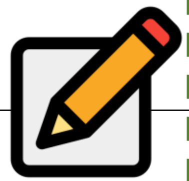
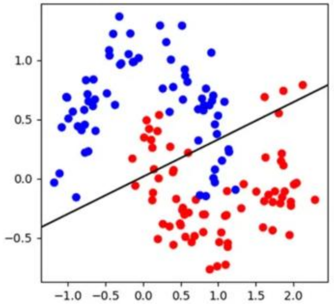
Swap Regret and Strategic Learning

Chen-Yu Wei

Recruiting

Known





Decision Algorithm

Applicants

(Strategic) Optimizer

Strategically change features

University

Learner

Setting: Optimizer and Learner

Optimizer / Learner's action sets: $[m]$ and $[n]$

Optimizer / Learner's utility: $u_O(i, j)$ and $u_L(i, j)$, $i \in [m], j \in [n]$

For $t = 1, 2, \dots, T$:

Optimizer choose $x_t \in \Delta_m$ and Learner chooses $y_t \in \Delta_n$ (simultaneously)

Draw actions $i_t \sim x_t$, $j_t \sim y_t$

Optimizer gains $u_O(i_t, j_t)$ and Learner gains $u_L(i_t, j_t)$

Reveal (x_t, y_t) to both players

Optimizer knows u_O and u_L; Learner only cares about u_L

Optimizer knows Learner's **algorithm** $\boldsymbol{A_L}$ where $A_L(x_1, x_2, \dots, x_{t-1}) = y_t$

No-Regret (NR) Learner

$$\text{Regret} = \max_{j \in [n]} \sum_{t=1}^{T} u_L(x_t, j) - \sum_{t=1}^{T} u_L(x_t, j_t) = o(T)$$

Many natural and standard algorithms are no-regret. For example,

Gradient ascent: $y_{t+1} = \Pi_{\Delta_n} (y_t + \eta_t \, u_L(x_t, \cdot))$

Exponential weights: $y_{t+1}(j) \propto y_t(j) \exp(\eta_t \, u_L(x_t, j))$

…

Mean-Based No-Regret (MB-NR) Learners

y_{t+1} is an increasing function of $\sum_{s=1}^{t} u_L(x_s,\ j)$, $\quad j \in [n]$

Follow the Regularized Leader: $y_{t+1} = \underset{y}{\operatorname{argmax}} \left\{ \sum_{s=1}^{t} u_L(x_s,\ y) + \frac{1}{\eta_t} H(y) \right\}$

Follow the Perturbed Leader: $j_{t+1} = \underset{j}{\operatorname{argmax}} \left\{ \sum_{s=1}^{t} u_L(x_s,\ j) + \frac{1}{\eta_t} \mathrm{perturb}_t(j) \right\}$

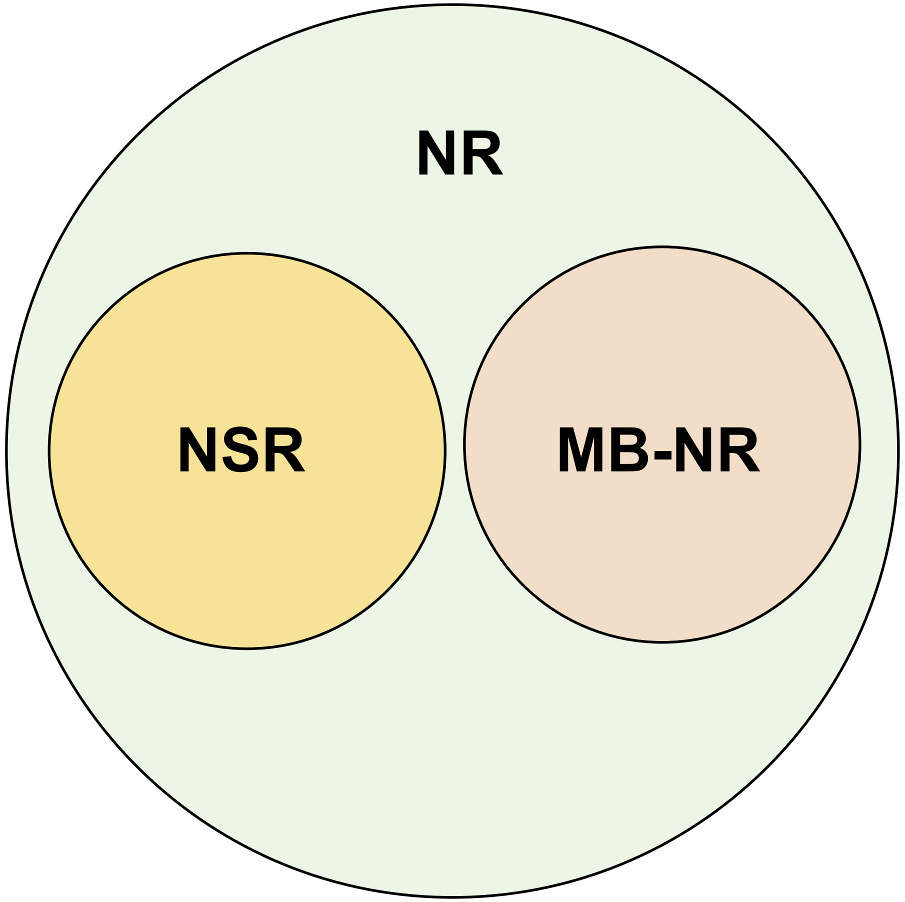
No-Swap-Regret (NSR) Learner

$$\text{SwapRegret} = \max_{\sigma \in [n] \to [n]} \sum_{t=1}^{T} u_L(x_t, \sigma(j_t)) - \sum_{t=1}^{T} u_L(x_t, j_t) = o(T)$$

$\text{Regret} \leq \text{SwapRegret}$ ∴ a NSR algorithm is also a NR algorithm.

NSR algorithms are NOT as natural as the NR algorithms we see previously.

There are general reductions to convert an NR into an NSR.


NR
NSR
MB-NR

Overview

Optimizer perspective: Optimizer can gain higher u_O **easier** when playing with a MB-NR Learner than playing with a NSR Learner.
Yuan Deng, Jon Schneider, Balusubramanian Sivan. Strategizing against No-regret Learners. NeurIPS 2019.

Learner perspective: Optimizer can cause lower u_L **easier** when playing with a MB-NR Learner than playing with a NSR Learner.

Eshwar Arunachaleswaran, Natalie Collina, Jon Schneider. Pareto-Optimal Algorithms for Learning in Games. EC 2024.

Optimizer can lead to higher u_O and u_L **easier** when playing with a MB-NR Learner than playing with a NSR Learner.

Guruganesh et al. Contracting with a Learning Agent. NeurIPS 2024.

EC 2025 “Swap Regret and Strategic Learning” Workshop (link)

Optimizer’s Perspective

Stackelberg Value for the Optimizer

$$\boxed{V = \max_{x} \max_{y \in \mathrm{BR}(x)} u_O(x, y)}$$

where $\mathrm{BR}(x) = \left\{ y: \quad u_L(x, y) = \max_{y'} u_L(x, y') \right\}$

$$u_O(A_O, A_L) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} u_O(x_t, y_t)$$

Theorem

If $A_L \in$ NR then $\max_{A_O} u_O(A_O, A_L) \geq V$ for any game

If $A_L \in$ NSR then $\max_{A_O} u_O(A_O, A_L) \leq V$ for any game

If $A_L \in$ MB-NR then $\max_{A_O} u_O(A_O, A_L) \geq V + \mathrm{const}$ for some game

If $A_L \in$ NSR then $\max\limits_{A_O} u_O(A_O, A_L) \leq V$ for any game

Consider a fixed Learner's action $j \in [n]$:

Define $\alpha_j \in \Delta_m$ as the Optimizer's action distribution when Learner chooses j:

$$\alpha_j = \frac{\sum_{t=1}^{T} \mathbb{I}[j_t = j] x_t}{T_j} \qquad \text{where } T_j = \sum_{t=1}^{T} \mathbb{I}[j_t = j]$$

$$\frac{1}{T}\sum_{t=1}^{T}\left(u_L(x_t, \sigma(j_t)) - u_L(x_t, j_t)\right) = \sum_{j \in [n]} \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[j_t = j]\left(u_L(x_t, \sigma(j)) - u_L(x_t, j)\right)$$

$$= \sum_{j \in [n]} \frac{T_j}{T}\left(u_L\left(\alpha_j, \sigma(j)\right) - u_L(\alpha_j, j)\right)$$

As A_L has No Swap Regret, for $j \notin \mathrm{BR}(\alpha_j)$, we have $T_j = o(T)$

Optimizer utility

$$= \sum_{j \in [n]} \frac{T_j}{T} u_O(\alpha_j, j)$$

$$= \sum_{j:\ j \in \mathrm{BR}(\alpha_j)} \frac{T_j}{T} u_O(\alpha_j, j) + \sum_{j:\ j \notin \mathrm{BR}(\alpha_j)} \frac{T_j}{T} u_O(\alpha_j, j)$$

$$\leq \sum_{j:\ j \in BR(\alpha_j)} \frac{T_j}{T} \times V + \sum_{j:\ j \notin BR(\alpha_j)} \frac{T_j}{T} \times 1$$

$$= V$$

If $A_L \in$ MB-NR then $\max_{A_O} u_O(A_O, A_L) \geq V + \text{const}$ for some game

	Left	Middle	Right
Up	0, ε	-2, -1	-2, 0
Down	0, -1	-2, 1	2, 0

The Stackelberg value is ***V* = 0**: (½ *Up* + ½ *Down*, *Right*)

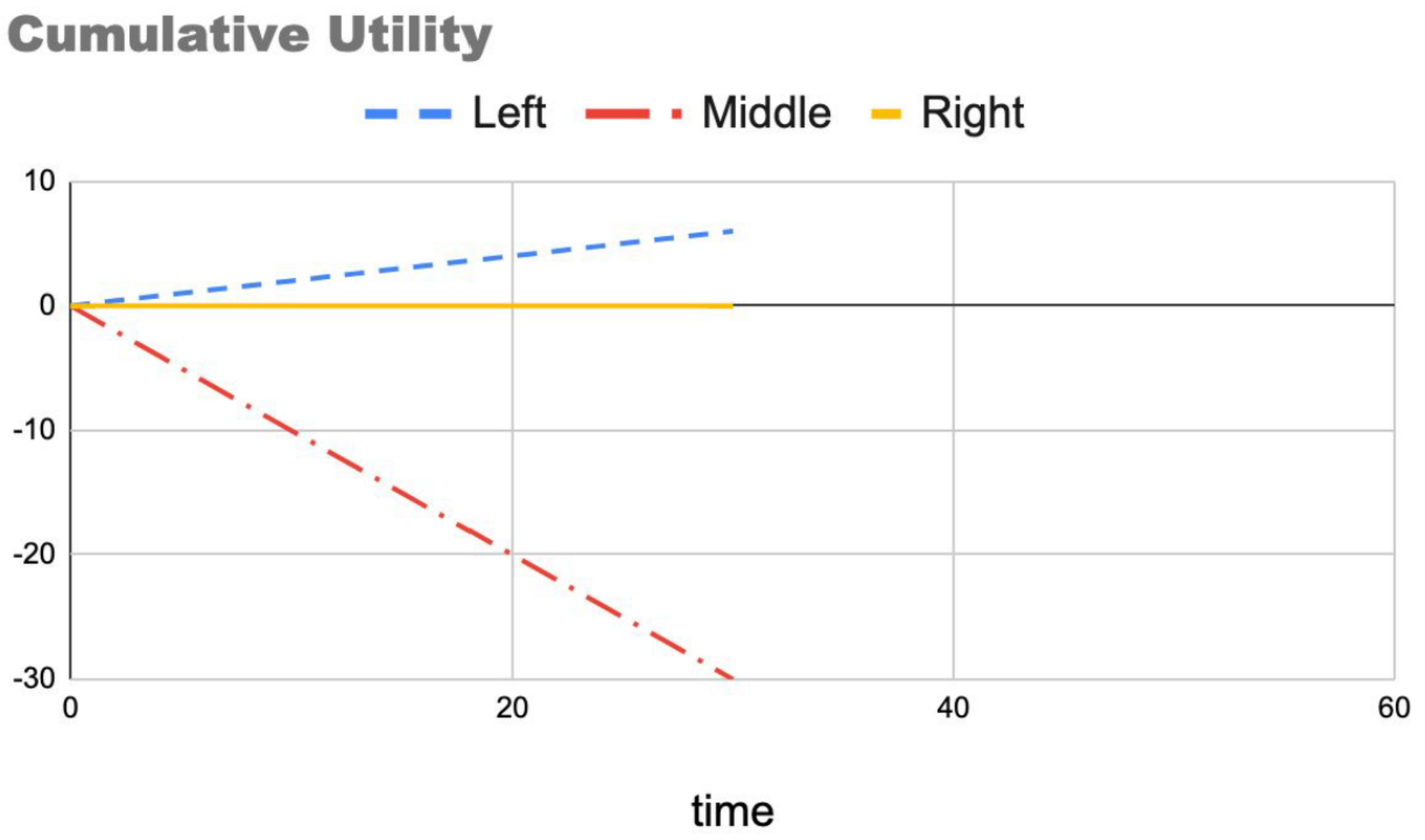
Taken from this tutorial

Against mean-based algorithm

	Left	Middle	Right
Up	0, ε	-2, -1	-2, 0
Down	0, -1	-2, 1	2, 0

Play *Up* for T/2 rounds

- learner plays *Left*
- earn 0 per round



Against mean-based algorithm

	Left	Middle	Right
Up	0, ε	-2, -1	-2, 0
Down	0, -1	-2, 1	2, 0

Play *Up* for T/2 rounds

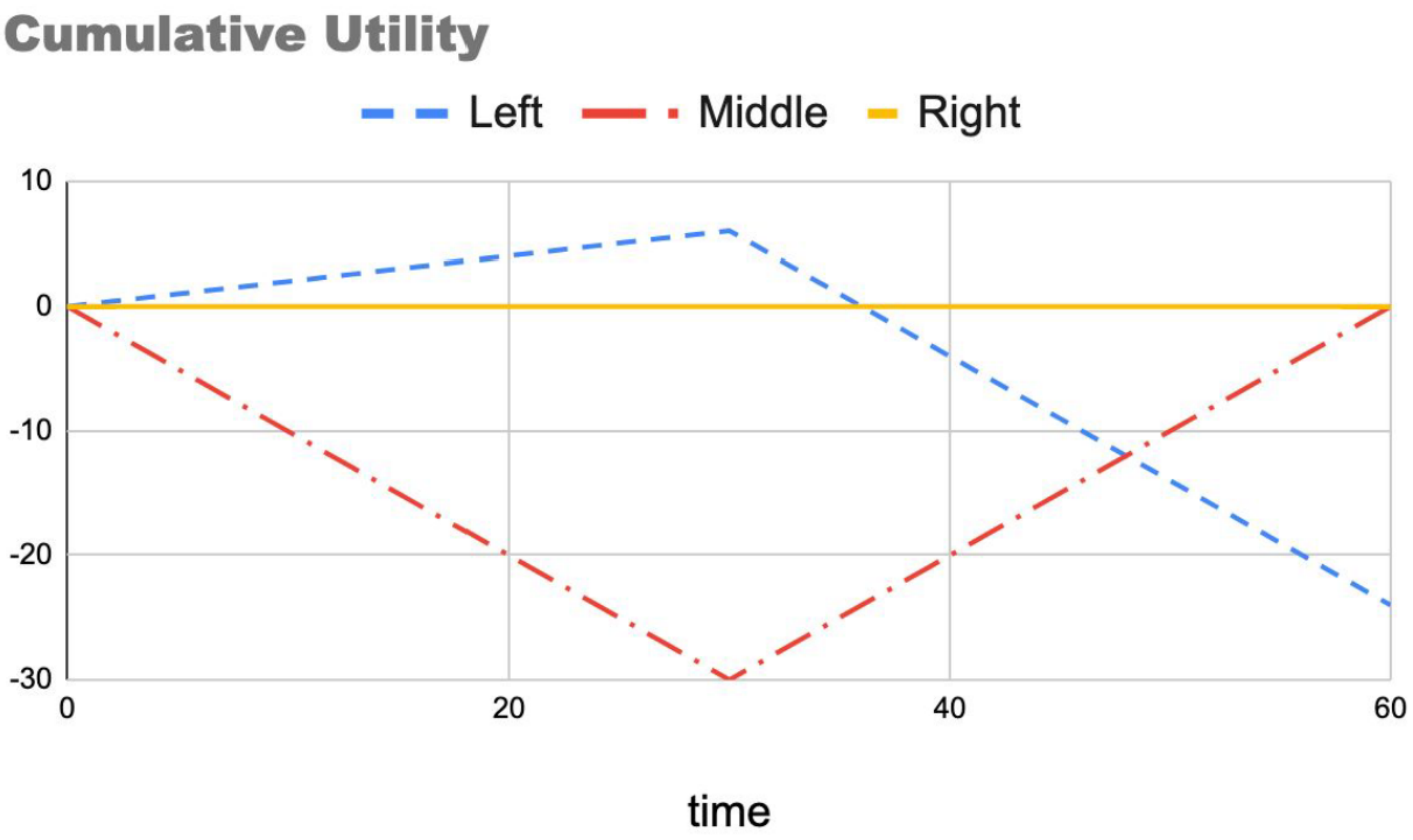
- learner plays *Left*
- earn 0 per round

then

Play *Down* for T/2 rounds

- learner plays *Right*
- earn ***2*** per round

(for most rounds)



Learner’s Perspective

Pareto Dominance

$$u_L(A_O, A_L) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} u_L(x_t, y_t)$$

For a Learner utility u_L, we say $\boldsymbol{A_L}$ **is Pareto-dominated by** $\boldsymbol{A'_L}$ if for all u_O,

$$u_L(\mathrm{BR}(A'_L), A'_L) \geq u_L(\mathrm{BR}(A_L), A_L) \qquad (\star)$$

where $\mathrm{BR}(A_L)$ is an Optimizer algorithm A_O such that $u_O(A_O, A_L) \geq \max_A u_O(A, A_L)$ (breaking ties in favor of the Learner)

and the inequality $(\star)$ holds strictly for at least one u_O.

Theorem

There is an u_L such that any $A_L \in$ MB-NR is Pareto-dominated.

For any u_L, any $A_L \in$ NSR is NOT Pareto-dominated (i.e., Pareto-optimal)

Geometric Interpretation

Menu

Given Learner's algorithm A_L

- Any Optimizer sequence $(x_1, x_2, \ldots, x_T)$ induces a *correlated strategy profile* (CSP) $\frac{1}{T}\sum_{t=1}^{T} x_t \otimes y_t \in \Delta_{mn}$
(recall that $y_t = A_L(x_1, .., x_{t-1})$)

- The menu $M(A_L) \in \Delta_{mn}$ produced by A_L is defined as

$$M(A_L) = \mathrm{ConvexHull}\left\{\frac{1}{T}\sum_{t=1}^{T} x_t \otimes y_t : \quad \text{all possible } x_1, \ldots, x_T\right\}$$

By selecting $x_1, .., x_T$ (essentially selecting a point $\phi \in M(A_L)$), the Optimizer can control the CSP induced by the players

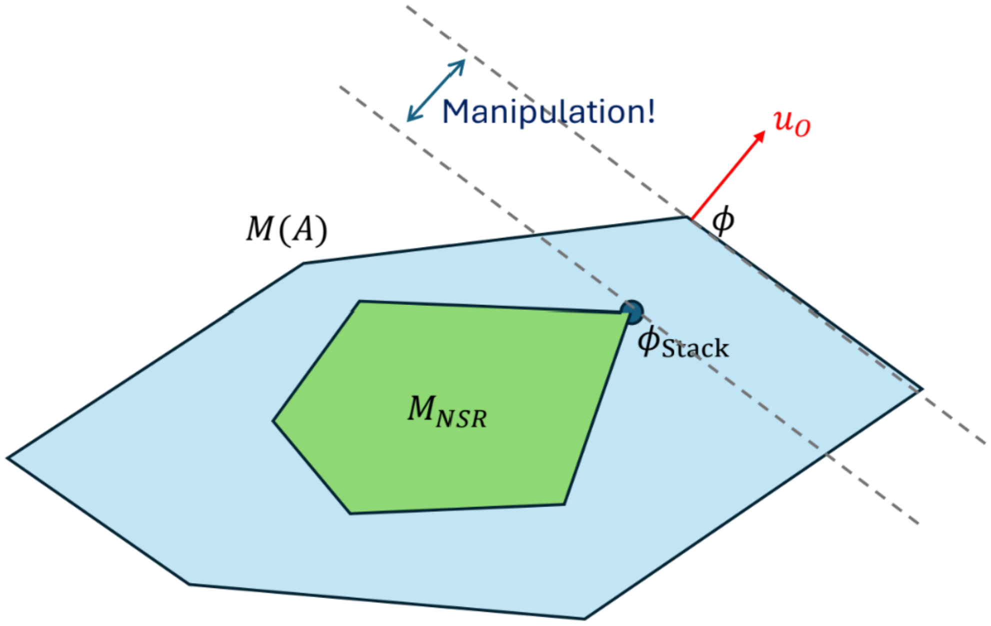
CSP directly affects utility $u_{L/O}(A_O, A_L) = \sum_{i,j} \phi_{ij}\, u_{L/O}(i, j)$

Menu

Lemma

All $A_L \in$ NSR induces the same menu $M(A_L) = M_{\text{NSR}}$

All $A_L \in$ NR induces a menu $M(A_L) \supseteq M_{\text{NSR}}$



Proof of the Lemma (1/2)

Claim: For any $A_L \in$ **NSR**, $M(A_L)$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in \mathrm{BR}(x)$.

Proof:

1. Any CSPs of the form $x \otimes \mathrm{BR}(x)$ is contained in $M(A_L)$

2. Any point $\phi \in M(A_L)$ can be written as a convex combination of $x \otimes \mathrm{BR}(x)$

$$\frac{1}{T}\sum_{t=1}^{T} x_t \otimes y_t \approx \frac{1}{T}\sum_{t=1}^{T} x_t \otimes e_{j_t} = \sum_{j\in[n]} \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[j_t = j]\, x_t \otimes e_j = \sum_{j\in[n]} \frac{T_j}{T}\, \alpha_j \otimes e_j$$

$$\alpha_j \triangleq \frac{\sum_{t=1}^{T} \mathbb{I}[j_t = j] x_t}{T_j}$$

$$T_j \triangleq \sum_{t=1}^{T} \mathbb{I}[j_t = j]$$

As A_L is NSR, either $\frac{T_j}{T} \to 0$ or $j = \mathrm{BR}(\alpha_j)$

$$\Rightarrow \frac{1}{T}\sum_{t=1}^{T} x_t \otimes y_t = \sum_{j} \frac{T_j}{T}\, \alpha_j \otimes \mathrm{BR}(\alpha_j)$$

Proof of the Lemma (2/2)

Claim: For any $A_L \in$ **NR**, $M(A_L)$ contains all CSPs of the form $x \otimes \mathrm{BR}(x)$

Summary

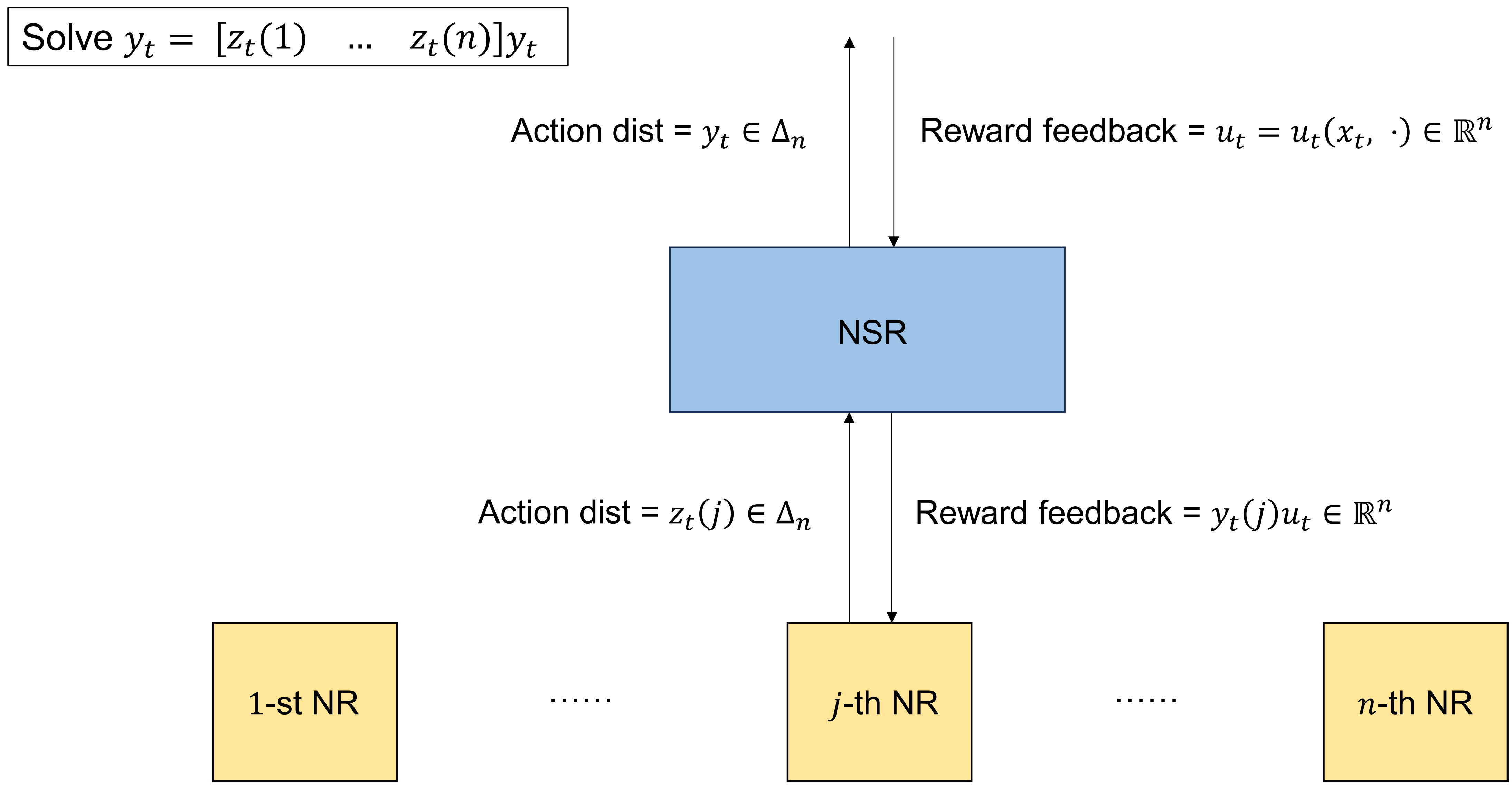
- The *correlated strategy profile* (CSP) induced by any Optimizer and a NSR Learner is always of the form:

$$\frac{1}{T}\sum_{t=1}^{T} x_t \otimes y_t = \sum_{x} c_x \; x \otimes \mathrm{BR}(x) + o(1)$$

This makes the time-averaged profile just like one-shot Stackelberg game.

This leaves less room of manipulation (good or bad) by the Optimizer.

Reduction: NSR to NR



Blum and Mansour. From External to Internal Regret. JMLR 2007.