E}_{s \sim Q} [f_i(d; s_{-i})]$$

$A_1 \setminus A_2$	A	B	C
A	(1, -1)	(-1, 1)	$(-\infty, -\infty)$
B	(-1, 1)	(1, -1)	$(-\infty, -\infty)$
C	$(-\infty, -\infty)$	$(-\infty, -\infty)$	<span style="border: 2px solid red; padding: 2px;">(-10, -10)</span>

CE

$$P(A, B) = P(B, A) = \\ P(A, A) = P(B, B) = \frac{1}{8}$$

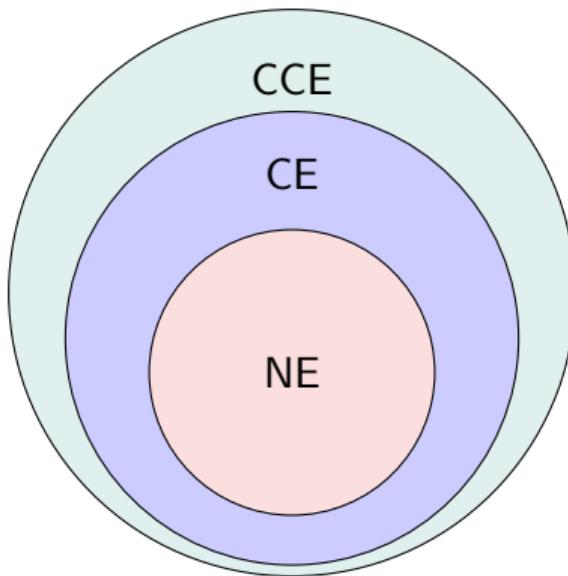
$$P(C, C) = \frac{1}{2}$$

CCE

$$P(A, A) = P(C, C) = \frac{1}{2}$$

# Hierarchy Of Equilibrium

---



# Online learning setting

---

A **single player** plays against an adversary for  $T$  rounds trying to maximize their **reward**.  
In each round  $t = 1, \dots, T$ :

# Online learning setting

---

A **single player** plays against an adversary for  $T$  rounds trying to maximize their **reward**. In each round  $t = 1, \dots, T$ :

- The player picks a probability distribution  $x^{(t)} = (x_1^{(t)}, \dots, x_N^{(t)})$  over  $A$ .

# Online learning setting

---

A **single player** plays against an adversary for  $T$  rounds trying to maximize their **reward**. In each round  $t = 1, \dots, T$ :

- The player picks a probability distribution  $x^{(t)} = (x_1^{(t)}, \dots, x_N^{(t)})$  over  $A$ .
- The adversary picks a reward vector  $f^{(t)} = (f_1^{(t)}, \dots, f_N^{(t)}) \in [0, 1]^N$ .

# Online learning setting

---

A **single player** plays against an adversary for  $T$  rounds trying to maximize their **reward**. In each round  $t = 1, \dots, T$ :

- The player picks a probability distribution  $x^{(t)} = (x_1^{(t)}, \dots, x_N^{(t)})$  over  $A$ .
- The adversary picks a reward vector  $f^{(t)} = (f_1^{(t)}, \dots, f_N^{(t)}) \in [0, 1]^N$ .
- The player achieves  $x^{(t)} \cdot f^{(t)}$  reward at round  $t$ .

# Regret in online learning

---

## Regret

The difference between the obtained reward and a benchmark

# Regret in online learning

---

## Regret

The difference between the obtained reward and a benchmark

$$\sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

# Regret in online learning

## Regret

The difference between the obtained reward and a benchmark

$$\sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

—————  
Benchmark: best  $\phi \in \Phi$

$$\sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)}$$

# Regret in online learning

## Regret

The difference between the obtained reward and a benchmark

$$\sum_{t=1}^T x^{(t)} \cdot f^{(t)} \xrightarrow{\text{Benchmark: best } \phi \in \Phi} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)}$$

- Where  $\Phi$  is a set of transformations  $\phi : \Delta_N \rightarrow \Delta_N$

# Regret in online learning

## Regret

The difference between the obtained reward and a benchmark

$$\sum_{t=1}^T x^{(t)} \cdot f^{(t)} \xrightarrow{\text{Benchmark: best } \phi \in \Phi} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)}$$

- Where  $\Phi$  is a set of transformations  $\phi : \Delta_N \rightarrow \Delta_N$

## Definition (Regret against $\Phi$ )

The regret of an online learner is defined as:

$$\text{Regret}_\Phi(x^{(1:T)}; f^{(1:T)}) = \left( \max_{\phi \in \Phi} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

# Swap Regret

## Definition (Swap Regret)

The **swap regret** of an online learner is defined as the regret against the set of **all maps**, namely:

$$\text{SR}(x^{(1:T)}; f^{(1:T)}) = \left( \max_{\phi: \Delta_N \rightarrow \Delta_N} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

# Swap Regret

## Definition (Swap Regret)

The **swap regret** of an online learner is defined as the regret against the set of **all maps**, namely:

$$\text{SR}(x^{(1:T)}; f^{(1:T)}) = \left( \max_{\phi: \Delta_N \rightarrow \Delta_N} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

## Theorem ( $\varepsilon$ -SR induces an $\varepsilon$ -CE)

*If each player achieves swap regret bounded above by  $\varepsilon$ , then the joint distribution of all player actions is an  $\varepsilon$ -Correlated equilibrium.*

# Swap Regret

## Definition (Swap Regret)

The **swap regret** of an online learner is defined as the regret against the set of **all maps**, namely:

$$\text{SR}(x^{(1:T)}; f^{(1:T)}) = \left( \max_{\phi: \Delta_N \rightarrow \Delta_N} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

## Theorem ( $\varepsilon$ -SR induces an $\varepsilon$ -CE)

*If each player achieves swap regret bounded above by  $\varepsilon$ , then the joint distribution of all player actions is an  $\varepsilon$ -Correlated equilibrium.*

- **Proof sketch:** for each player, the maximum over all maps is at least as big as over a **swap function**. If we now define the same swap on the distribution, we get the definition of  $\varepsilon$ -CE.

## $\varepsilon$ -SR induces an $\varepsilon$ -CE

---

Assume each player achieves swap-regret bounded by  $\varepsilon$ .

## $\varepsilon$ -SR induces an $\varepsilon$ -CE

---

Assume each player achieves swap-regret bounded by  $\varepsilon$ . namely  $\forall i \in [K]$  and  $\forall \phi^* : \Delta_N \rightarrow \Delta_N$ :

$$\mathbb{E}_{s \sim \phi^*(X)} [f_i(s)] - \mathbb{E}_{s \sim X} [f_i(s)] \leq \left( \max_{\phi: \Delta_N \rightarrow \Delta_N} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)} \leq \varepsilon$$

## $\varepsilon$ -SR induces an $\varepsilon$ -CE

Assume each player achieves swap-regret bounded by  $\varepsilon$ . namely  $\forall i \in [K]$  and  $\forall \phi^* : \Delta_N \rightarrow \Delta_N$ :

$$\mathbb{E}_{s \sim \phi^*(X)} [f_i(s)] - \mathbb{E}_{s \sim X} [f_i(s)] \leq \left( \max_{\phi: \Delta_N \rightarrow \Delta_N} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)} \leq \varepsilon$$

Let  $\phi : A_i \rightarrow A_i$  be a swap function. Define  $\hat{\phi} : \Delta_N \rightarrow \Delta_N$  such that  $x \sim X \iff \phi(x) \sim \hat{\phi}(X)$ .

## $\varepsilon$ -SR induces an $\varepsilon$ -CE

Assume each player achieves swap-regret bounded by  $\varepsilon$ . namely  $\forall i \in [K]$  and  $\forall \phi^* : \Delta_N \rightarrow \Delta_N$ :

$$\mathbb{E}_{s \sim \phi^*(X)} [f_i(s)] - \mathbb{E}_{s \sim X} [f_i(s)] \leq \left( \max_{\phi: \Delta_N \rightarrow \Delta_N} \sum_{t=1}^T \phi(x^{(t)}) \cdot f^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)} \leq \varepsilon$$

Let  $\phi : A_i \rightarrow A_i$  be a swap function. Define  $\hat{\phi} : \Delta_N \rightarrow \Delta_N$  such that  $x \sim X \iff \phi(x) \sim \hat{\phi}(X)$ .

We have:

$$\mathbb{E}_{s \sim X} [f_i(\phi(s_i); s_{-i})] - \mathbb{E}_{s \sim X} [f_i(s)] = \mathbb{E}_{s \sim \hat{\phi}(X)} [f_i(s)] - \mathbb{E}_{s \sim X} [f_i(s)] \leq \varepsilon$$

# External regret

## Definition (External regret)

The **external regret** of an online learner is defined as the regret against the set of all **constant maps**. (Map  $x^{(t)}$  to a distribution where  $\exists i \in [N] : P(i) = 1$ )

Namely:

$$\text{ER}(x^{(1:T)}, f^{(1:T)}) = \left( \max_{i \in [N]} \sum_{t=1}^T f_i^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

# External regret

## Definition (External regret)

The **external regret** of an online learner is defined as the regret against the set of all **constant maps**. (Map  $x^{(t)}$  to a distribution where  $\exists i \in [N] : P(i) = 1$ )

Namely:

$$\text{ER}(x^{(1:T)}, f^{(1:T)}) = \left( \max_{i \in [N]} \sum_{t=1}^T f_i^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

- External regret minimization is a much(**!!**) easier task than swap regret minimization

# External regret

## Definition (External regret)

The **external regret** of an online learner is defined as the regret against the set of all **constant maps**. (Map  $x^{(t)}$  to a distribution where  $\exists i \in [N] : P(i) = 1$ )

Namely:

$$\text{ER}(x^{(1:T)}, f^{(1:T)}) = \left( \max_{i \in [N]} \sum_{t=1}^T f_i^{(t)} \right) - \sum_{t=1}^T x^{(t)} \cdot f^{(t)}$$

- External regret minimization is a much(**!!**) easier task than swap regret minimization
- This is enough to guarantee swap regret minimization! (*foreshadowing* 

# No-external-regret learning

---

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

# No-external-regret learning

---

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

# No-external-regret learning

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	

# No-external-regret learning

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$

# No-external-regret learning

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	

# No-external-regret learning

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(1)})$

# No-external-regret learning

## Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization)

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(1)})$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(3)})$
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(2)}$	

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$

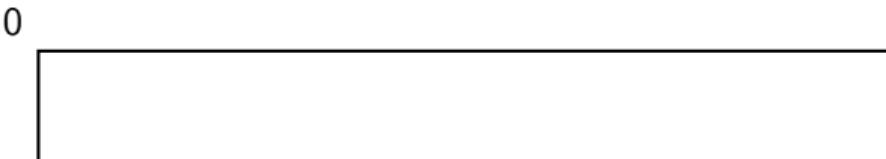
# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$



# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	T
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$	1	
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$	2	
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$	3	
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$	$\dots$	
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		$M$

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$	1	
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$	2	
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$	3	
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$	$\cdots$	
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		$M$

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$	$F^{(0)}$	

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		

The diagram shows a timeline from 0 to  $T = BM$ . It features a horizontal bar divided into segments by vertical red lines. The first segment contains two points labeled  $x^{(1)}$  and  $x^{(1)}$ . Below the bar, the labels  $F^{(0)}$  and  $F^{(B)}$  are positioned such that they align with the start of the first segment and the point before the second vertical line respectively.

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		

The diagram illustrates the sequence of actions taken over time. It shows a timeline from  $F^{(0)}$  to  $F^{(B)}$ , where  $F^{(B)}$  is the final state after  $B$  rounds. The actions  $x^{(1)}, x^{(2)}, \dots, x^{(B)}$  are grouped into blocks of length  $B$ . The first block contains  $x^{(1)}, x^{(2)}$ . The second block contains  $x^{(B)}, x^{(B)}$ . There are vertical red lines separating these blocks, and horizontal red lines extending from the vertical lines to the right, indicating the continuation of the sequence.

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		



The diagram illustrates the sequence of actions over time. A horizontal timeline is divided into segments by vertical red lines. The first segment contains actions  $x^{(1)}$  and  $x^{(1)}$ . The second segment contains actions  $x^{(B)}$  and  $x^{(B)}$ . The third segment is empty. The fourth segment is empty. The fifth segment is empty. Below the timeline, the segments are labeled  $F^{(0)}$ ,  $F^{(B)}$ ,  $F^{(2B)}$ , and so on, indicating the cumulative sum of rewards.

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$		
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$		
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$		

The diagram illustrates the sequence of actions over time. A horizontal timeline is shown with vertical red lines indicating updates every  $B$  rounds. The timeline is divided into segments labeled  $F^{(0)}$ ,  $F^{(B)}$ ,  $F^{(2B)}$ , and so on. Within each segment, multiple actions are taken, labeled  $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}$ . The final label  $T = BM$  indicates the total number of rounds.

# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions	0	$T = BM$
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$	$x^{(1)}$	$x^{(1)}$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$	$x^{(B)}$	$x^{(B)}$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$	$x^{(2B)}$	$x^{(2B)}$
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$	$x^{(3B)}$	$x^{(3B)}$
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$	$\dots$	

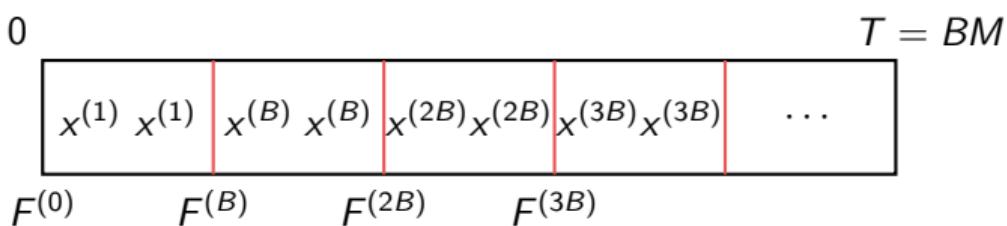
# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$



MWU:  $\varepsilon$ -ER for  $T = \Omega\left(\frac{\log N}{\varepsilon^2}\right)$

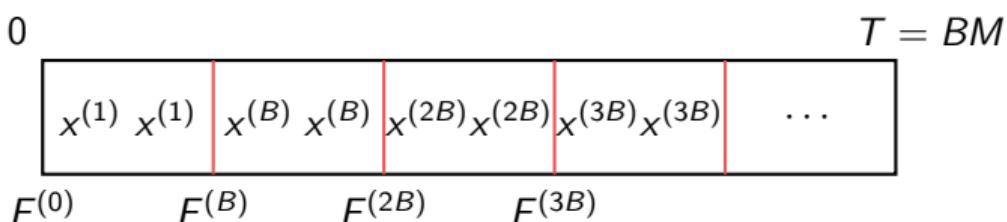
# No-external-regret learning

## Lazy Multiplicative Weight Updates

Look at the past, do the best thing (with some normalization) **Every B rounds**

$$F^{(t)} := \frac{1}{t} \sum_{i=1}^t f^{(i)}$$

Rewards	Actions
$F^{(0)}$	$x^{(1)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(1)}$	$x^{(2)} = \text{softmax}(\eta \cdot F^{(0)})$
$F^{(2)}$	$x^{(3)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(3)}$	$x^{(4)} = \text{softmax}(\eta \cdot F^{(2)})$
$F^{(4)}$	$x^{(5)} = \text{softmax}(\eta \cdot F^{(4)})$



MWU:  $\varepsilon$ -ER for  $T = \Omega\left(\frac{\log N}{\varepsilon^2}\right)$

**Lazy** MWU:  $\varepsilon$ -ER for  $M = \Omega\left(\frac{\log N}{\varepsilon^2}\right)$

# Blum-Mansour algorithm

---

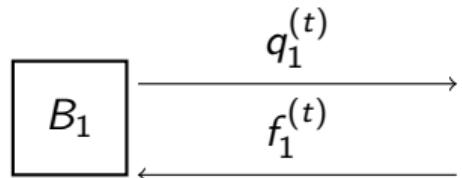
$B_1$

⋮

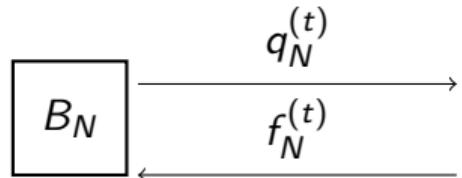
$B_N$

# Blum-Mansour algorithm

---

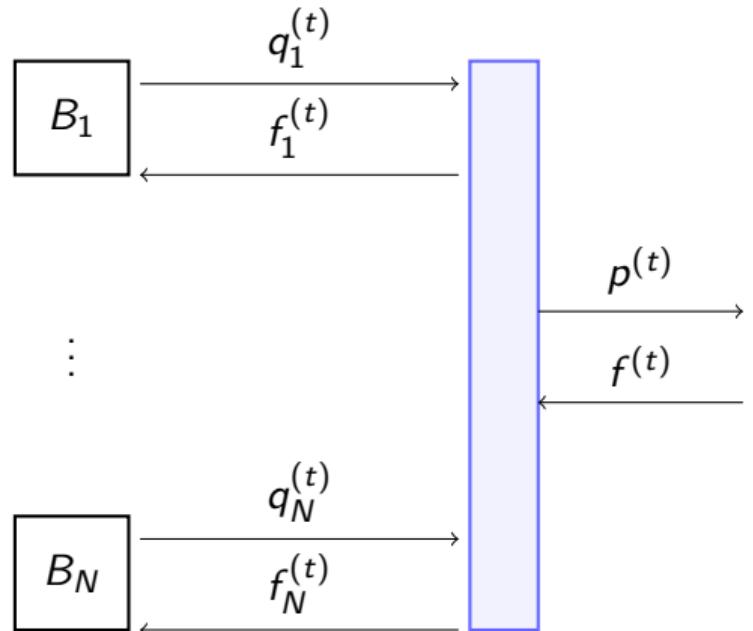


⋮

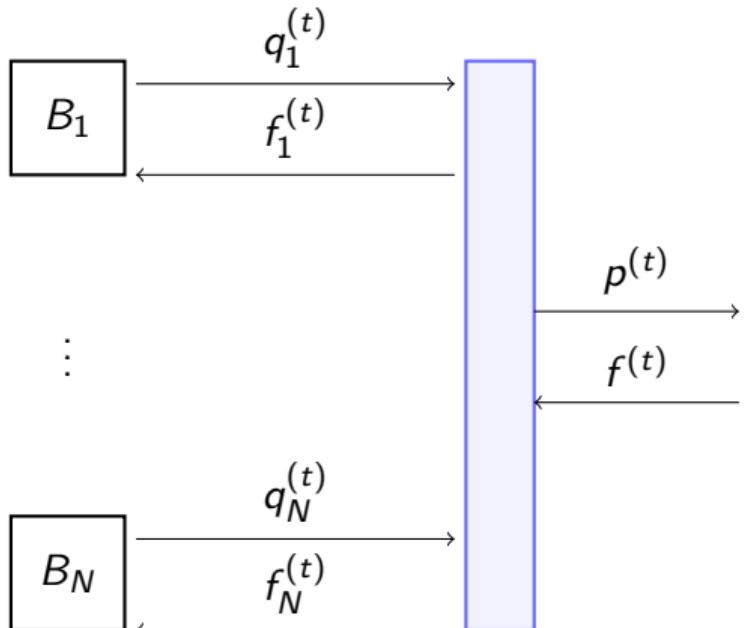


# Blum-Mansour algorithm

---

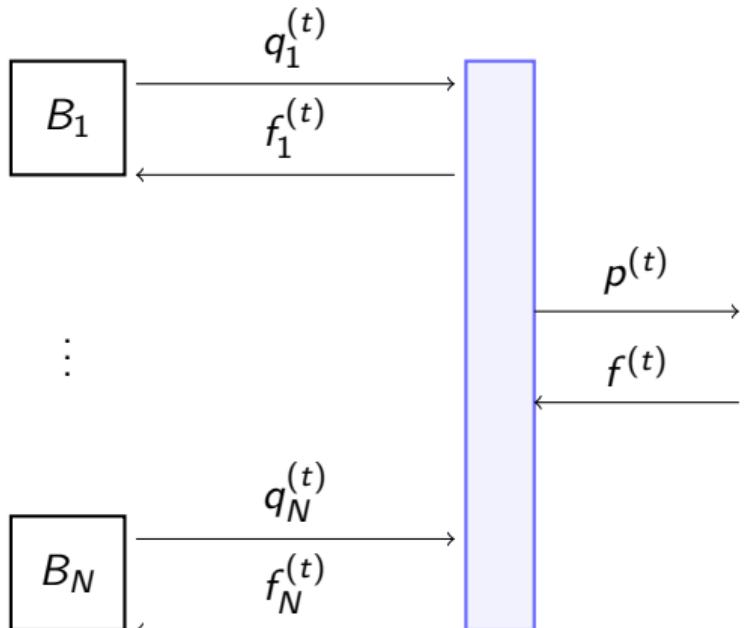


# Blum-Mansour algorithm



i) Calculating  $p^{(t)}$  from  $\vec{q}_1^{(t)}, \dots, \vec{q}_N^{(t)}$ .

# Blum-Mansour algorithm



- i) Calculating  $p^{(t)}$  from  $\vec{q}_1^{(t)}, \dots, \vec{q}_N^{(t)}$ .
- ii) Distributing the reward across the algorithms  $B_1, \dots, B_N$ .

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

**Calculate p: LATER!**

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

**Calculate p: LATER!**

Denote:

$$Q = \begin{pmatrix} q_{11} & \dots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{N1} & \dots & q_{NN} \end{pmatrix} = \begin{pmatrix} \vec{q}_1^{(t)} \\ \vdots \\ \vec{q}_N^{(t)} \end{pmatrix}$$

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

**Calculate p: LATER!**

Denote:

$$Q = \begin{pmatrix} q_{11} & \dots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{N1} & \dots & q_{NN} \end{pmatrix} = \begin{pmatrix} \vec{q}_1^{(t)} \\ \vdots \\ \vec{q}_N^{(t)} \end{pmatrix}$$

$F^{(t)} :=$  Alg reward at time  $t$

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

**Calculate p: LATER!**

Denote:

$$Q = \begin{pmatrix} q_{11} & \dots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{N1} & \dots & q_{NN} \end{pmatrix} = \begin{pmatrix} \vec{q}_1^{(t)} \\ \vdots \\ \vec{q}_N^{(t)} \end{pmatrix}$$

$F^{(t)}$  := Alg reward at time  $t$

$F^T := \sum_{i=1}^T F^{(t)}$

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

**Calculate p: LATER!**

Denote:

$$Q = \begin{pmatrix} q_{11} & \dots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{N1} & \dots & q_{NN} \end{pmatrix} = \begin{pmatrix} \vec{q}_1^{(t)} \\ \vdots \\ \vec{q}_N^{(t)} \end{pmatrix}$$

$F^{(t)}$  := Alg reward at time  $t$

$F^T := \sum_{i=1}^T F^{(t)}$

$F_{B_i}^{(t)}$  :=  $B_i$  reward at time  $t$

# Blum-Mansour algorithm

---

**Distribute:**  $B_i \leftarrow f_i^{(t)} = f^{(t)} \cdot p_i^{(t)}$

**Calculate p: LATER!**

Denote:

$$Q = \begin{pmatrix} q_{11} & \dots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{N1} & \dots & q_{NN} \end{pmatrix} = \begin{pmatrix} \vec{q}_1^{(t)} \\ \vdots \\ \vec{q}_N^{(t)} \end{pmatrix}$$

$F^{(t)}$  := Alg reward at time  $t$

$F^T := \sum_{i=1}^T F^{(t)}$

$F_{B_i}^{(t)} := B_i$  reward at time  $t$

$F_{B_i}^T := \sum_{i=1}^T F_{B_i}^{(t)}$

# Proof.

---

$B_i$ 's Reward is:

$$F_{B_i}^{(t)} = (p_i^{(t)} f^{(t)}) \cdot \vec{q}_i^{(t)} = p_i^{(t)} (f^{(t)} \cdot \vec{q}_i^{(t)})$$

# Proof.

---

$B_i$ 's Reward is:

$$F_{B_i}^{(t)} = (p_i^{(t)} f^{(t)}) \cdot \vec{q}_i^{(t)} = p_i^{(t)} (f^{(t)} \cdot \vec{q}_i^{(t)})$$

$B_i$  Is a no ER Algorithm!

# Proof.

---

$B_i$ 's Reward is:

$$F_{B_i}^{(t)} = (p_i^{(t)} f^{(t)}) \cdot \vec{q}_i^{(t)} = p_i^{(t)} (f^{(t)} \cdot \vec{q}_i^{(t)})$$

$B_i$  Is a no ER Algorithm!

$$\varepsilon + F_{B_i}^T = \varepsilon + \sum_{t=1}^T p_i^{(t)} (\vec{q}_i^{(t)} \cdot f^{(t)}) \geq \sum_{t=1}^T p_i^{(t)} \cdot f_{\phi(j)}^{(t)}$$

# Proof.

---

$B_i$ 's Reward is:

$$F_{B_i}^{(t)} = (p_i^{(t)} f^{(t)}) \cdot \vec{q}_i^{(t)} = p_i^{(t)} (f^{(t)} \cdot \vec{q}_i^{(t)})$$

$B_i$  Is a no ER Algorithm!

$$\varepsilon + F_{B_i}^T = \varepsilon + \sum_{t=1}^T p_i^{(t)} (\vec{q}_i^{(t)} \cdot f^{(t)}) \geq \sum_{t=1}^T p_i^{(t)} \cdot f_{\phi(j)}^{(t)}$$

For every  $t \in [T]$ :

$$\sum_{i=1}^N (p_i^{(t)} \cdot \vec{q}_i^{(t)}) f^{(t)} = \sum_{i=1}^N p_i^{(t)} (\vec{q}_i^{(t)} \cdot f^{(t)}) = p^{(t)} Q f^{(t)} = \underbrace{p^{(t)} \cdot f^{(t)}}_{\text{Choose } p = pQ} = F^{(t)}$$

# Proof.

---

$B_i$ 's Reward is:

$$F_{B_i}^{(t)} = (p_i^{(t)} f^{(t)}) \cdot \vec{q}_i^{(t)} = p_i^{(t)} (f^{(t)} \cdot \vec{q}_i^{(t)})$$

$B_i$  Is a no ER Algorithm!

$$\varepsilon + F_{B_i}^T = \varepsilon + \sum_{t=1}^T p_i^{(t)} (\vec{q}_i^{(t)} \cdot f^{(t)}) \geq \sum_{t=1}^T p_i^{(t)} \cdot f_{\phi(j)}^{(t)}$$

For every  $t \in [T]$ :

$$\sum_{i=1}^N (p_i^{(t)} \cdot \vec{q}_i^{(t)}) f^{(t)} = \sum_{i=1}^N p_i^{(t)} (\vec{q}_i^{(t)} \cdot f^{(t)}) = p^{(t)} Q f^{(t)} = \underbrace{p^{(t)} \cdot f^{(t)}}_{\text{Choose } p = pQ} = F^{(t)}$$

Hence our total reward is:

$$\varepsilon + F^T = \varepsilon + \sum_{i=1}^N F_{B_i}^T \geq \sum_{i=1}^N \sum_{t=1}^T p_i^{(t)} \cdot f_{\phi(i)}^{(t)}$$

# Tree-Swap

---

Theorem (From external to swap regret)

Let  $M, d \in \mathbb{N}$ , and let  $\mathcal{F}$  be a function class.

If there exists a learner for  $\mathcal{F}$  that achieves  $\varepsilon$  **external-regret** after  $M$  rounds, then there is a learner for  $\mathcal{F}$  that achieves **swap-regret** of at most  $\varepsilon + \frac{1}{d}$  after  $T = M^d$  rounds.

# Tree-Swap

---

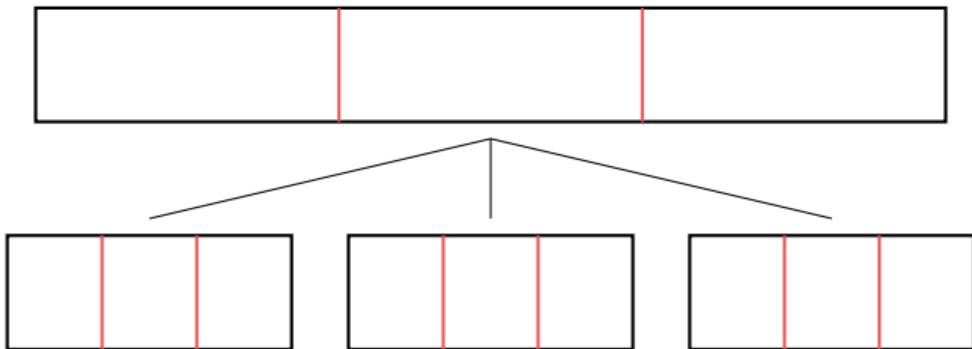
$$T = M^d$$



# Tree-Swap

---

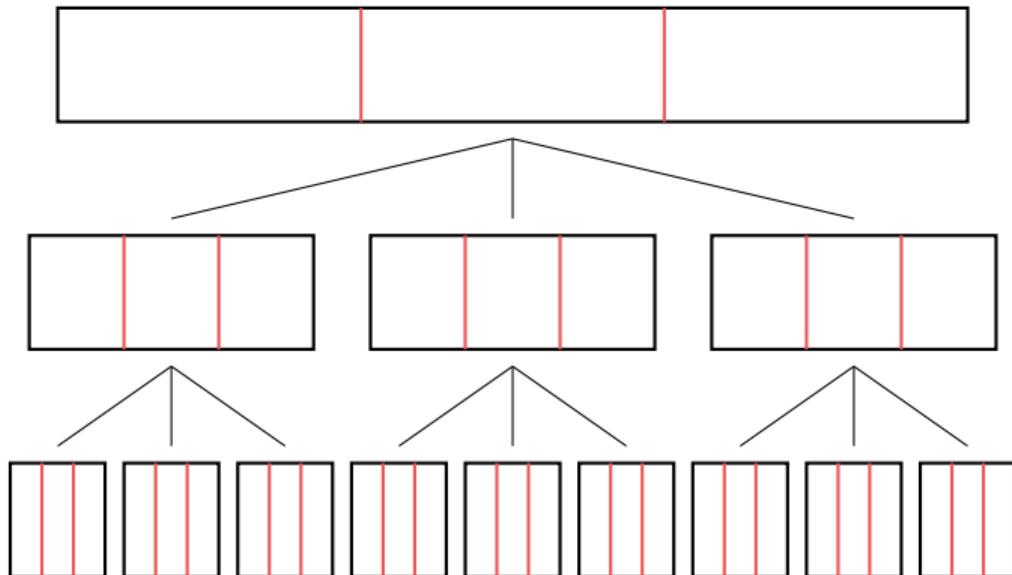
$$T = M^d$$



# Tree-Swap

---

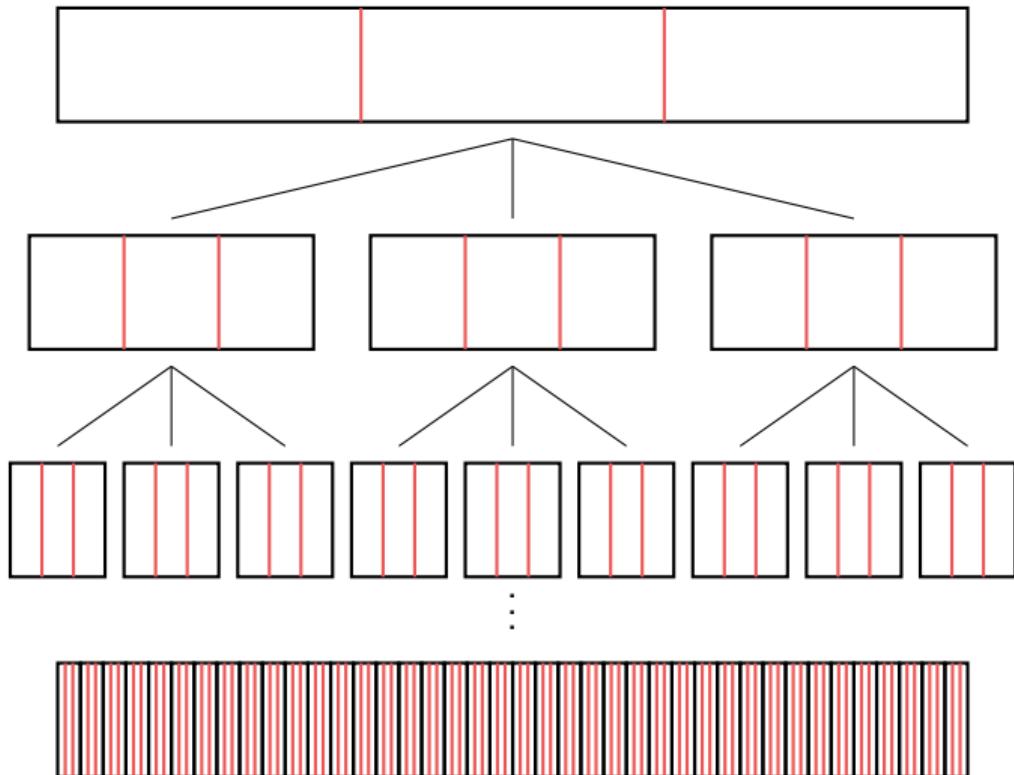
$$T = M^d$$



# Tree-Swap

---

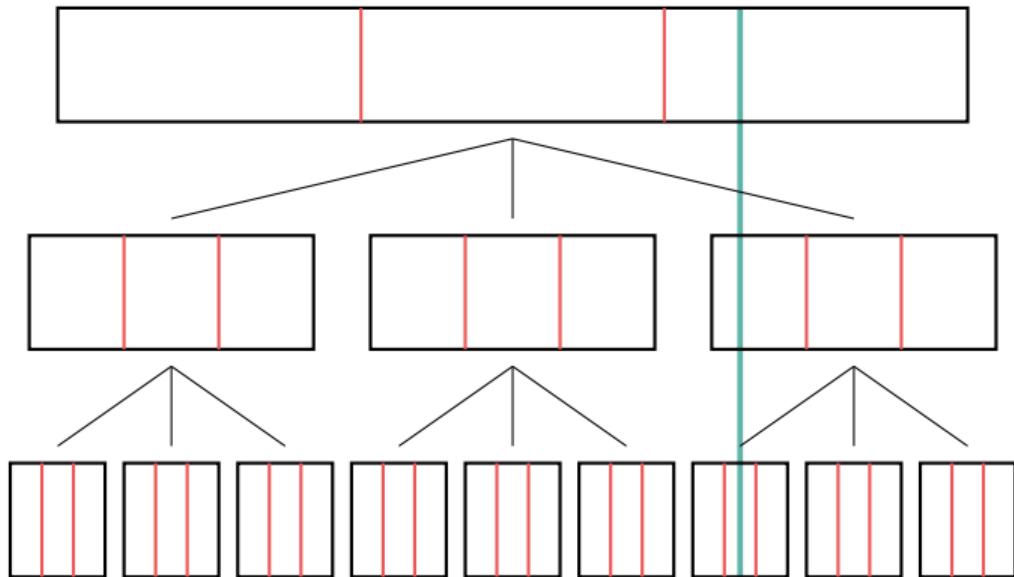
$$T = M^d$$



# Tree-Swap

---

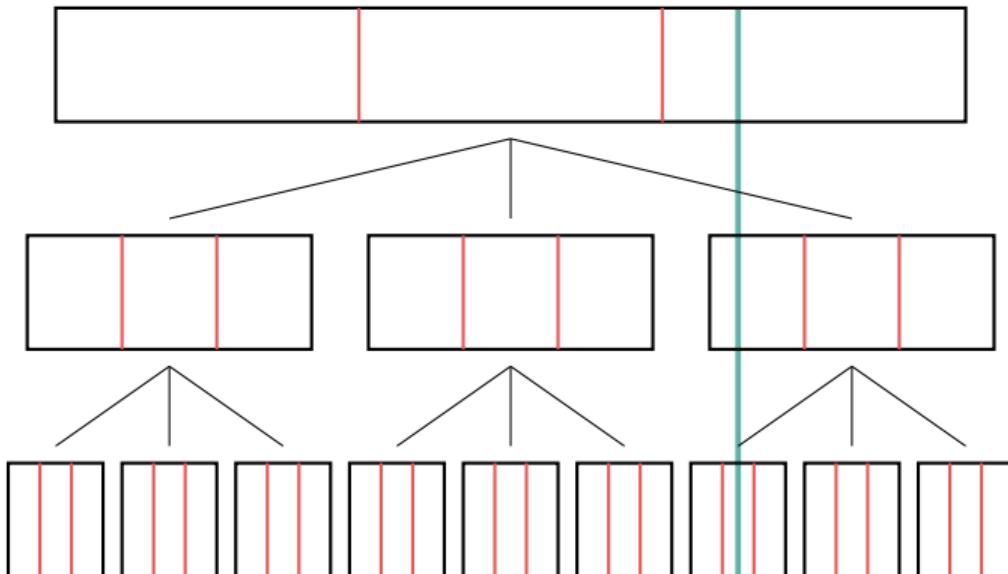
$$T = M^d$$



# Tree-Swap

---

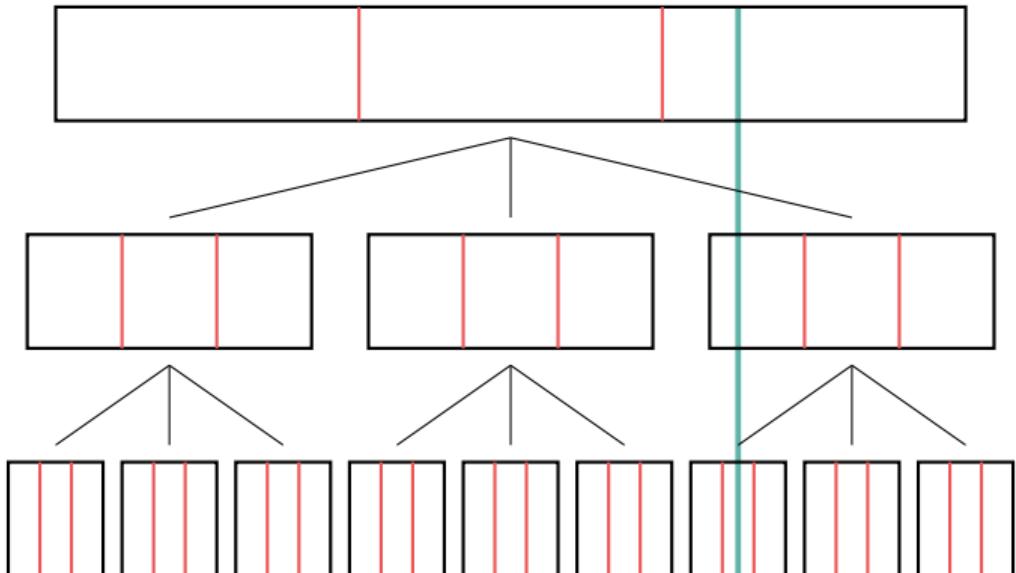
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .

# Tree-Swap

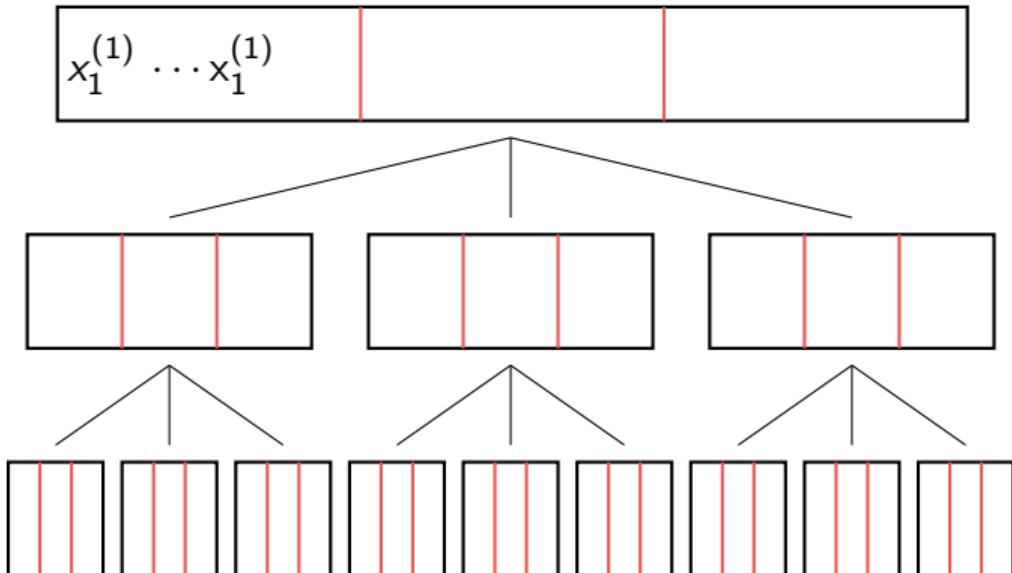
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Tree-Swap

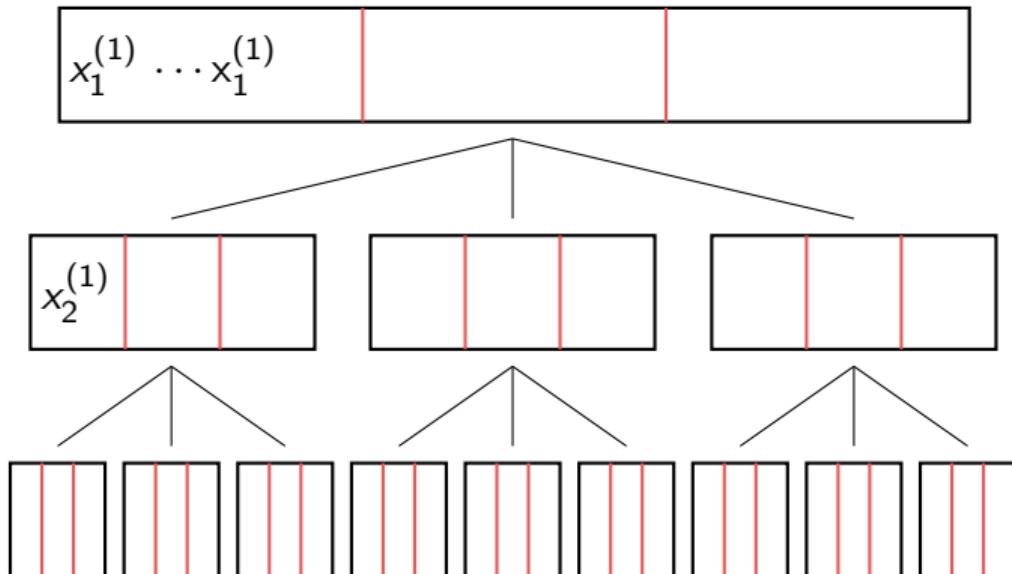
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Tree-Swap

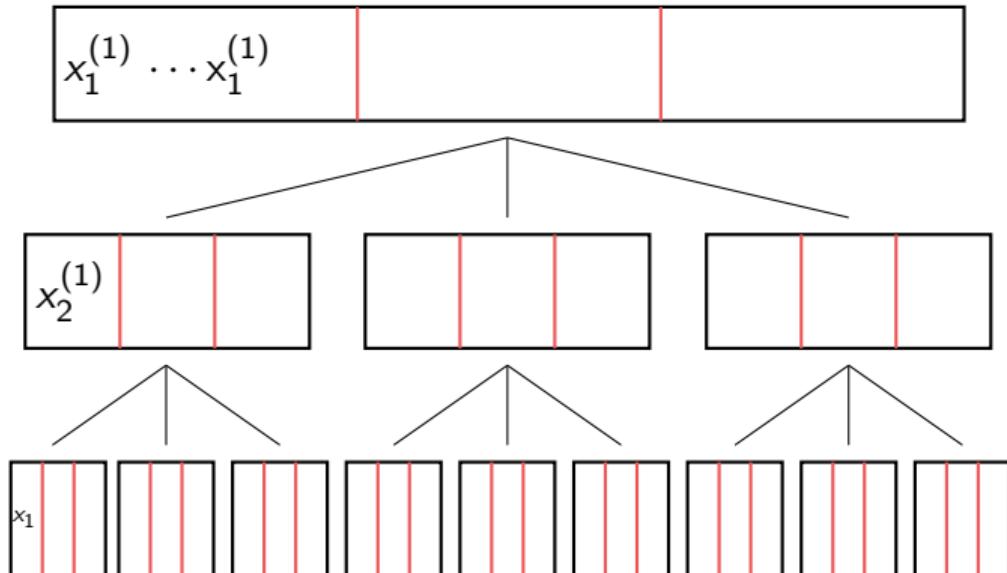
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Tree-Swap

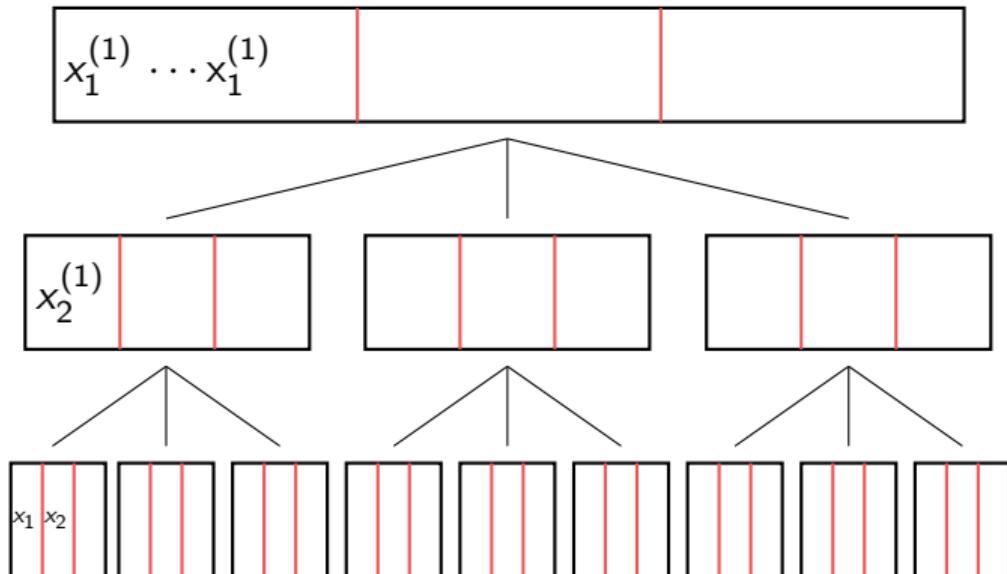
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "Active time"

# Tree-Swap

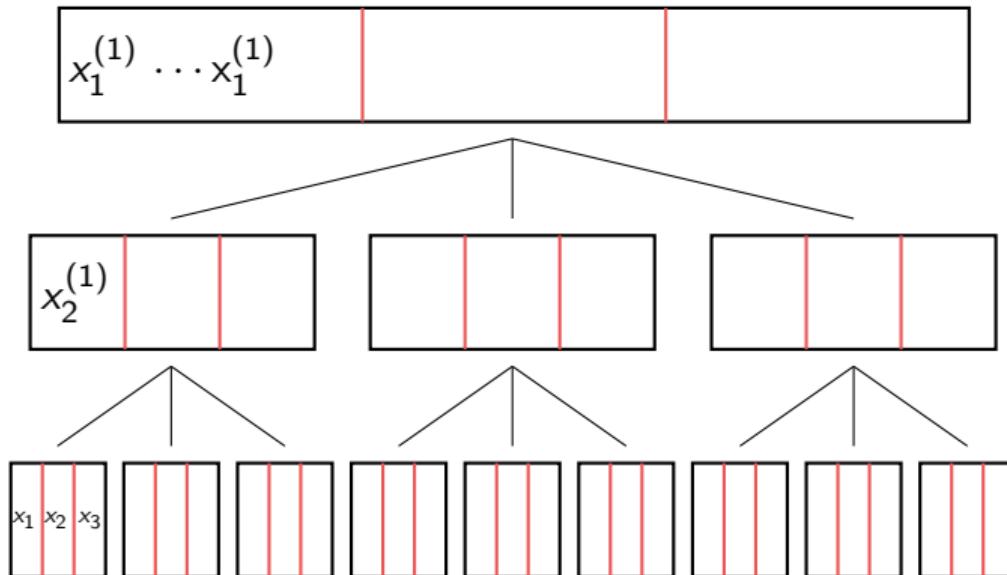
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in its "*Active time*"

# Tree-Swap

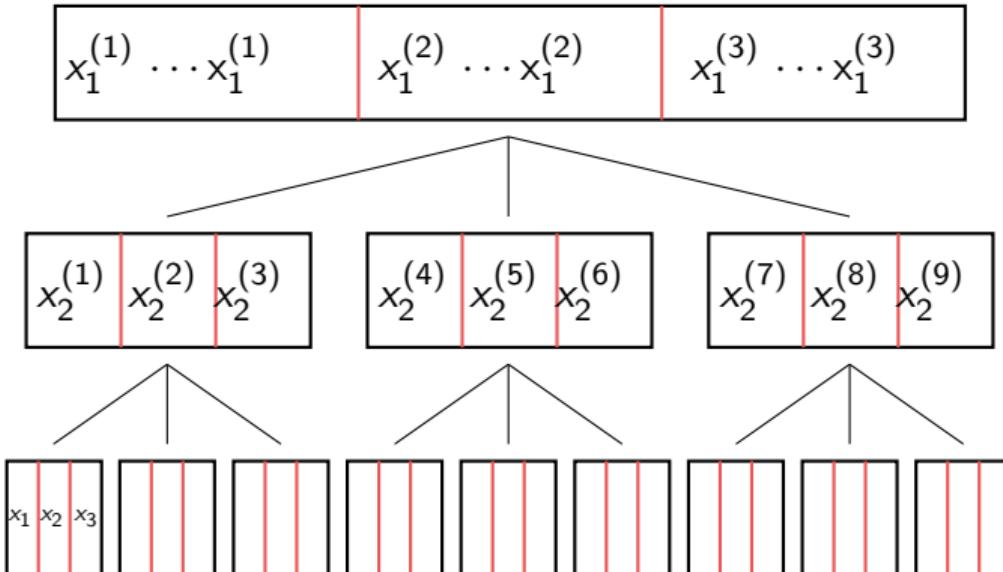
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in its "*Active time*"

# Tree-Swap

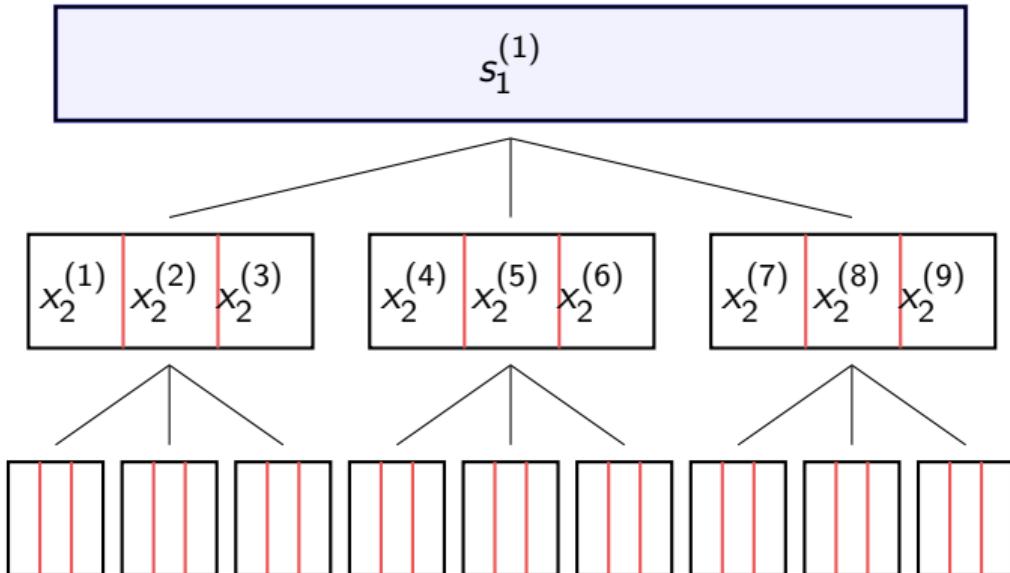
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

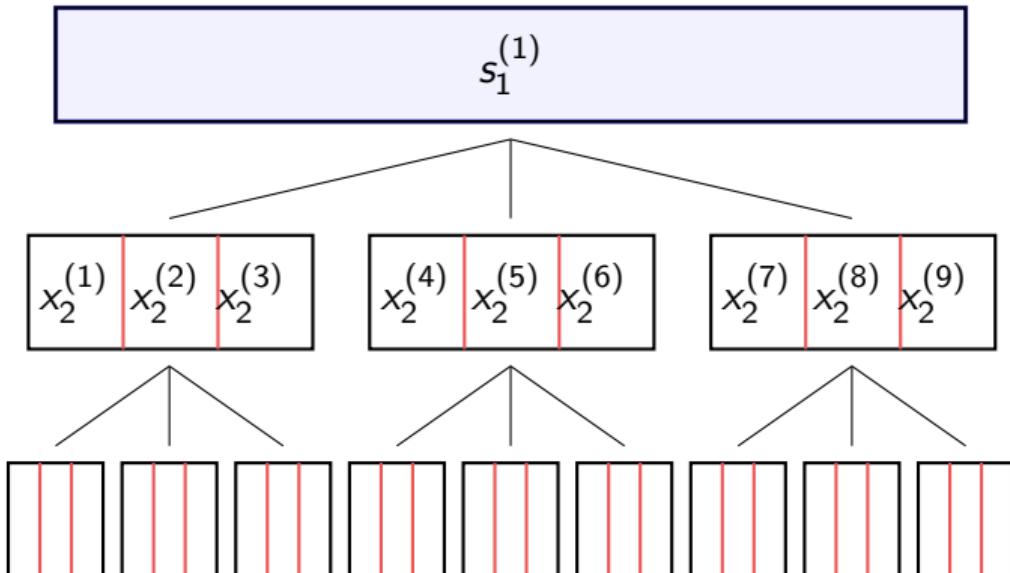
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

$$T = M^d$$

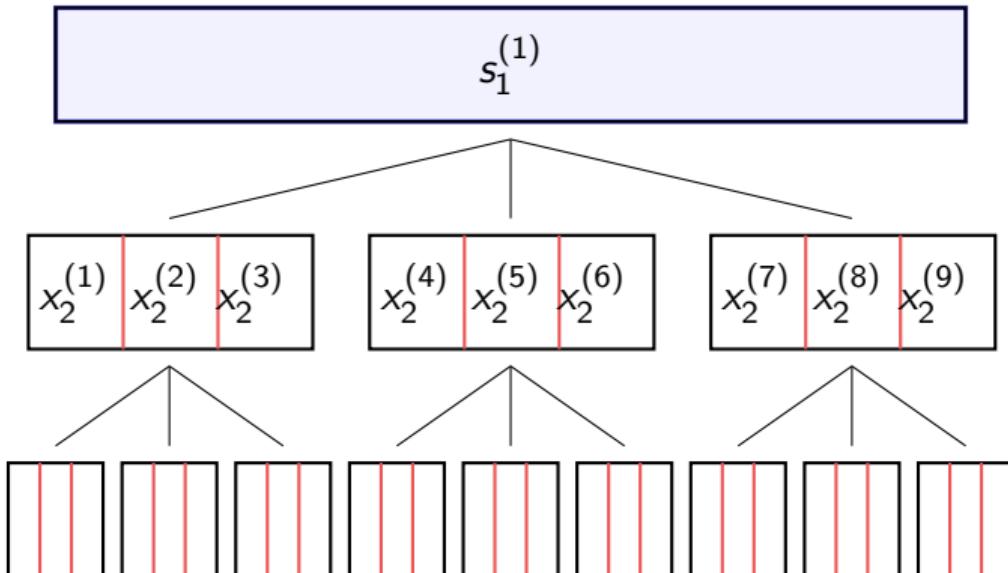


- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

---

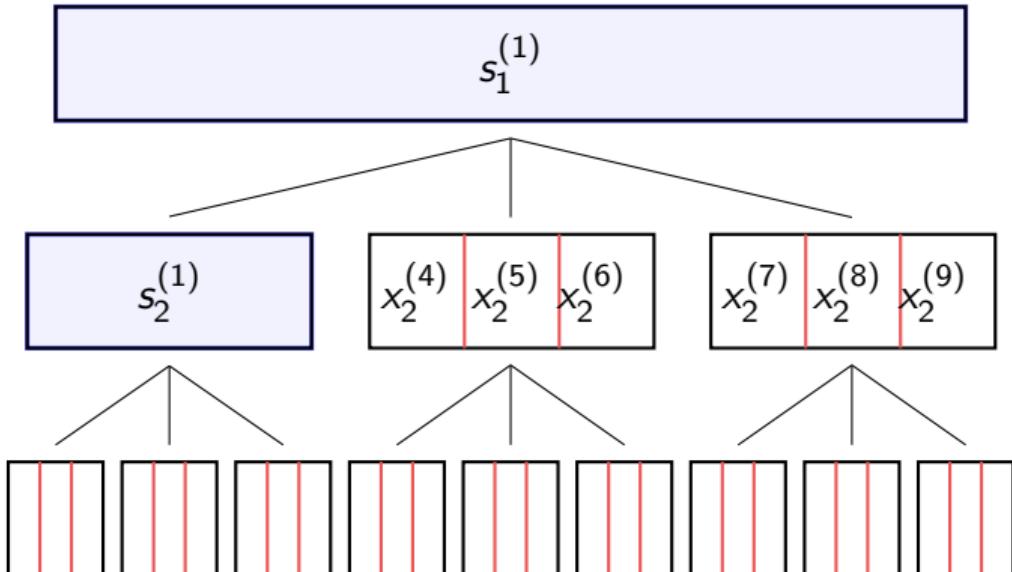
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

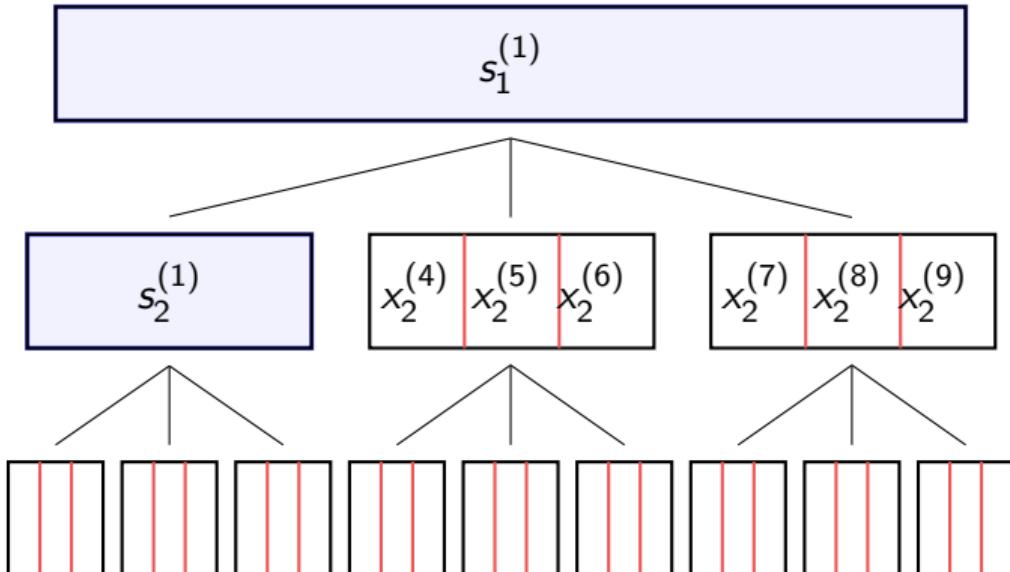
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

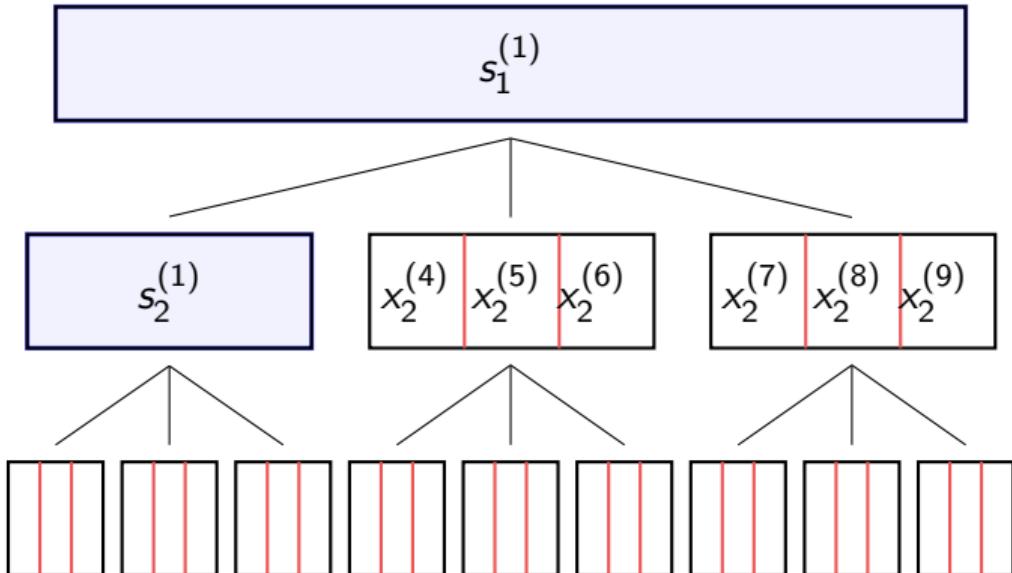
$$T = M^d$$



- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

$$T = M^d$$

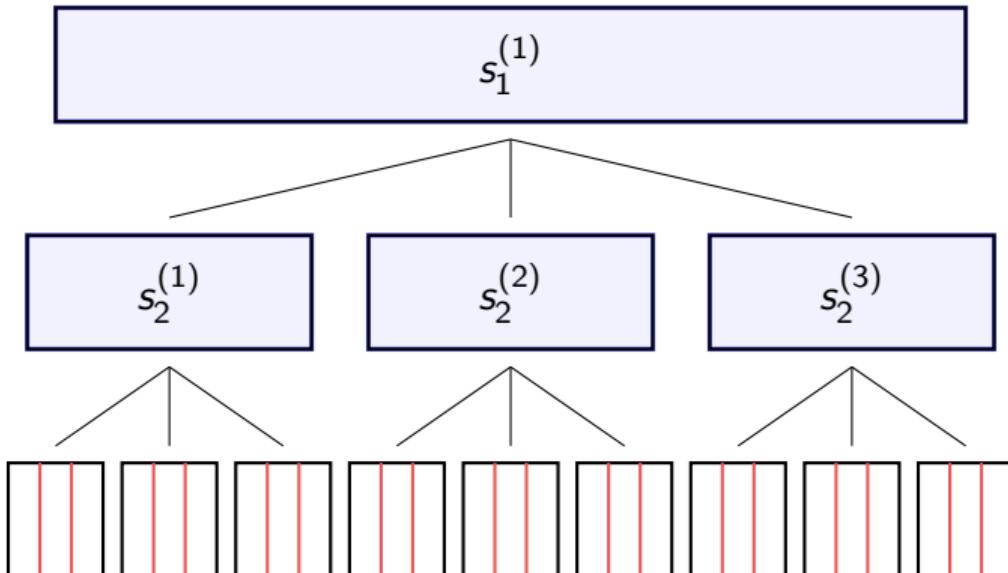


- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

---

$$T = M^d$$

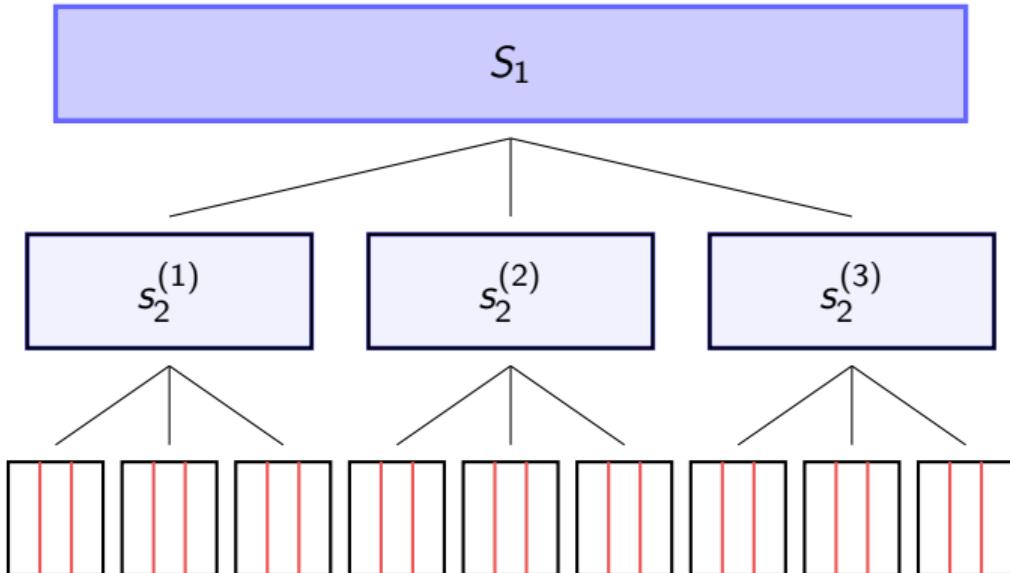


- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

---

$$T = M^d$$

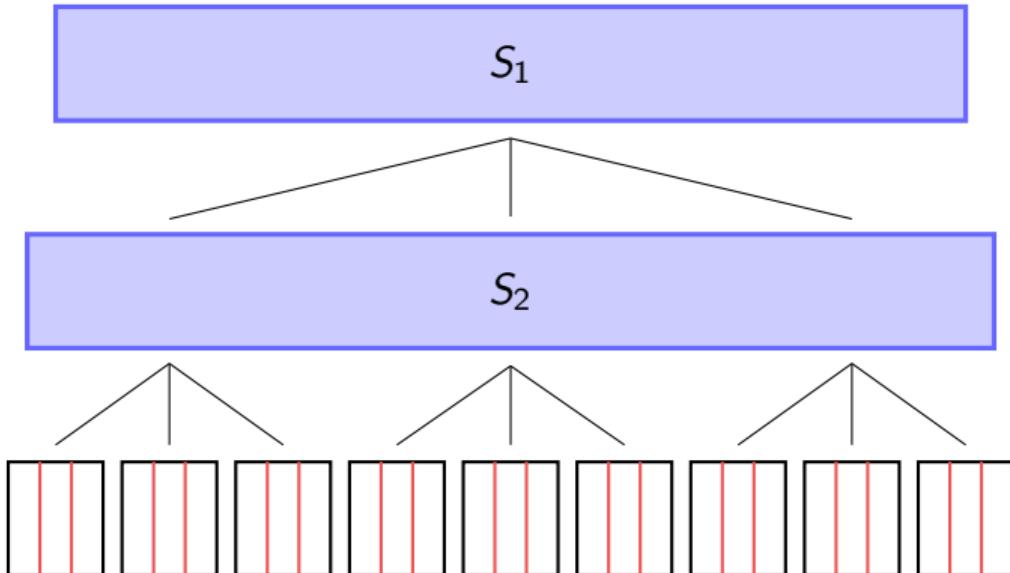


- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

---

$$T = M^d$$

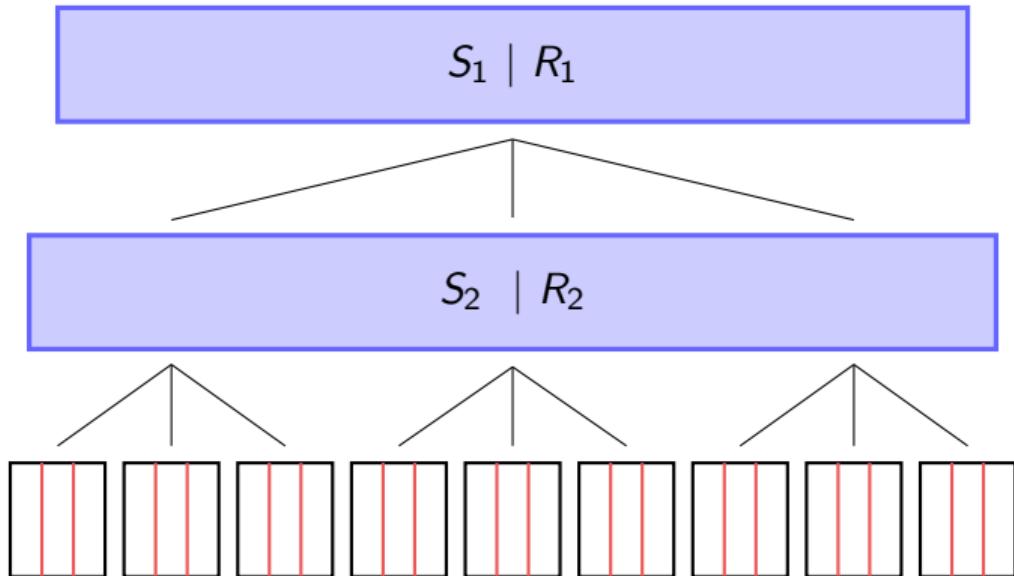


- At time  $t$ , the output is the average of plays on the path to leaf No. $t$ .
- Each MWU algorithm updates according to average reward in it's "*Active time*"

# Proof-Sketch

---

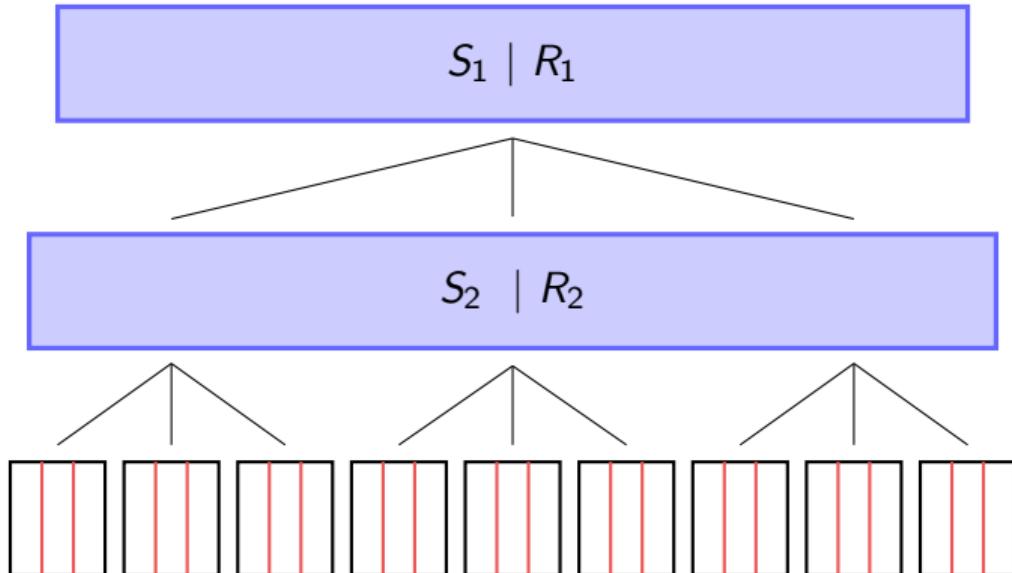
$$T = M^d$$



# Proof-Sketch

---

$$T = M^d$$

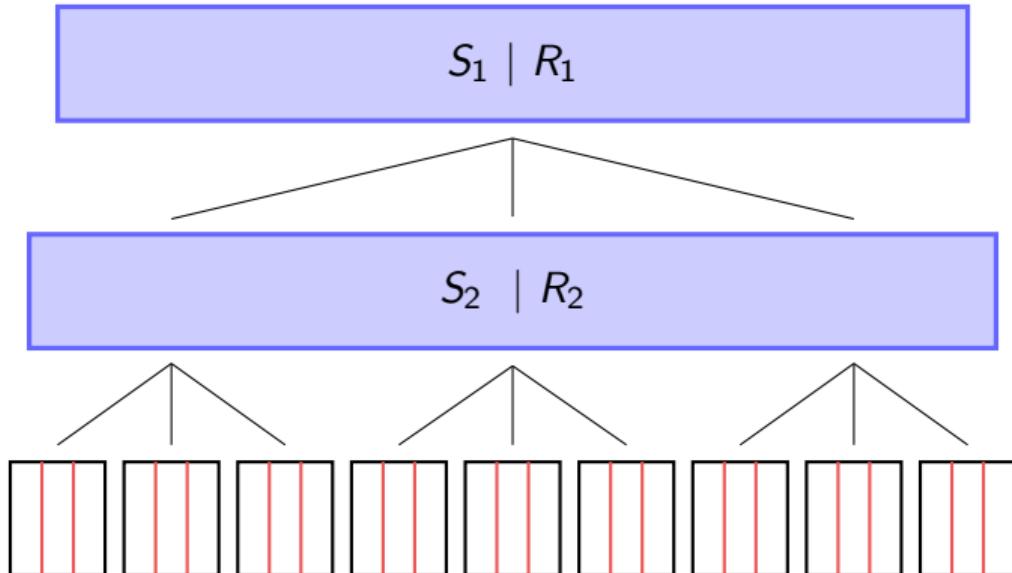


- By the *ER* assumption:  
 $S_i - R_i \leq \varepsilon$

# Proof-Sketch

---

$$T = M^d$$

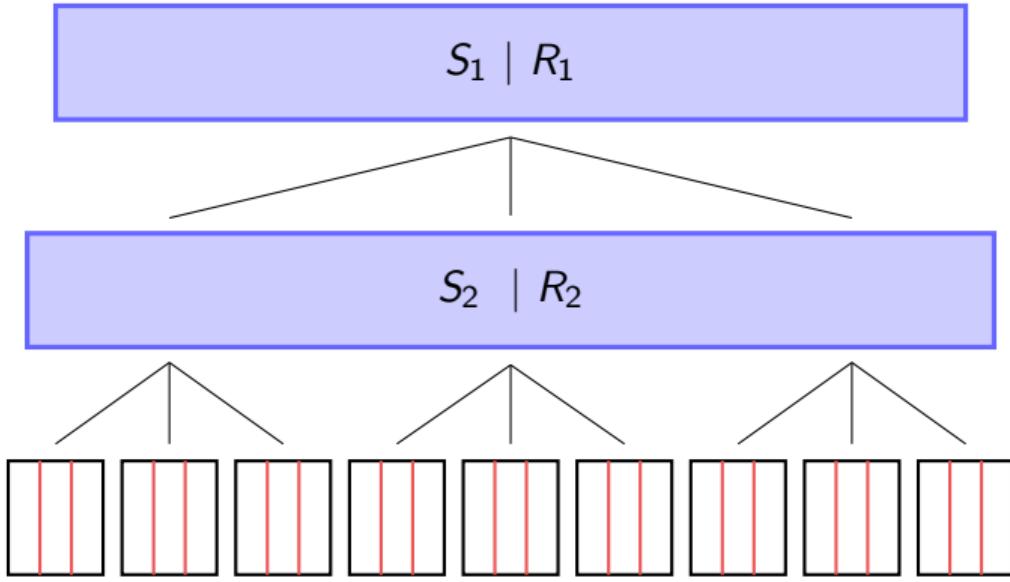


- By the *ER* assumption:  
 $S_i - R_i \leq \varepsilon$
- $\text{SwapRegret}_i \leq S_{i+1} - R_i$

# Proof-Sketch

---

$$T = M^d$$



- By the *ER* assumption:  
 $S_i - R_i \leq \varepsilon$
- $\text{SwapRegret}_i \leq S_{i+1} - R_i$
- Thus the total swap-regret of tree-swap is bounded by:

$$\frac{1}{d} \sum_{i=0}^{d-1} (S_{i+1} - R_i)$$

- By the *ER* assumption:  $S_i - R_i \leq \varepsilon$
- $\text{SwapRegret}_i \leq S_{i+1} - R_i$
- Thus the total swap-regret of tree-swap is bounded by:

$$\frac{1}{d} \sum_{i=0}^{d-1} (S_{i+1} - R_i) = \frac{1}{d} \sum_{i=0}^{d-1} (S_i - R_i) + \frac{S_d - S_0}{d}$$

- By the *ER* assumption:  $S_i - R_i \leq \varepsilon$
- $\text{SwapRegret}_i \leq S_{i+1} - R_i$
- Thus the total swap-regret of tree-swap is bounded by:

$$\frac{1}{d} \sum_{i=0}^{d-1} (S_{i+1} - R_i) = \frac{1}{d} \sum_{i=0}^{d-1} (S_i - R_i) + \frac{S_d - S_0}{d} \leq \varepsilon + \frac{1}{d}$$

# Tree-swap (Pseudo code)

**Require:** Action set  $\mathcal{X}$ , utility class  $\mathcal{F}$ , no-external regret algorithm  $Alg$ , time horizon  $T$ , parameters  $M, d$  with  $T \leq M^d$ .

- 1: For each sequence  $\sigma \in \bigcup_{h=1}^d \{0, 1, \dots, M-1\}^{h-1}$ , initialize an instance of  $Alg$  with time horizon  $M$ , denoted  $Alg_\sigma$ .
- 2: **for**  $1 \leq t \leq T$  **do**
- 3:   Let  $\sigma = (\sigma_1, \dots, \sigma_d)$  denote the base- $M$  representation of  $t-1$ .
- 4:   **for**  $1 \leq h \leq d$  **do**
- 5:     **if**  $\sigma_{h+1} = \dots = \sigma_d = 0$  or  $h = d$  **then**
- 6:       If  $\sigma_h > 0$ , call  $Alg_{\sigma_{1:h-1}}.\text{update}\left(\frac{1}{M^{d-h}} \cdot \sum_{s=t-M^{d-h}}^{t-1} f^{(s)}\right)$ .
- 7:        $Alg_{\sigma_{1:h-1}}.\text{curAction} \leftarrow Alg_{\sigma_{1:h-1}}.\text{act}()$ .
- 8:     **end if**
- 9:   **end for**
- 10:   Output the uniform mixture  $x^{(t)} := \frac{1}{d} \sum_{h=1}^d Alg_{\sigma_{1:h-1}}.\text{curAction}$ , and observe  $f^{(t)}$ .
- 11: **end for**

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR		

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime		

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	$\tilde{O}(N)$

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	$\tilde{O}(N)$
Finite Littlestone dim. upper bound		

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	$\tilde{O}(N)$
Finite Littlestone dim. upper bound		

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	$\tilde{O}(N)$
Finite Littlestone dim. upper bound	?	$(L_{dim}(\mathcal{X})/\varepsilon^2)^{\Omega(1/\varepsilon)}$

# Main results

---

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	$\tilde{O}(N)$
Finite Littlestone dim. upper bound		$(L_{dim}(\mathcal{X})/\varepsilon^2)^{\Omega(1/\varepsilon)}$

- Note that even the existence of approximate correlated equilibria in games of finite Littlestone dimension was previously unknown.

# Main results

	Best previously known	Tree-swap's result
Upper bound on $T$ for $\varepsilon$ SR	$\tilde{\Omega}(N/\varepsilon^2)$	$(\log(N)/\varepsilon^2)^{\Omega(1/\varepsilon)}$
Per-iteration runtime	$\Omega(N^3)$	$\tilde{O}(N)$
Finite Littlestone dim. upper bound		$(L_{dim}(\mathcal{X})/\varepsilon^2)^{\Omega(1/\varepsilon)}$

- Note that even the existence of approximate correlated equilibria in games of finite Littlestone dimension was previously unknown.
- Additionally, the paper proved a lower bound against an oblivious adversary:

$$\text{Let: } T \leq O(1) \cdot \min \left\{ \exp(O(\varepsilon^{-1/6})), \frac{N}{\log^{12}(N) \cdot \varepsilon^2} \right\}$$

Then, there exists an oblivious adversary, such that any learning algorithm run over  $T$  steps will incur swap regret at least  $\varepsilon$

# Critical Points

---

# Critical Points

---

- No known practical use for correlated equilibria in games 😞

# Critical Points

---

- No known practical use for correlated equilibria in games 😞
- Upper bound is pretty bad... exponential dependence on  $\frac{1}{\varepsilon}$  !!

# Future Research

---



# References

---



Dagan, Daskalakis, Fishelson, Golowich

STOC '24

► *From External to Swap Regret 2.0*



Blum, Mansour

Journal of Machine Learning Research 8 (2007)

► *From External to Internal Regret*



Yishay Mansour

Tel-Aviv University

► *Algorithmic Game Theory Course*



Prof. Dr. Thomas Kesselheim

University of Bonn

► *Algorithmic Game Theory Course*



Maxwell Fishelson

► *Tree Swap Presentation*

# The End

