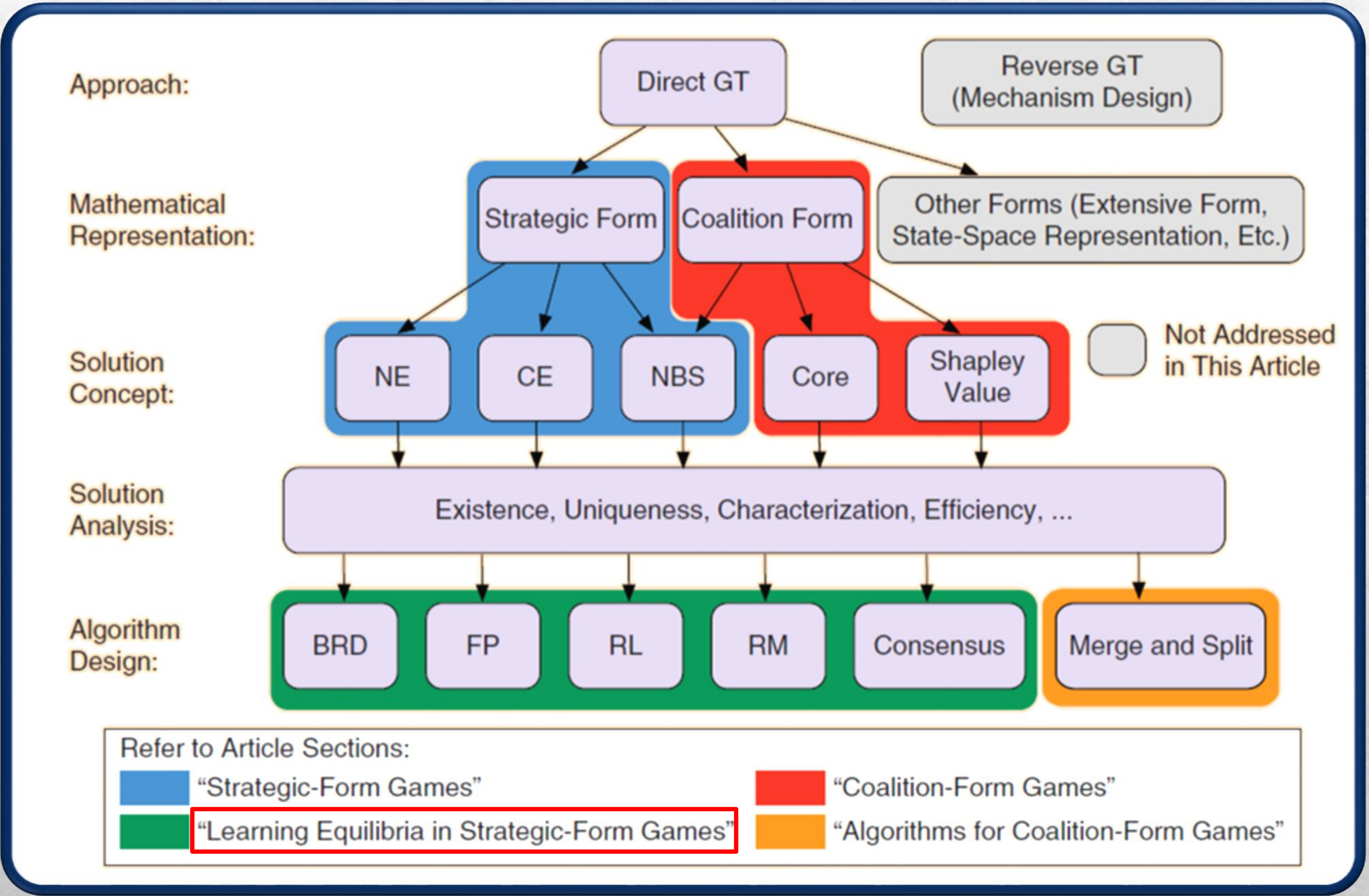
Game Theory for Networks

Learning Algorithms

董皓文 2016.5.2



Approach:
Direct GT
Reverse GT (Mechanism Design)
Mathematical Representation:
Strategic Form
Coalition Form
Other Forms (Extensive Form, State-Space Representation, Etc.)
Solution Concept:
NE
CE
NBS
Core
Shapley Value
Not Addressed in This Article
Solution Analysis:
Existence, Uniqueness, Characterization, Efficiency, ...
Algorithm Design:
BRD
FP
RL
RM
Consensus
Merge and Split
Refer to Article Sections:
"Strategic-Form Games"
"Coalition-Form Games"
"Learning Equilibria in Strategic-Form Games"
"Algorithms for Coalition-Form Games"

Outline

- BRD – Best Response Dynamics
- RL - Reinforcement Learning
- RM – Regret Matching Learning
- Performance and Efficiency Comparison
- Consensus Algorithm

Best Response Dynamics

- Various disciplines
- Examples:
  - Gauss-Seidel Model
  - Lloyd-Max algorithm
  - Cournot tâtonnement
  - IWFA algorithms
  - FP algorithms

Gauss-Seidel Model

- Examples

observations

$$\begin{pmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \end{pmatrix}$$

actions of the players

<u>Step 1</u> - Solve $x_1(t+1)$:

iterations

$$a_{11}x_1(t+1) + a_{12}x_2(t) - y_1 = 0$$

<u>Step 2</u> - Solve $x_2(t+1)$:

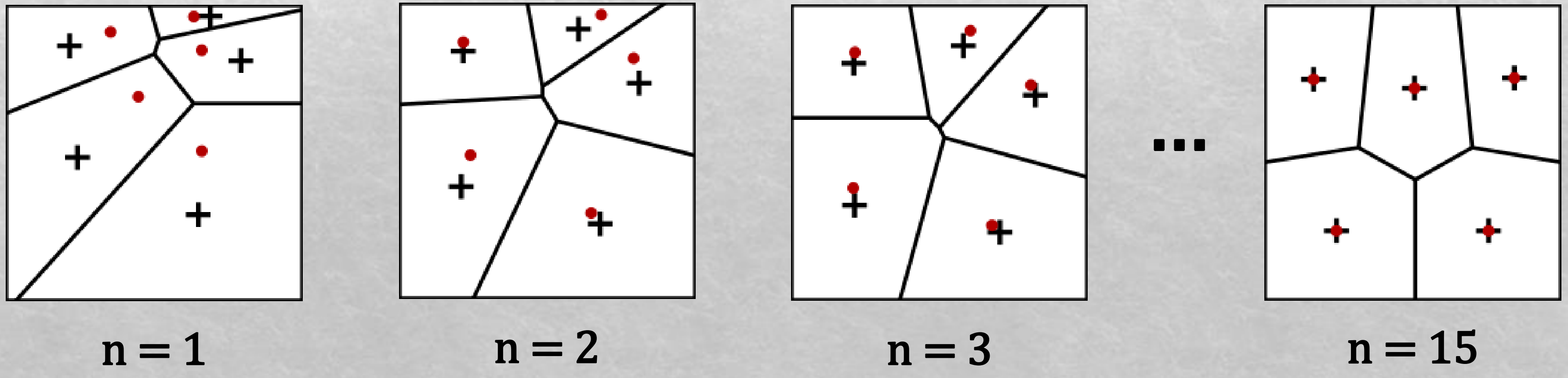
$$a_{21}x_1(t+1) + a_{22}x_2(t+1) - y_2 = 0$$

Lloyd-Max Algorithm

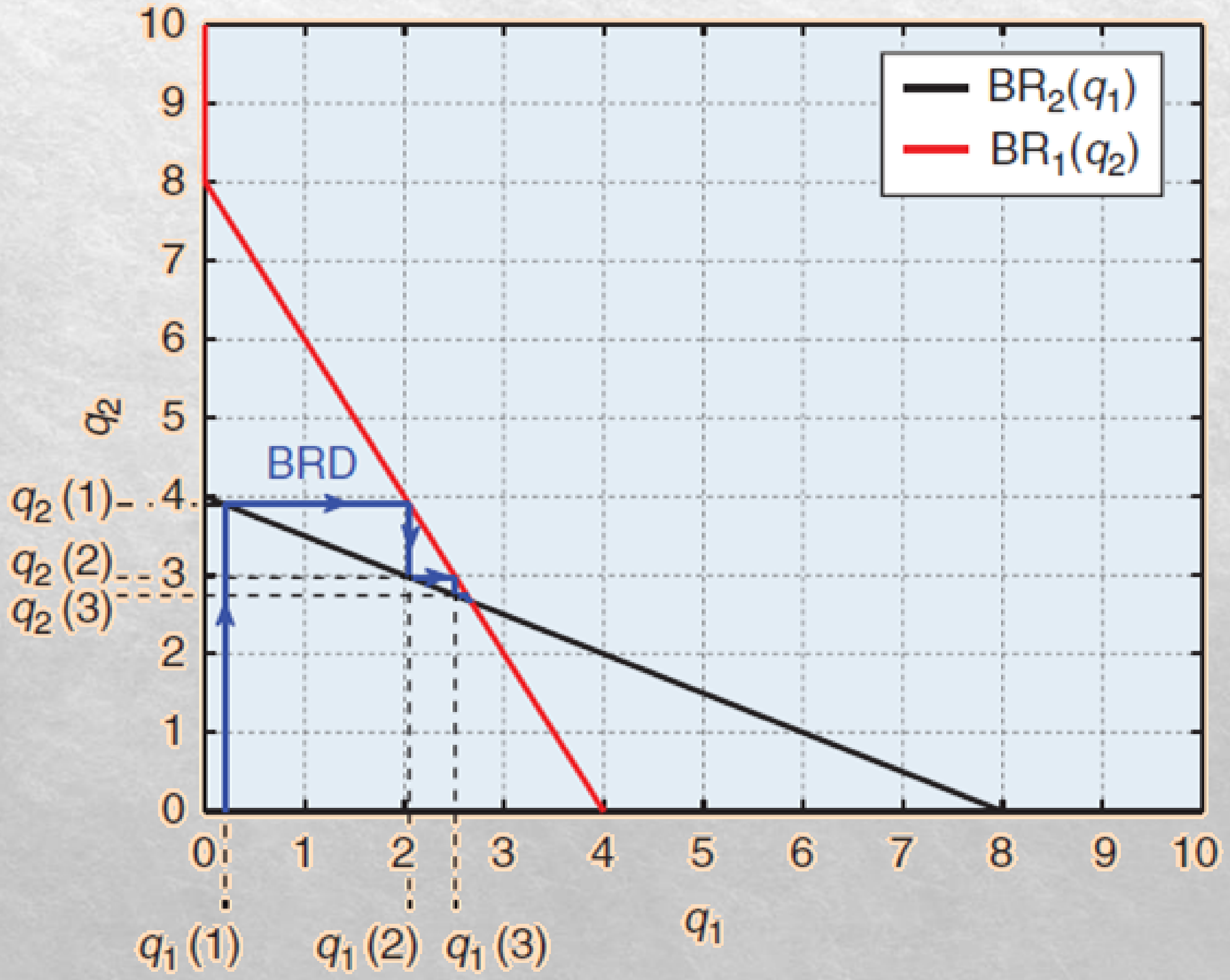
- Examples
  - Signal quantizer - choosing how to partition the source signal space into cells or regions and choosing a representative for each of them
  - Goal: **minimize the distortion**

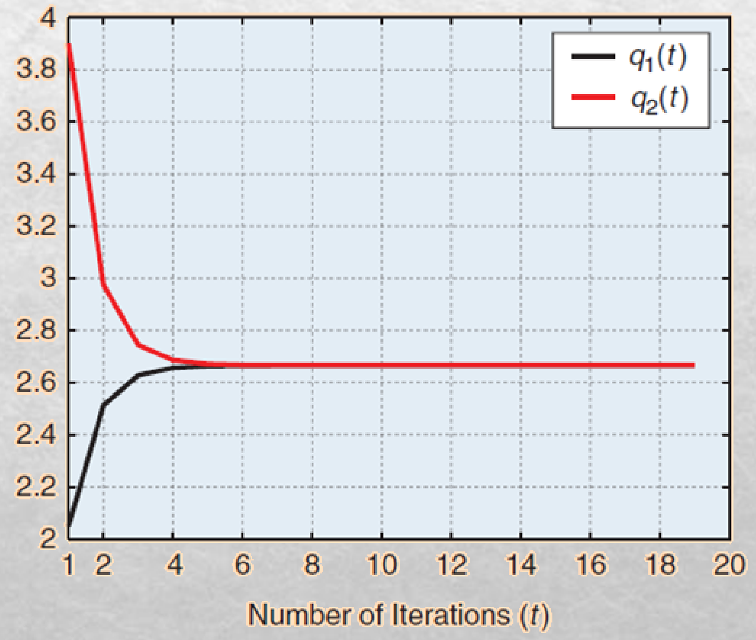
Lloyd-Max Algorithm

- Iterations
  - Step 1 – fix a set of regions and compute the best representatives in the sense of the distortion
  - Step 2 – for these representatives, one updates the regions so that the distortion is minimized



Cournat Tâtonnement





BRD Formulation

$$\boldsymbol{a_k(t+1) \in BR_k[\underbrace{\underline{a_1(t+1), a_2(t+1), \ldots, a_{k-1}(t+1)}, \underline{a_k(t), \ldots a_K(t)}}_{\text{take turns}}]}$$

$$\boldsymbol{a_k(t+1) \in BR_k[\underbrace{\underline{a_{-k}(t)}}_{\text{update simultaneously}}]}$$

BRD Algorithm

Algorithm 1: The BRD.

---

initialization

set $t = 0$

initialize $a_k(0) \in \mathcal{S}_k$ for all players $k \in \mathcal{K}$ (e.g., using a random initialization)

iterations

repeat

for $k = 1$ to K do

update $a_k(t+1)$ using (22) or (23)

end for

update $t = t + 1$

until $|a_k(t) - a_k(t-1)| \leq \varepsilon$ for all $k \in \mathcal{K}$

convergence check

discrete case: ε =0
continuous case: ε = certain threshold

BRD Convergence

- Theorems
  - In potential and supermodular games, the sequential BRD converges to a pure NE with probability one.
  - If the BRs of a strategic-form game are standard functions, them the BRD converges to the unique pure NE with probability one.

Reinforcement Learning

- A player receives a numerical utility signal and updates its strategy accordingly
- It's shown that feeding back to the players only the realizations of their utilities is enough to drive seemingly complex interactions to a steady state or, at least, to a predictable evolution of the state

Reinforcement Learning

$$\mathbf{1}_{\{a_k(t)=a_{k,n}\}} = \begin{cases} 1, if\, a_k(t) = a_{k,n} \\ 0, otherwise \end{cases}$$

$$\pi_{k,a_{k,n}}(t+1) = \pi_{k,a_{k,n}}(t) + \lambda_k^{RL}(t)u_k(t)\left[\mathbf{1}_{\{a_k(t)=a_{k,n}\}} - \pi_{k,a_{k,n}}(t)\right]$$

$$= \begin{cases} \pi_{k,a_{k,n}}(t) + \lambda_k^{RL}(t)u_k(t)\left(1 - \pi_{k,a_{k,n}}(t)\right), if\ a_k(t) = a_{k,n}(t) \\ \pi_{k,a_{k,n}}(t) - \lambda_k^{RL}(t)u_k(t)\pi_{k,a_{k,n}}(t), otherwise \end{cases}$$

- $\lambda_k^{RL}(t)$ is a known function that regulates the learning rate of player k, where $0 < \lambda_k^{RL}(t) < 1$ and $\lambda_k^{RL}(t)u_k(t) < 1$
- RL algorithm usually requires a large number of iterations to converge compared to the BRD algorithm.

RM Learning

Algorithm 2: The regret-matching-learning algorithm.

initialization
- set $t = 0$
- initialize $\pi_k(0)$ s.t. $\sum_{n=1}^{N_k} \pi_{k,n}(0) = 1$ for all players $k \in \mathcal{K}$
- (e.g., using a random initialization)

iterations
```
repeat
  for k = 1 to K do
    for n = 1 to N_k do
      update r_{k,n}(t+1) using (36)
    end for
    for n = 1 to N_k do
      update π_{k,n}(t+1) using (37)
    end for
    choose a_k(t+1) according to the distribution π_k(t+1)
  end for
  update t = t+1
until |a_k(t) − a_k(t−1)| ≤ ε for all k ∈ K
```
(convergence check)

$$r_{k,a_{k,n}}(t+1) = \frac{1}{t}\sum_{t'=1}^{t}\Big(u_k\big(a_{k,n}, a_{-k}(t')\big) - u_k\big(a_k(t'), a_{-k}(t')\big)\Big)$$

the regret at time t for player k

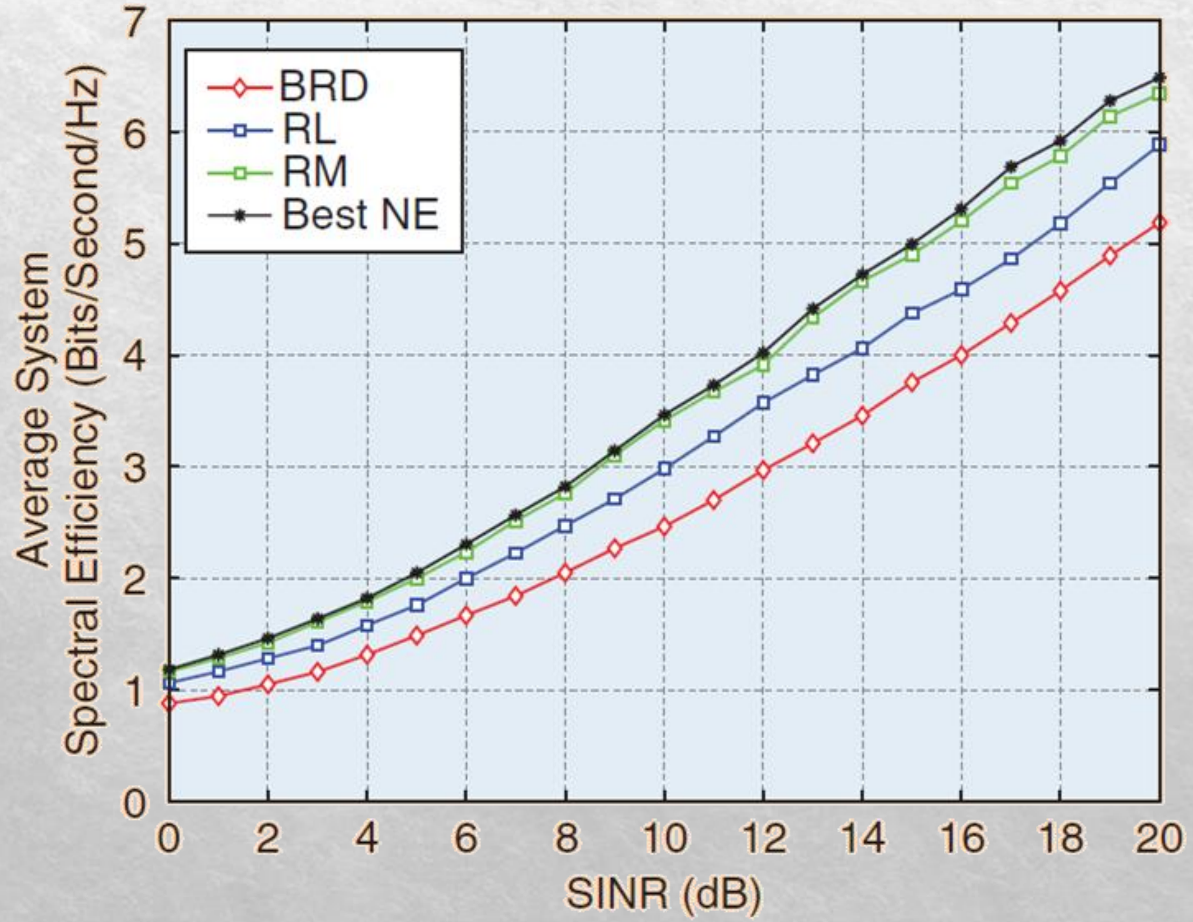
$$\pi_{k,a_{k,n}}(t+1) = \frac{\left[r_{k,a_{k,n}}(t+1)\right]^+}{\sum_{n'=1}^{N_k}\left[r_{k,a_{k,n}}(t+1)\right]^+}$$

$$[x]^+ = max(0, x)$$

Comparison

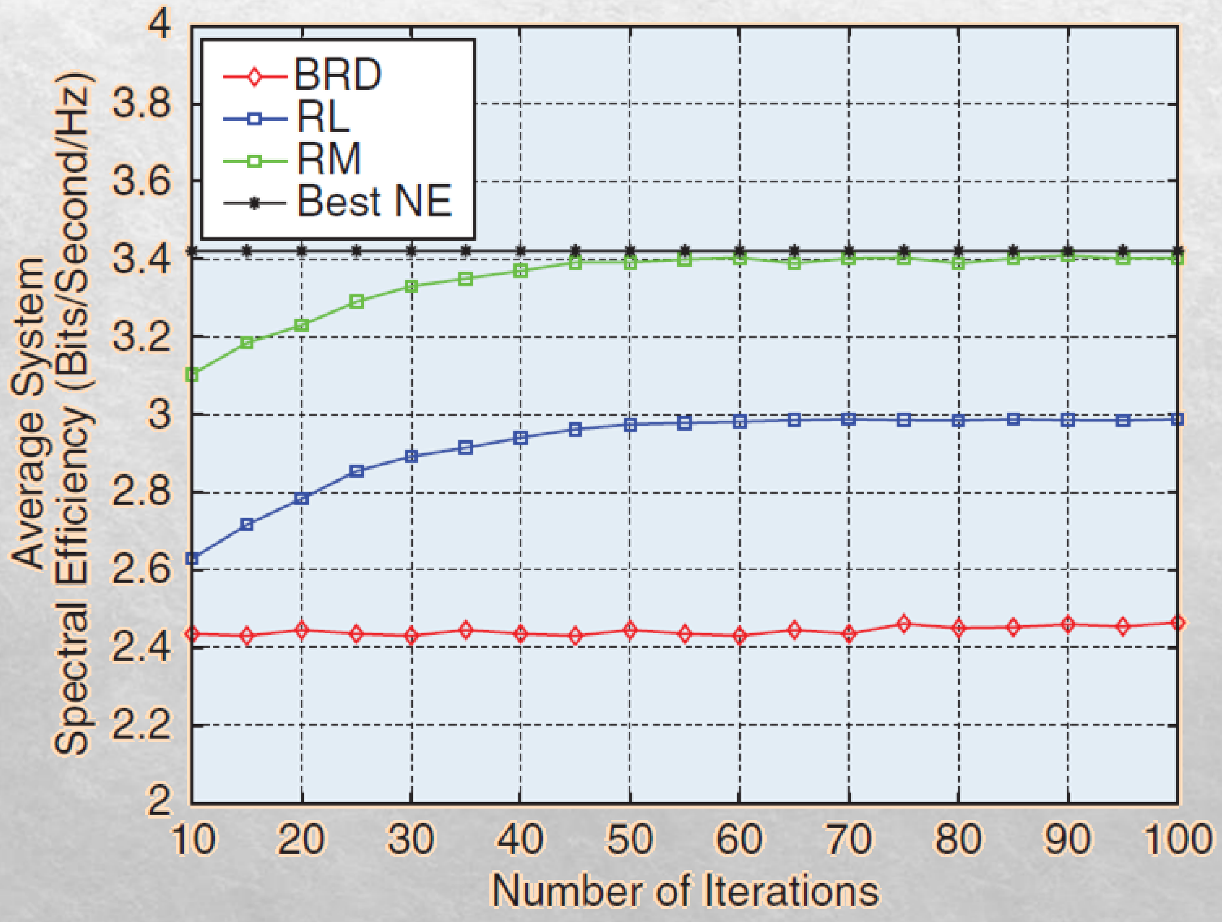
	BRD	RL	RM
Action Sets	continuous or discrete	discrete	discrete
Convergence	sufficient conditions	sufficient conditions	always
Convergence Points	pure NE or boundary points	pure NE or boundary points	CCE
Observation (typically required)	actions of the others	value of the utility function	actions of the others
Knowledge (typically required)	utility functions and action sets	action sets	utility functions and action sets
Convergence Speed	fast	slow	medium
Performance (typical)	low	low	medium

Comparison – Performance



- Under different noise level, the RM algorithm has higher spectral efficiency than the RM algorithm, which is better than the BRD algorithm.
- Under little noise level, the RM algorithm has almost the same spectral efficiency as the best NE does. Under higher noise level, the RM algorithm is still closed to the best NE.

Comparison – Convergence Speed

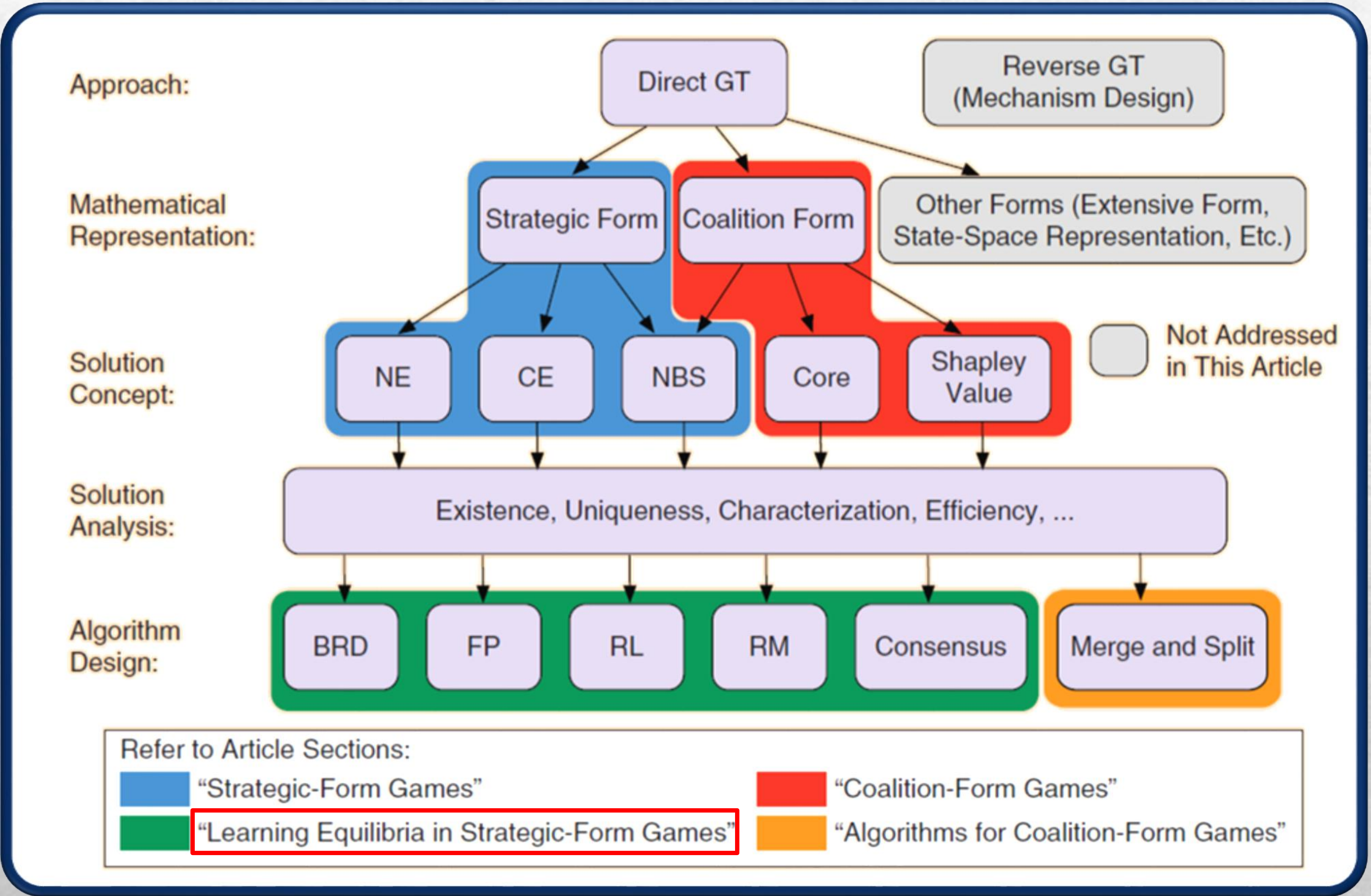


- The RL and RM algorithms require larger number of iterations to converge than the BRD algorithm does.
- The BRD algorithm converges in 10 iterations.
- The RL algorithms converges in 45 iterations.
- The RM algorithms converges in 60 iterations

Consensus Algorithms

$$\boldsymbol{a_k(t+1) = a_k(t) + \sum_{j \in \mathcal{A}_k} \beta_{k,j}\left(a_j(t) - a_k(t)\right)}$$

- Requires a well-determined topology for the network and explicit knowledge of the actions chosen by the other players.
- Assume $\forall k \in \mathcal{K}$, $a_k \in \mathbb{R}$, and the networks should be designed to operated at a given point $a^* = (a_1^*, \ldots, a_k^*) \in \mathbb{R}^K$ referred to as consensus.


Approach:
Direct GT
Reverse GT (Mechanism Design)
Mathematical Representation:
Strategic Form
Coalition Form
Other Forms (Extensive Form, State-Space Representation, Etc.)
Solution Concept:
NE
CE
NBS
Core
Shapley Value
Not Addressed in This Article
Solution Analysis:
Existence, Uniqueness, Characterization, Efficiency, ...
Algorithm Design:
BRD
FP
RL
RM
Consensus
Merge and Split
Refer to Article Sections:
“Strategic-Form Games”
“Coalition-Form Games”
“Learning Equilibria in Strategic-Form Games”
“Algorithms for Coalition-Form Games”

Thank you for your kindly attention!

Any Question?