

Algorithmic game theory

Martin Balko

8th lecture

November 25th 2025



Applications of regret minimization

Concluding the story of NE

Concluding the story of NE

- We learned that Nash equilibria (NE) always exist.

Concluding the story of NE

- We learned that Nash equilibria (NE) always exist. However, there seem to be no polynomial-time algorithm for computing NE.

Concluding the story of NE

- We learned that Nash equilibria (NE) always exist. However, there seem to be no polynomial-time algorithm for computing NE.
- Therefore we came up with “relaxations” of NE.

Concluding the story of NE

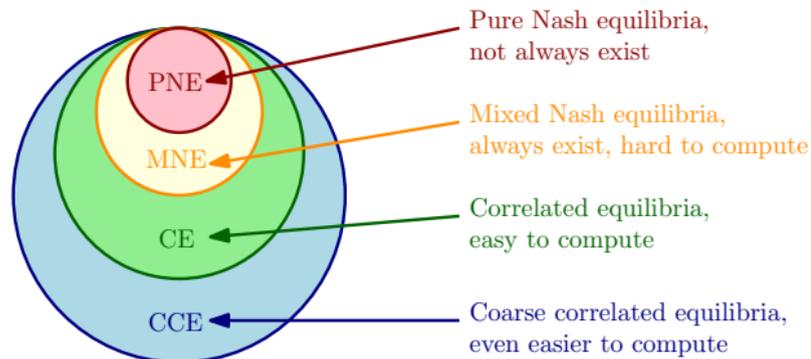
- We learned that Nash equilibria (NE) always exist. However, there seem to be no polynomial-time algorithm for computing NE.
- Therefore we came up with “relaxations” of NE. Correlated equilibria (CE) look particularly interesting as they are natural and we can compute them in polynomial time using linear programming.

Concluding the story of NE

- We learned that **Nash equilibria (NE)** always exist. However, there seem to be **no polynomial-time algorithm for computing NE**.
- Therefore we came up with “relaxations” of NE. **Correlated equilibria (CE)** look particularly interesting as they are natural and we can compute them in polynomial time using linear programming.
- Using **external regret minimization**, we can apply **No-regret dynamics** to converge to more general **coarse correlated equilibria (CCE)**.

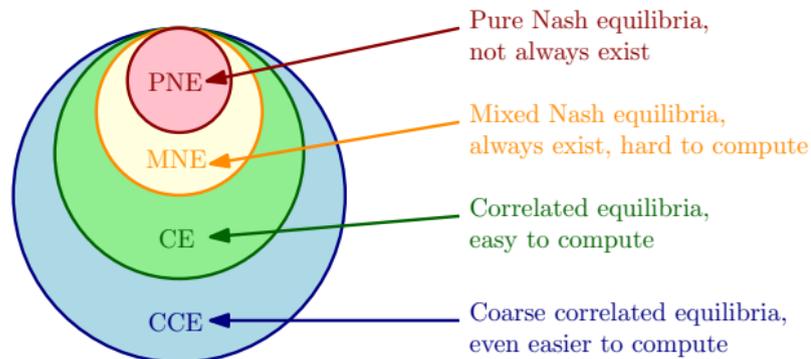
Concluding the story of NE

- We learned that **Nash equilibria (NE)** always exist. However, there seem to be **no polynomial-time algorithm for computing NE**.
- Therefore we came up with “relaxations” of NE. **Correlated equilibria (CE)** look particularly interesting as they are natural and we can compute them in polynomial time using linear programming.
- Using **external regret minimization**, we can apply **No-regret dynamics** to converge to more general **coarse correlated equilibria (CCE)**.



Concluding the story of NE

- We learned that **Nash equilibria (NE)** always exist. However, there seem to be **no polynomial-time algorithm for computing NE**.
- Therefore we came up with “relaxations” of NE. **Correlated equilibria (CE)** look particularly interesting as they are natural and we can compute them in polynomial time using linear programming.
- Using **external regret minimization**, we can apply **No-regret dynamics** to converge to more general **coarse correlated equilibria (CCE)**.



- Today, we show that the **No-swap-regret dynamics** converges to CE.

Our notation

Our notation

- At each step $t = 1, \dots, T$:

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X ,

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$,

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
- The agent A receives loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ at step t .

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
- The agent A receives loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ at step t . The cumulative loss of A is $L_A^T = \sum_{t=1}^T \ell_A^t$.

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
- The agent A receives loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ at step t . The cumulative loss of A is $L_A^T = \sum_{t=1}^T \ell_A^t$. The cumulative loss of i is $L_i^T = \sum_{t=1}^T \ell_i^t$.

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
- The agent A receives loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ at step t . The cumulative loss of A is $L_A^T = \sum_{t=1}^T \ell_A^t$. The cumulative loss of i is $L_i^T = \sum_{t=1}^T \ell_i^t$.

Weather					Loss
Algorithm					1
Umbrella					1
Sunscreen					3

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
- The agent A receives loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ at step t . The cumulative loss of A is $L_A^T = \sum_{t=1}^T \ell_A^t$. The cumulative loss of i is $L_i^T = \sum_{t=1}^T \ell_i^t$.

Weather					Loss
Algorithm					1
Umbrella					1
Sunscreen					3

- We modify a sequence $(p^t)_{t=1}^T$ with $F: X \rightarrow X$ by replacing it with a sequence $(f^t)_{t=1}^T$, where $f^t = (f_1^t, \dots, f_N^t)$ and $f_i^t = \sum_{j: F(j)=i} p_j^t$.

Our notation

- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
- The agent A receives loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$ at step t . The cumulative loss of A is $L_A^T = \sum_{t=1}^T \ell_A^t$. The cumulative loss of i is $L_i^T = \sum_{t=1}^T \ell_i^t$.

Weather					Loss
Algorithm					1
Umbrella					1
Sunscreen					3

- We modify a sequence $(p^t)_{t=1}^T$ with $F: X \rightarrow X$ by replacing it with a sequence $(f^t)_{t=1}^T$, where $f^t = (f_1^t, \dots, f_N^t)$ and $f_i^t = \sum_{j: F(j)=i} p_j^t$.
- The cumulative loss of A modified by F is $L_{A,F}^T = \sum_{t=1}^T \sum_{i=1}^N f_i^t \ell_i^t$.

All the regrets we have

All the regrets we have

- We also recall all variants of regret that we discussed.

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i: i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the external regret:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i : i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the **external regret**:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who always play the same action.”

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i : i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the **external regret**:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who always play the same action.”
- For $\mathcal{F}^{\text{in}} = \{F_{i,j} : (i, j) \in X \times X, i \neq j\}$ where $F_{i,j}$ is defined by $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for each $i' \neq i$, we get the **internal regret**:

$$R_{A, \mathcal{F}^{\text{in}}}^T = \max_{F \in \mathcal{F}^{\text{in}}} \{L_A^T - L_{A, F}^T\}.$$

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i : i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the **external regret**:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who always play the same action.”
- For $\mathcal{F}^{\text{in}} = \{F_{i,j} : (i, j) \in X \times X, i \neq j\}$ where $F_{i,j}$ is defined by $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for each $i' \neq i$, we get the **internal regret**:

$$R_{A, \mathcal{F}^{\text{in}}}^T = \max_{F \in \mathcal{F}^{\text{in}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who play j whenever A plays i .”

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i: i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the **external regret**:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who always play the same action.”
- For $\mathcal{F}^{\text{in}} = \{F_{i,j}: (i, j) \in X \times X, i \neq j\}$ where $F_{i,j}$ is defined by $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for each $i' \neq i$, we get the **internal regret**:

$$R_{A, \mathcal{F}^{\text{in}}}^T = \max_{F \in \mathcal{F}^{\text{in}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who play j whenever A plays i .”
- For the set \mathcal{F}^{sw} of all modification rules, we get the **swap regret**:

$$R_{A, \mathcal{F}^{\text{sw}}}^T = \max_{F \in \mathcal{F}^{\text{sw}}} \{L_A^T - L_{A, F}^T\}.$$

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i : i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the **external regret**:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who always play the same action.”
- For $\mathcal{F}^{\text{in}} = \{F_{i,j} : (i, j) \in X \times X, i \neq j\}$ where $F_{i,j}$ is defined by $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for each $i' \neq i$, we get the **internal regret**:

$$R_{A, \mathcal{F}^{\text{in}}}^T = \max_{F \in \mathcal{F}^{\text{in}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who play j whenever A plays i .”
- For the set \mathcal{F}^{sw} of all modification rules, we get the **swap regret**:

$$R_{A, \mathcal{F}^{\text{sw}}}^T = \max_{F \in \mathcal{F}^{\text{sw}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. all his modifications.”

All the regrets we have

- We also recall all variants of regret that we discussed.
- For a set $\mathcal{F}^{\text{ex}} = \{F_i : i \in X\}$ of rules where F_i always outputs action i , we obtain exactly the **external regret**:

$$R_{A, \mathcal{F}^{\text{ex}}}^T = \max_{F \in \mathcal{F}^{\text{ex}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who always play the same action.”
- For $\mathcal{F}^{\text{in}} = \{F_{i,j} : (i, j) \in X \times X, i \neq j\}$ where $F_{i,j}$ is defined by $F_{i,j}(i) = j$ and $F_{i,j}(i') = i'$ for each $i' \neq i$, we get the **internal regret**:

$$R_{A, \mathcal{F}^{\text{in}}}^T = \max_{F \in \mathcal{F}^{\text{in}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. agents who play j whenever A plays i .”
- For the set \mathcal{F}^{sw} of all modification rules, we get the **swap regret**:

$$R_{A, \mathcal{F}^{\text{sw}}}^T = \max_{F \in \mathcal{F}^{\text{sw}}} \{L_A^T - L_{A, F}^T\}.$$

- “Agent A vs. all his modifications.” Note: $R_{A, \mathcal{F}^{\text{ex}}}^T, R_{A, \mathcal{F}^{\text{in}}}^T \leq R_{A, \mathcal{F}^{\text{sw}}}^T$.

Reduction from external regret to swap regret

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small external regret.

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**?

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**? Yes, using a clever **black-box reduction**!

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**? Yes, using a clever **black-box reduction**!
- An **R -external regret algorithm A** guarantees that for every sequence $(\ell^t)_{t=1}^T$ of loss vectors and for every action $j \in X$, we have

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**? Yes, using a clever **black-box reduction**!
- An **R -external regret algorithm A** guarantees that for every sequence $(\ell^t)_{t=1}^T$ of loss vectors and for every action $j \in X$, we have

$$L_A^T = \sum_{t=1}^T \ell_A^t \leq \sum_{t=1}^T \ell_j^t + R = L_j^T + R.$$

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**? Yes, using a clever **black-box reduction**!
- An **R -external regret algorithm A** guarantees that for every sequence $(\ell^t)_{t=1}^T$ of loss vectors and for every action $j \in X$, we have

$$L_A^T = \sum_{t=1}^T \ell_A^t \leq \sum_{t=1}^T \ell_j^t + R = L_j^T + R.$$

Theorem 2.55

For every R -external regret algorithm A , there exists an algorithm $M = M(A)$ such that, for every $F: X \rightarrow X$ and $T \in \mathbb{N}$, we have

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**? Yes, using a clever **black-box reduction**!
- An **R -external regret algorithm A** guarantees that for every sequence $(\ell^t)_{t=1}^T$ of loss vectors and for every action $j \in X$, we have

$$L_A^T = \sum_{t=1}^T \ell_A^t \leq \sum_{t=1}^T \ell_j^t + R = L_j^T + R.$$

Theorem 2.55

For every R -external regret algorithm A , there exists an algorithm $M = M(A)$ such that, for every $F: X \rightarrow X$ and $T \in \mathbb{N}$, we have

$$L_M^T \leq L_{M,F}^T + NR.$$

Reduction from external regret to swap regret

- We have the PW algorithm with arbitrarily small **external regret**.
- Can we design such an algorithm also for **swap regret**? Yes, using a clever **black-box reduction**!
- An **R -external regret algorithm A** guarantees that for every sequence $(\ell^t)_{t=1}^T$ of loss vectors and for every action $j \in X$, we have

$$L_A^T = \sum_{t=1}^T \ell_A^t \leq \sum_{t=1}^T \ell_j^t + R = L_j^T + R.$$

Theorem 2.55

For every R -external regret algorithm A , there exists an algorithm $M = M(A)$ such that, for every $F: X \rightarrow X$ and $T \in \mathbb{N}$, we have

$$L_M^T \leq L_{M,F}^T + NR.$$

That is, the **swap regret of M** is at most NR .

Proof of the reduction I

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A .

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$.

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $Q_{i,j}^t = q_{i,j}^t$.

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $Q_{i,j}^t = q_{i,j}^t$.
- It can be shown that p^t exists and is efficiently computable.

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $Q_{i,j}^t = q_{i,j}^t$.
- It can be shown that p^t exists and is efficiently computable.
 - It is a “stationary distribution of the transition matrix of a Markov chain”.

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $Q_{i,j}^t = q_{i,j}^t$.
- It can be shown that p^t exists and is efficiently computable.
 - It is a “stationary distribution of the transition matrix of a Markov chain”.
- This choice of p^t guarantees that we can consider action selection in two equivalent ways.

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $Q_{i,j}^t = q_{i,j}^t$.
- It can be shown that p^t exists and is efficiently computable.
 - It is a “stationary distribution of the transition matrix of a Markov chain”.
- This choice of p^t guarantees that we can consider action selection in two equivalent ways. An action $j \in X$ is either selected with a probability p_j^t

Proof of the reduction I

- Assume that A_1, \dots, A_N are copies of the algorithm A . In every time step t , each A_i outputs a probability distribution $q_i^t = (q_{i,1}^t, \dots, q_{i,N}^t)$, where $q_{i,j}^t$ is the fraction A_i assigns to an action $j \in X$.
- We construct the master algorithm M by combining these copies of A .
- We construct a single probability distribution $p^t = (p_1^t, \dots, p_N^t)$ by letting $p_j^t = \sum_{i=1}^N p_i^t q_{i,j}^t$ for every $j \in X$. That is, $(p^t)^\top = (p^t)^\top Q^t$, where Q^t is an $N \times N$ matrix with $Q_{i,j}^t = q_{i,j}^t$.
- It can be shown that p^t exists and is efficiently computable.
 - It is a “stationary distribution of the transition matrix of a Markov chain”.
- This choice of p^t guarantees that we can consider action selection in two equivalent ways. An action $j \in X$ is either selected with a probability p_j^t or we first select an algorithm A_i with probability p_i^t and then use the algorithm A_i to select j with probability $q_{i,j}^t$.

Proof of the reduction II

Proof of the reduction II

- We show that the total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$,

Proof of the reduction II

- We show that the total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$, the actual loss of M .

Proof of the reduction II

- We show that the total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$, the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i .

Proof of the reduction II

- We show that the total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$, the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.

Proof of the reduction II

- We show that the total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$, the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.
- Since A_i is an R -external regret algorithm, we have, for each $j \in X$,

Proof of the reduction II

- We show that the **total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$** , the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.
- Since A_i is an **R -external regret algorithm**, we have, for each $j \in X$,

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t)$$

Proof of the reduction II

- We show that the **total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$** , the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.
- Since A_i is an **R -external regret algorithm**, we have, for each $j \in X$,

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

Proof of the reduction II

- We show that the **total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$** , the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.
- Since A_i is an **R -external regret algorithm**, we have, for each $j \in X$,

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

- Summing the losses of all algorithms A_i over $i \in X$, we get the total loss $\sum_{i=1}^N p_i^t (q_i^t \cdot \ell^t) = (p^t)^\top Q^t \ell^t$ of all algorithms A_i at time step t .

Proof of the reduction II

- We show that the **total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$** , the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.
- Since A_i is an **R -external regret algorithm**, we have, for each $j \in X$,

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

- Summing the losses of all algorithms A_i over $i \in X$, we get the total loss $\sum_{i=1}^N p_i^t (q_i^t \cdot \ell^t) = (p^t)^\top Q^t \ell^t$ of all algorithms A_i at time step t .
- By the choice of p^t , we have $(p^t)^\top = (p^t)^\top Q^t$.

Proof of the reduction II

- We show that the **total loss of all algorithms A_i at step t equals $p^t \cdot \ell^t$** , the actual loss of M .
- After receiving a loss vector ℓ^t , we give, for each $i \in X$, a loss vector $p_i^t \ell^t$ to A_i . Then, A_i experiences loss $(p_i^t \ell^t) \cdot q_i^t = p_i^t (q_i^t \cdot \ell^t)$.
- Since A_i is an **R -external regret algorithm**, we have, for each $j \in X$,

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

- Summing the losses of all algorithms A_i over $i \in X$, we get the total loss $\sum_{i=1}^N p_i^t (q_i^t \cdot \ell^t) = (p^t)^\top Q^t \ell^t$ of all algorithms A_i at time step t .
- By the choice of p^t , we have $(p^t)^\top = (p^t)^\top Q^t$. Thus we get what we wanted.

Proof of the reduction III

Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

- The right-hand side of is true for every action $j \in X$,

Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

- The right-hand side of is true for every action $j \in X$, so we obtain, for every function $F: X \rightarrow X$,

$$L_M^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t \ell_{F(i)}^t + NR$$

Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

- The right-hand side of is true for every action $j \in X$, so we obtain, for every function $F: X \rightarrow X$,

$$L_M^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t \ell_{F(i)}^t + NR = L_{M,F}^T + NR.$$

Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

- The right-hand side of is true for every action $j \in X$, so we obtain, for every function $F: X \rightarrow X$,

$$L_M^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t \ell_{F(i)}^t + NR = L_{M,F}^T + NR.$$



Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

- The right-hand side of is true for every action $j \in X$, so we obtain, for every function $F: X \rightarrow X$,

$$L_M^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t \ell_{F(i)}^t + NR = L_{M,F}^T + NR.$$

□

- Using the PW algorithm as A , we get an algorithm with swap regret at most $O(N\sqrt{T \log N})$.

Proof of the reduction III

- Thus, summing

$$\sum_{t=1}^T p_i^t (q_i^t \cdot \ell^t) \leq \sum_{t=1}^T p_i^t \ell_j^t + R.$$

over all actions $i \in X$, the left-hand side equals L_M^T .

- The right-hand side of is true for every action $j \in X$, so we obtain, for every function $F: X \rightarrow X$,

$$L_M^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t \ell_{F(i)}^t + NR = L_{M,F}^T + NR.$$

□

- Using the PW algorithm as A , we get an algorithm with swap regret at most $O(N\sqrt{T \log N})$.
- That is, its average swap regret goes to 0 with $T \rightarrow \infty$.

No-swap-regret dynamics

No-swap-regret dynamics

- Using **swap regret** instead of external regret, we get:

No-swap-regret dynamics

- Using **swap regret** instead of external regret, we get:

Algorithm 0.3: NO-SWAP-REGRET DYNAMICS(G, T, ε)

Input : A normal-form game $G = (P, A, C)$ of n players, $T \in \mathbb{N}$, and $\varepsilon > 0$.

Output : A prob. distribution p_i^t on A_i for each $i \in P$ and $t \in \{1, \dots, T\}$.

for every step $t = 1, \dots, T$

do $\left\{ \begin{array}{l} \text{Each player } i \in P \text{ independently chooses a mixed strategy } p_i^t \\ \text{using an algorithm with average } \text{swap regret} \text{ at most } \varepsilon, \text{ with} \\ \text{actions corresponding to pure strategies.} \\ \text{Each player } i \in P \text{ receives a loss vector } \ell_i^t = (\ell_i^t(a_j))_{a_j \in A_j}, \text{ where} \\ \ell_i^t(a_j) \leftarrow \mathbb{E}_{a_{-i}^t \sim p_{-i}^t} [C_i(a_j; a_{-i}^t)] \text{ for the product distribution} \\ p_{-i}^t = \prod_{j \neq i} p_j^t. \end{array} \right.$

Output $\{p^t : t \in \{1, \dots, T\}\}$.

No-swap-regret dynamics

- Using **swap regret** instead of external regret, we get:

Algorithm 0.4: NO-SWAP-REGRET DYNAMICS(G, T, ε)

Input : A normal-form game $G = (P, A, C)$ of n players, $T \in \mathbb{N}$, and $\varepsilon > 0$.

Output : A prob. distribution p_i^t on A_i for each $i \in P$ and $t \in \{1, \dots, T\}$.

for every step $t = 1, \dots, T$

do $\left\{ \begin{array}{l} \text{Each player } i \in P \text{ independently chooses a mixed strategy } p_i^t \\ \text{using an algorithm with average } \text{swap regret} \text{ at most } \varepsilon, \text{ with} \\ \text{actions corresponding to pure strategies.} \\ \text{Each player } i \in P \text{ receives a loss vector } \ell_i^t = (\ell_i^t(a_j))_{a_j \in A_j}, \text{ where} \\ \ell_i^t(a_j) \leftarrow \mathbb{E}_{a_{-i}^t \sim p_{-i}^t} [C_i(a_j; a_{-i}^t)] \text{ for the product distribution} \\ p_{-i}^t = \prod_{j \neq i} p_j^t. \end{array} \right.$

Output $\{p^t : t \in \{1, \dots, T\}\}$.

- No-swap-regret dynamics then converges to a correlated equilibrium.**

Converging to CE

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- Proof:

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)]$$

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)], \quad \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})]$$

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)], \quad \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})]$$

- The right-hand sides are time-averaged expected costs of i when playing according to the algorithm with small **swap** regret and when playing $F(a_i)$ instead of a_i .

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)], \quad \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})]$$

- The right-hand sides are time-averaged expected costs of i when playing according to the algorithm with small **swap** regret and when playing $F(a_i)$ instead of a_i . Since every player has **regret at most ε** , we obtain

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)], \quad \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})]$$

- The right-hand sides are time-averaged expected costs of i when playing according to the algorithm with small **swap** regret and when playing $F(a_i)$ instead of a_i . Since every player has **regret at most** ε , we obtain

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)] \leq \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})] + \varepsilon.$$

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)], \quad \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})]$$

- The right-hand sides are time-averaged expected costs of i when playing according to the algorithm with small **swap** regret and when playing $F(a_i)$ instead of a_i . Since every player has **regret at most ε** , we obtain

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)] \leq \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})] + \varepsilon.$$

- This verifies the ε -CE condition for $p = \frac{1}{T} \sum_{t=1}^T p_t$.

Converging to CE

Theorem 2.57

For every $G = (P, A, C)$, $\varepsilon > 0$, and $T = T(\varepsilon) \in \mathbb{N}$, if after T steps of the **No-swap-regret** dynamics, each player $i \in P$ has time-averaged expected regret at most ε , then p is ε -CE where $p^t = \prod_{i=1}^n p_i^t$ and $p = \frac{1}{T} \sum_{t=1}^T p^t$.

- **Proof:** We want to prove $\mathbb{E}_{a \sim p}[C_i(a)] \leq \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] + \varepsilon$.
- By the definition of p , we have, for every player $i \in P$ and $F: X \rightarrow X$,

$$\mathbb{E}_{a \sim p}[C_i(a)] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)], \quad \mathbb{E}_{a \sim p}[C_i(F(a_i); a_{-i})] = \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})]$$

- The right-hand sides are time-averaged expected costs of i when playing according to the algorithm with small **swap** regret and when playing $F(a_i)$ instead of a_i . Since every player has **regret at most ε** , we obtain

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(a)] \leq \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{a \sim p^t}[C_i(F(a_i); a_{-i})] + \varepsilon.$$

- This verifies the ε -CE condition for $p = \frac{1}{T} \sum_{t=1}^T p^t$. □

Games in extensive form

Games in extensive form

Games in extensive form

- In **normal-form games**, players act simultaneously resulting in a **static** description of a game.

Games in extensive form

- In **normal-form games**, players act simultaneously resulting in a **static** description of a game.
- Today, we describe a **different representation** of games which provides a **dynamic** description where players act sequentially.

Games in extensive form

- In **normal-form games**, players act simultaneously resulting in a **static** description of a game.
- Today, we describe a **different representation** of games which provides a **dynamic** description where players act sequentially.
- Instead of tables, we describe games using **trees**.



Zdroj: <https://cz.pinterest.com>

Games in extensive form

- In **normal-form games**, players act simultaneously resulting in a **static** description of a game.
- Today, we describe a **different representation** of games which provides a **dynamic** description where players act sequentially.
- Instead of tables, we describe games using **trees**.



Zdroj: <https://cz.pinterest.com>

- For some of these games, we **show how to compute NE**.

Example: normal-form of chess

Example: normal-form of chess



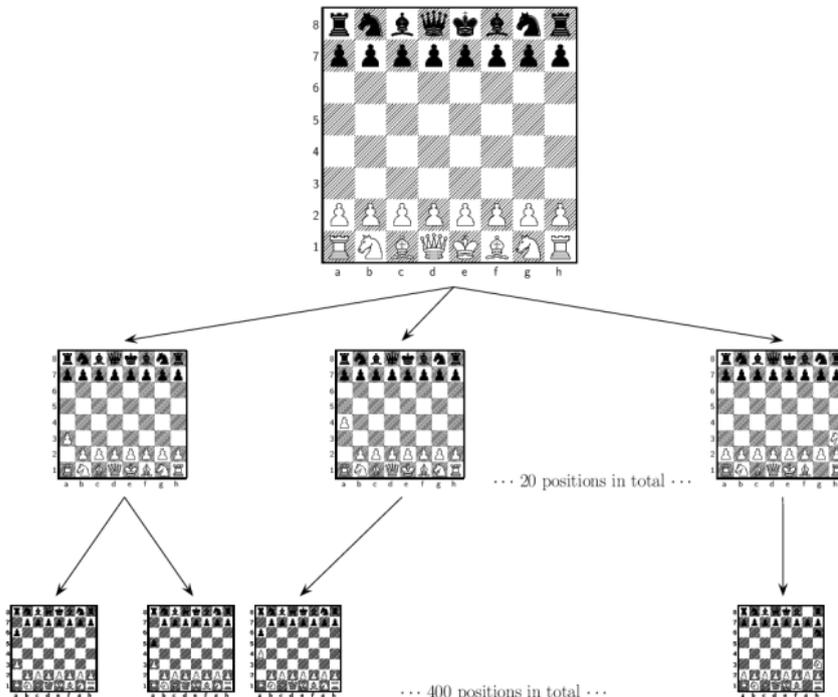
Source: <https://edition.cnn.com/>

- **Chess as a normal-form game:** Each action of player $i \in \{\text{black, white}\}$ is a list of all possible situations that can happen on the board together with the move player i would make in that situation. Then we can simulate the whole game of chess in one round.

Example: extensive form of chess

Example: extensive form of chess

- Root corresponds to the initial position of the chessboard. Each decision node represents a position on the chessboard and its outgoing edges correspond to possible moves in such a position.



Basic definitions

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).
 - For example, Chess.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).
 - For example, Chess.
- In **imperfect-information games** players have only partial knowledge.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).
 - For example, Chess.
- In **imperfect-information games** players have only partial knowledge.
 - For example, Poker.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).
 - For example, Chess.
- In **imperfect-information games** players have only partial knowledge.
 - For example, Poker.
 - We partition decision nodes into **information sets** where all nodes belong to the same player, and have the same moves.

Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).
 - For example, Chess.
- In **imperfect-information games** players have only partial knowledge.
 - For example, Poker.
 - We partition decision nodes into **information sets** where all nodes belong to the same player, and have the same moves.
 - For player i , let H_i be the set of information sets of i

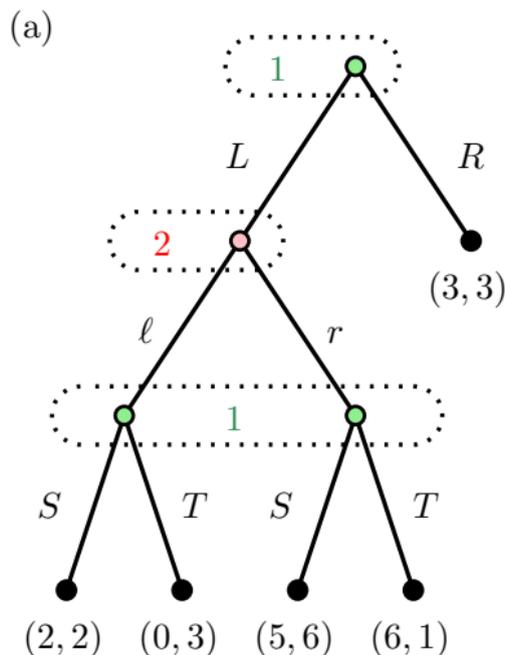
Basic definitions

- **Extensive game** consists of a directed tree where nodes represent game states. The tree encodes the full history of play.
- The game starts at the root of the tree and ends at a leaf, where each player receives a payoff. A tree edge corresponds to one player moving from one state to a different state of the game.
- Each node that is not a leaf is called a **decision node**.
- **Moves** a player can make in a given state are assigned to the outgoing edges of the corresponding decision node.
- In **perfect-information game** all players know the node they are in (that is, they know the history of the play that led them there).
 - For example, Chess.
- In **imperfect-information games** players have only partial knowledge.
 - For example, Poker.
 - We partition decision nodes into **information sets** where all nodes belong to the same player, and have the same moves.
 - For player i , let H_i be the set of information sets of i and, for $h \in H_i$, let C_h be the set of moves at h .

Example: imperfect-information game

Example: imperfect-information game

- An example of an imperfect-information game in extensive form (**part (a)**) and its normal-form (**part (b)**).



(b)

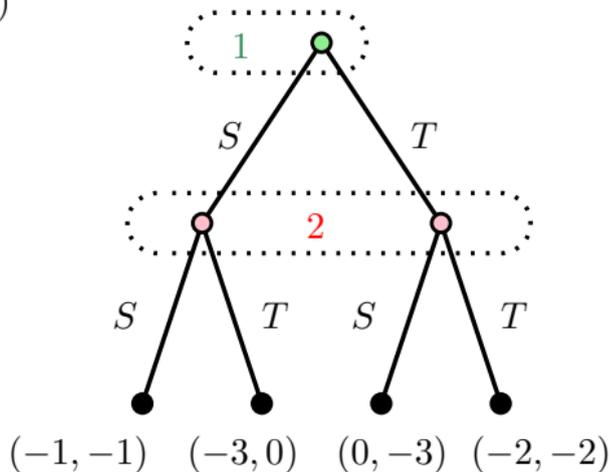
	(ℓ)	(r)
(L, S)	$(2, 2)$	$(5, 6)$
(L, T)	$(0, 3)$	$(6, 1)$
(R, S)	$(3, 3)$	$(3, 3)$
(R, T)	$(3, 3)$	$(3, 3)$

Example: Prisoner's dilemma

Example: Prisoner's dilemma

- Prisoner's dilemma in extensive form (part (a)) and its normal-form (part (b)).

(a)



(b)

	T	S
T	$(-2, -2)$	$(0, -3)$
S	$(-3, 0)$	$(-1, -1)$

Strategies in extensive games

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.
- **Mixed strategies** of G are the mixed strategies of G' .

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.
- **Mixed strategies** of G are the mixed strategies of G' .
- In the same way, we also define the set of **Nash equilibria** of G .

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.
- **Mixed strategies** of G are the mixed strategies of G' .
- In the same way, we also define the set of **Nash equilibria** of G .
- A **behavioral strategy** of player i is a probability distribution on C_h for each $h \in H_i$.

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.
- **Mixed strategies** of G are the mixed strategies of G' .
- In the same way, we also define the set of **Nash equilibria** of G .
- A **behavioral strategy** of player i is a probability distribution on C_h for each $h \in H_i$.
 - This is a strategy in which each player's choice at each information set is made independently of his choices at other information sets.

Strategies in extensive games

- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.
- **Mixed strategies** of G are the mixed strategies of G' .
- In the same way, we also define the set of **Nash equilibria** of G .
- A **behavioral strategy** of player i is a probability distribution on C_h for each $h \in H_i$.
 - This is a strategy in which each player's choice at each information set is made independently of his choices at other information sets.
 - So a behavioral strategy is a vector of probability distributions while a mixed strategy is a probability distribution over vectors.

Strategies in extensive games

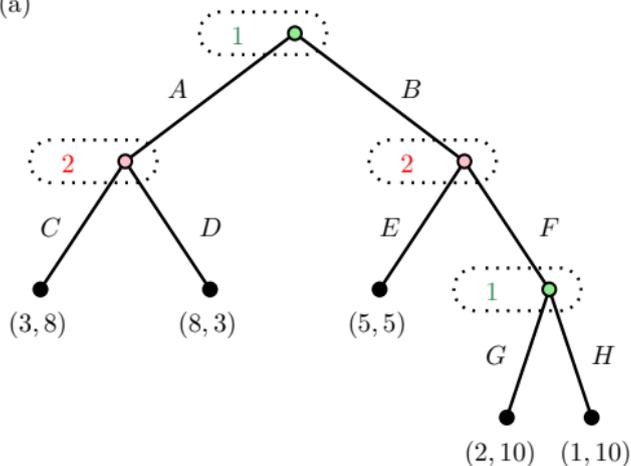
- A **pure strategy** for player i is a complete specification of which deterministic action to take at every information set belonging to i .
 - Formally, a pure strategy of player i is a vector $(c_h)_{h \in H_i}$ from the Cartesian product $\prod_{h \in H_i} C_h$.
 - Using pure strategies, we can transform an extensive game G into a normal-form game G' simply by tabulating all pure strategies of the players and recording the resulting expected payoffs.
- **Mixed strategies** of G are the mixed strategies of G' .
- In the same way, we also define the set of **Nash equilibria** of G .
- A **behavioral strategy** of player i is a probability distribution on C_h for each $h \in H_i$.
 - This is a strategy in which each player's choice at each information set is made independently of his choices at other information sets.
 - So a behavioral strategy is a vector of probability distributions while a mixed strategy is a probability distribution over vectors.
 - Unlike in mixed strategy, here a player might play different moves in different encounters of h .

Example: behavioral strategy

Example: behavioral strategy

- An example of a perfect-information game in extensive form (**part (a)**) and its normal-form (**part (b)**).

(a)



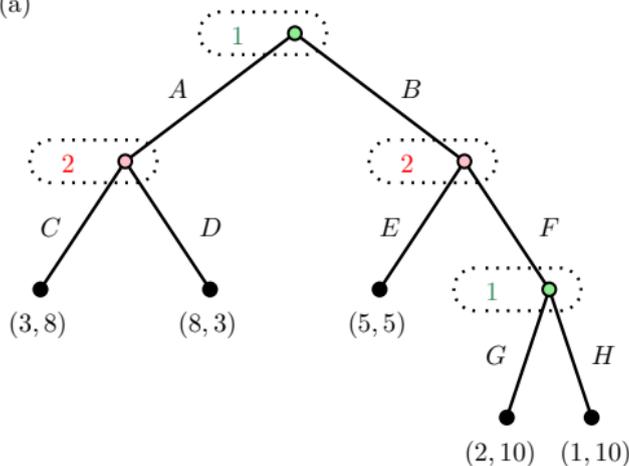
(b)

	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	(3, 8)	(3, 8)	(8, 3)	(8, 3)
(A, H)	(3, 8)	(3, 8)	(8, 3)	(8, 3)
(B, G)	(5, 5)	(2, 10)	(5, 5)	(2, 10)
(B, H)	(5, 5)	(1, 10)	(5, 5)	(1, 10)

Example: behavioral strategy

- An example of a perfect-information game in extensive form (**part (a)**) and its normal-form (**part (b)**).

(a)



(b)

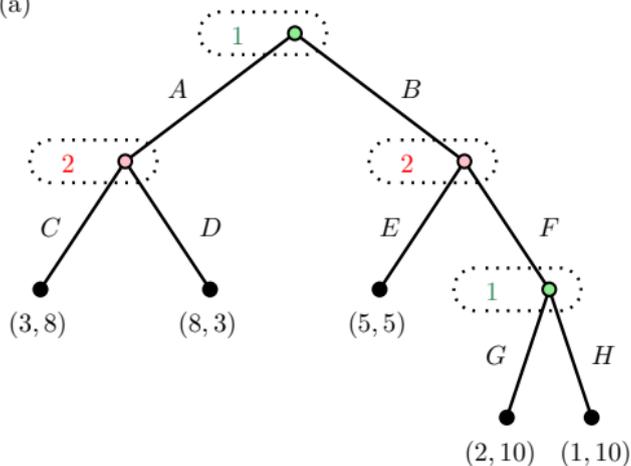
	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	(3,8)	(3,8)	(8,3)	(8,3)
(A, H)	(3,8)	(3,8)	(8,3)	(8,3)
(B, G)	(5,5)	(2,10)	(5,5)	(2,10)
(B, H)	(5,5)	(1,10)	(5,5)	(1,10)

- A strategy of player 1 that selects A with probability $\frac{1}{2}$ and G with probability $\frac{1}{3}$ is a **behavioral strategy**.

Example: behavioral strategy

- An example of a perfect-information game in extensive form (**part (a)**) and its normal-form (**part (b)**).

(a)



(b)

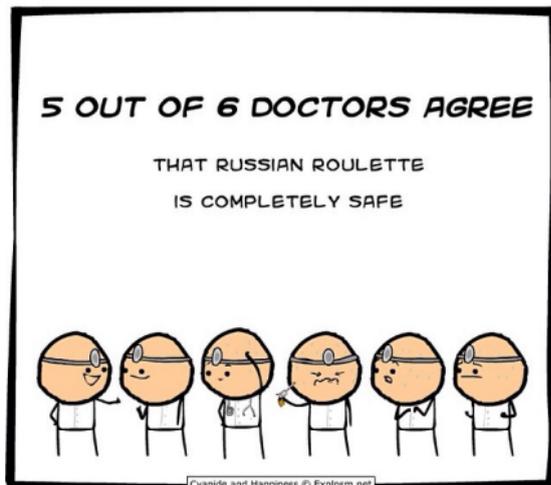
	(C, E)	(C, F)	(D, E)	(D, F)
(A, G)	(3,8)	(3,8)	(8,3)	(8,3)
(A, H)	(3,8)	(3,8)	(8,3)	(8,3)
(B, G)	(5,5)	(2,10)	(5,5)	(2,10)
(B, H)	(5,5)	(1,10)	(5,5)	(1,10)

- A strategy of player 1 that selects A with probability $\frac{1}{2}$ and G with probability $\frac{1}{3}$ is a **behavioral strategy**.
- The mixed strategy $(\frac{3}{5}(A, G), \frac{2}{5}(B, H))$ is **not a behavioral strategy** for 1 as the choices made by him at the two nodes are not independent.

Example: Russian roulette

Example: Russian roulette

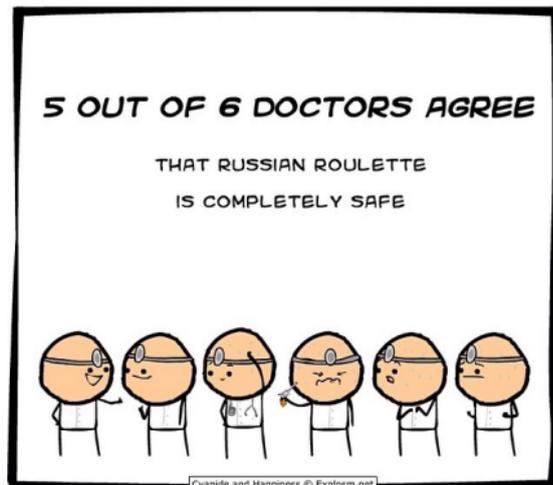
- We have two players with a six-shot revolver containing a single bullet. Each player has two moves: **shoot** or **give up**. If player gives up, he loses the game immediately. If he shoots, then he either dies or survives, in which case the other player is on turn.



Source: <https://www.memedroid.com/>

Example: Russian roulette

- We have two players with a six-shot revolver containing a single bullet. Each player has two moves: **shoot** or **give up**. If player gives up, he loses the game immediately. If he shoots, then he either dies or survives, in which case the other player is on turn.



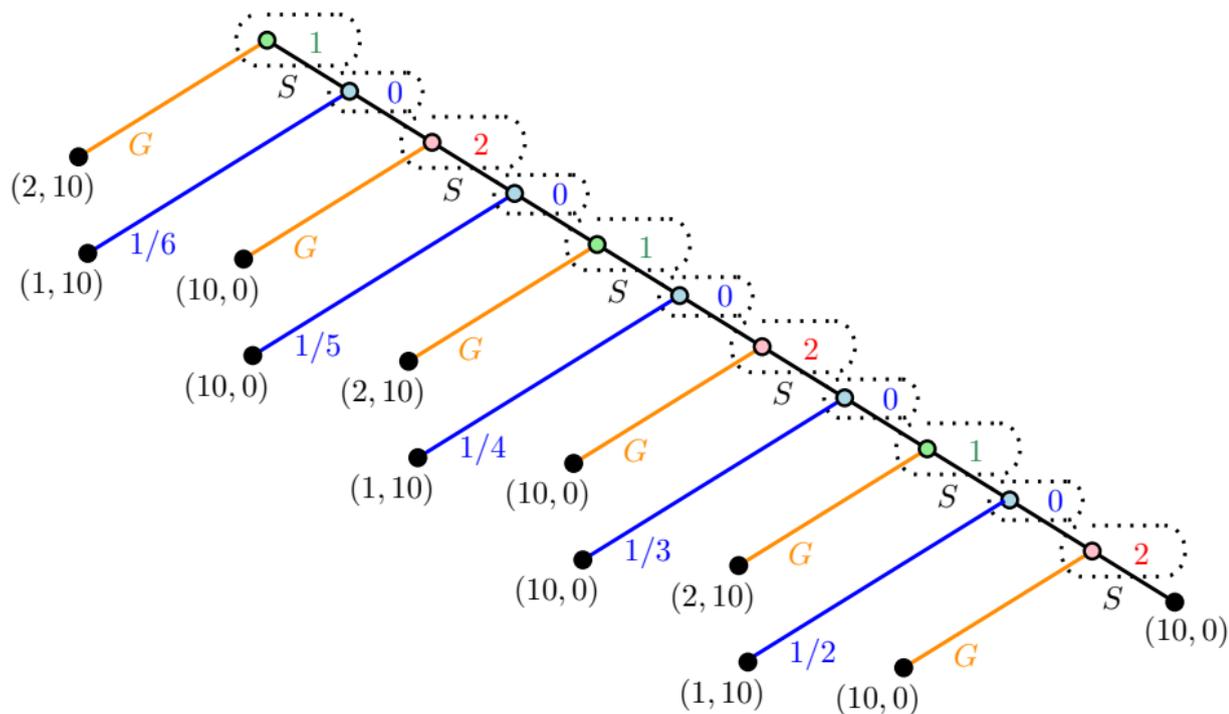
Source: <https://www.memedroid.com/>

- Consider that player 1 has payoffs $(10, 2, 1)$ for (Win, Loss, Death) and that player 2 has payoffs $(10, 0, 0)$.

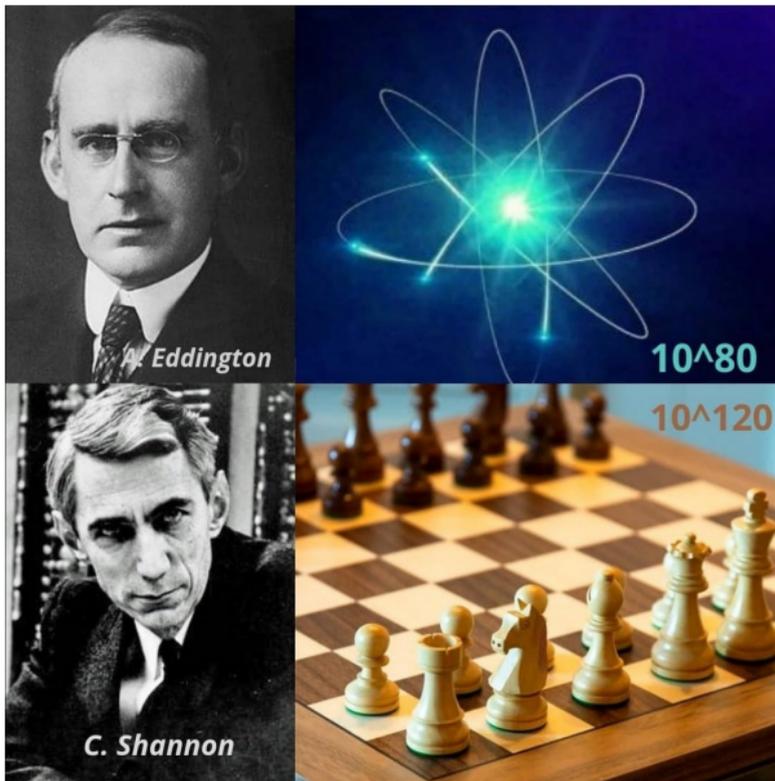
Example: Russian roulette

Example: Russian roulette

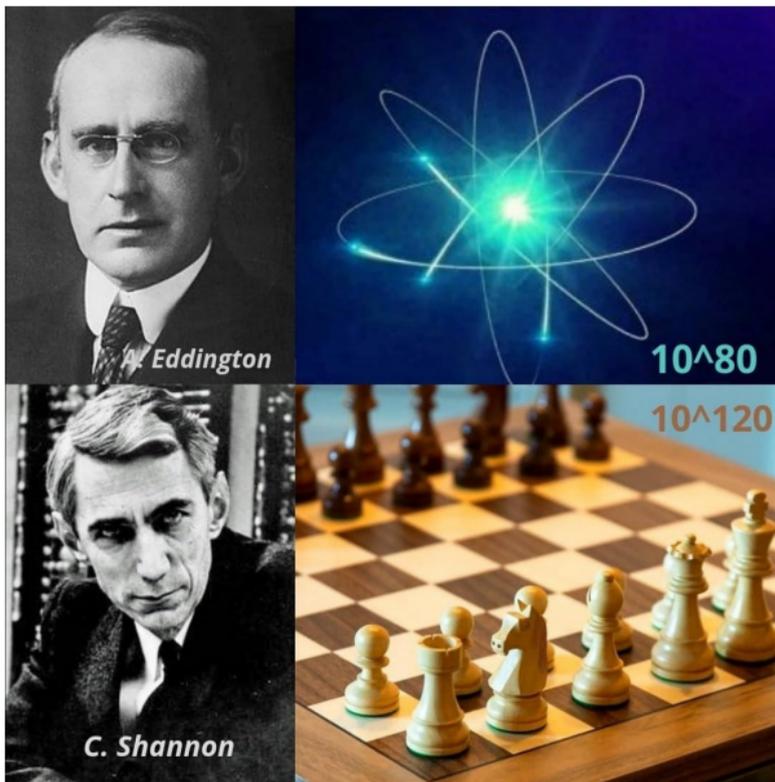
- The **Russian roulette** in the extensive form using the random player who plays according to a known behavior strategy β_0 .







Source: <https://twitter.com/curiosite12>



Source: <https://twitter.com/curiosite12>

Thank you for your attention.