COSC-6590/GSCS-6390

Games: Theory and Applications

Lecture 13 - Classes of Potential Games

Luis Rodolfo Garcia Carrillo

School of Engineering and Computing Sciences
Texas A&M University - Corpus Christi, USA

Table of contents

Identical Interests Plus Dummy Games

Identical Interests Plus Dummy Games

Suppose we can write the outcomes J_i of game G as

$$J_i(\gamma_i, \gamma_{-i}) = \phi(\gamma_i, \gamma_{-i}) + Q_i(\gamma_{-i}) \quad \forall \gamma_i \in \Gamma_i, \gamma_{-i} \in \Gamma_{-i}, \quad i \in \{1, 2, \ldots, N\}$$

for appropriate functions $\phi(\gamma)$, $Q_i(\gamma_{-i})$, $i \in \{1, 2, \ldots, N\}$.

G is the sum of an identical interests (II) game with outcomes $\phi(\gamma)$ and a dummy game with outcomes $Q_i(\gamma_{-i})$.

From **Proposition 12.5** we can conclude that

- G is an exact potential game
- ϕ is an exact potential for G

Other constructions lead to potential games, but arise more naturally in the context of specific applications.

Decoupled Plus Dummy Games

Decoupled Plus Dummy Games

A game H with outcomes H_i is a **decoupled game** if the outcome for player P_i only depends on P_i's own policy γ_i, i.e., if

$$H_i(\gamma_i, \gamma_{-i}) = H_i(\gamma_i, \bar{\gamma}_{-i}) = H_i(\gamma_i),$$
$$\forall \gamma_i \in \Gamma_i, \gamma_{-i}, \bar{\gamma}_{-i} \in \Gamma_{-i}, i \in \{1, 2, \ldots, N\}$$

Decoupled games are potential games with potential

$$\phi(\gamma) = \sum_{i=1}^{N} H_i(\gamma_i), \quad \forall \gamma_i \in \Gamma_i$$

Suppose we can write the outcomes J_i of game G as

$$J_i(\gamma_i, \gamma_{-i}) = H_i(\gamma_i) + Q_i(\gamma_{-i}) \quad \forall \gamma_i \in \Gamma_i, \gamma_{-i} \in \Gamma_{-i}, \quad i \in \{1, 2, \ldots, N\}$$

for appropriate functions $H_i(\gamma_i)$, $Q_i(\gamma_{-i})$, $i \in \{1, 2, \ldots, N\}$.

Decoupled Plus Dummy Games

$$J_i(\gamma_i, \gamma_{-i}) = H_i(\gamma_i) + Q_i(\gamma_{-i}) \qquad \forall \gamma_i \in \Gamma_i, \gamma_{-i} \in \Gamma_{-i}, \quad i \in \{1, 2, \ldots, N\}$$

Therefore, G is the sum of

- a decoupled game with outcomes $H_i(\gamma_i)$, and
- a dummy game with outcomes $Q_i(\gamma_{-i})$.

From **Proposition 12.4**: G is an exact potential game.

Since any dummy game admits a zero potential, then

$$\phi(\gamma) = \sum_{i=1}^{N} H_i(\gamma_i), \qquad \forall \gamma_i \in \Gamma_i$$

is also a potential for G.

Decoupled Plus Dummy Games

Example 13.1 (Wireless power control game).

Scenario

- N players want to transmit data through a wireless medium.
- players have to decide how much power to use in their transmitters.

Each player P_i must select a **power level** γ_i within a (perhaps finite) set of admissible power levels

$$\gamma_i \in \Gamma_i := [p_{\min}, p_{\max}] \subset (0, \infty)$$

However, the transmission of each P_i appears as background noise for the remaining players P_{-i}.

Decoupled Plus Dummy Games

In particular, P_i observes a Signal to Interference plus Noise Ratio (SINR) given by

$$\frac{\gamma_i}{n + \alpha \sum_{j \neq i} \gamma_j}$$

- n: constant background noise.
- $\alpha \in (0, 1)$: a gain that reflects how well the receiver can reject power from the signals that it is not trying to decode.

Goal of each P_i: minimize a cost that is the sum of two terms.

The first term is given by

$$-\beta \log \left(\frac{\gamma_i}{n + \alpha \sum_{j \neq i} \gamma_j} \right)$$

and expresses the fact that it is desirable to have a large SINR to increase the effective bit-rate.

Decoupled Plus Dummy Games

The second term expresses the cost of utilizing a large amount of power and is given by

$$C(\gamma_i)$$

for some monotone non-decreasing **cost function**.

The outcome that P_i wants to minimize is then given by

$$J_i(\gamma) = C(\gamma_i) - \beta \log(\gamma_i) + \beta \log \left(n + \alpha \sum_{j \neq i} \gamma_j \right)$$

which is of the form

$$J_i(\gamma_i, \gamma_{-i}) = H_i(\gamma_i) + Q_i(\gamma_{-i}) \qquad \forall \gamma_i \in \Gamma_i, \gamma_{-i} \in \Gamma_{-i}, \quad i \in \{1, 2, \ldots, N\}$$

Decoupled Plus Dummy Games

Therefore, from **Proposition 12.4** we have a potential game with potential

$$\phi(\gamma) := \sum_{i=1}^{N} (C(\gamma_i) - \beta \log(\gamma_i))$$

This means that any global minimizer to this potential $\phi(\gamma)$ is a Nash equilibrium (NE) to this game.

Decoupled Plus Dummy Games

Suppose cost $C(\gamma_i)$ is controlled by a Network Administrator (NA) that charges the players by their use of power.

For example, if the NA sets

$$C(s) := \beta \log(s) + s$$

then the potential $\phi(\gamma)$ becomes

$$\phi(\gamma) := \sum_{i=1}^{N} \gamma_i$$

i.e., total power that will interfere with other wireless users.

Note that the global minimum of $\phi(\gamma)$ corresponds to all players using the least amount of power.

- by playing at NE, the players achieve the social optimum.

Decoupled Plus Dummy Games

Alternatively, the NA could set

$$C(s) := \beta^* \log(s)$$

where β^* is the price for power (in dBs). This leads to the potential

$$\phi(\gamma) := (\beta^* - \beta) \sum_{i=1}^{N} \log(\gamma_i)$$

which is minimized for

- $\gamma_i = p_{\min}$ if $\beta^* < \beta$, and $\gamma_i = p_{\max}$ if $\beta^* > \beta$

Decoupled Plus Dummy Games

By adjusting the price β^* and observing the players' reaction, the NA could learn the value of the players utility parameter β.

By charging different costs, the NA can **force** the players to **selfishly** minimize a wide range of other costs.

What would be achieved by selecting

$$C(s) := \beta \log(s) - s$$

or by selecting

$$C(s) := \beta \log(s) - (s - s^*)^2$$

for a given $s^* \in \mathbb{R}$?

Bilateral Symmetric Games

Bilateral Symmetric Games

A **bilateral symmetric game** has outcomes J_i of the form

$$J_i(\gamma) = H_i(\gamma_i) + \sum_{j \neq i} W_{ij}(\gamma_i, \gamma_j), \quad \forall \gamma_i \in \Gamma_i, \quad i \in \{1, 2, \ldots, N\}$$

for functions $H_i(\gamma_i)$, $W_{ij}(\gamma_i, \gamma_j)$, $i, j \in \{1, 2, \ldots, N\}$, with

$$W_{ij}(\gamma_i, \gamma_j) = W_{ji}(\gamma_j, \gamma_i), \quad \forall \gamma_i, \gamma_j, \in \Gamma_i, \quad i, j, \in \{1, 2, \ldots, N\}$$

We can view the outcome $J_i(\gamma)$ as the sum of

- a **decoupled** term $H_i(\gamma_i)$ that only depends on the policy of P_i, and
- several terms $W_{ij}(\gamma_i, \gamma_j)$ $j \neq i$ that express **bilateral interactions** between P_i and other players P_j, $j \neq i$.

Bilateral Symmetric Games

In **bilateral symmetric games**, the interactions must be **symmetric** in the sense of $W_{ij}(\gamma_i, \gamma_j) = W_{ji}(\gamma_j, \gamma_i)$, i.e.,

The effect $W_{ij}(\gamma_i, \gamma_j)$ of player P_j on player P_i

must be the same as

The effect $W_{ji}(\gamma_j, \gamma_i)$ of player P_i on player P_j.

A bilateral symmetric game G is an exact potential game with the following potential:

$$\phi(\gamma) = \sum_{i=1}^{N} \left(H_i(\gamma_i) + \sum_{j=1}^{k-1} W_{ij}(\gamma_i, \gamma_j) \right), \quad \forall \gamma_i \in \Gamma_i$$

Congestion Games

Congestion Games

Scenario:

- N players must share R resources.
- each player must decide which resources to use.

A **policy** γ_i for P_i is a subset of the R resources that the player wants to use, i.e.,

$$\gamma_i \subset \mathbb{R} := \{1, 2, \ldots, R\}$$

The **action space** Γ_i for P_i is the collection of all such sets that can be used to accomplish a particular goal for P_i.

In a **congestion game**, the outcomes for the players depend on the total number of players that use each resource $r \in \mathbb{R}$, which can be written as

$$|\mathcal{S}_r(\gamma_1, \gamma_2, \ldots, \gamma_N)|$$

Note: $|\mathcal{A}|$ is the cardinality (# of elements) of the set $\mathcal{A}$.

Congestion Games

Therefore

$$\mathcal{S}_r(\gamma_1, \gamma_2, \ldots, \gamma_N) := \{i \in \{1, 2, \ldots, N\} : r \in \gamma_i\}$$

denotes the set of players using resource $r \in \mathbb{R}$.

The **outcome** for P_i is of the form

$$J_i(\gamma) := \sum_{r \in \gamma_i} C_r\left(|\mathcal{S}_r|\right), \quad \gamma := (\gamma_1, \gamma_2, \ldots, \gamma_N)$$

where

- the summation is over every resource $r \in \mathbb{R}$ used by the policy γ_i of P_i.
- function $C_r(n)$ provides the cost to each player of using resource r when n players share this resource.

Congestion Games

$C_r(n)$ expresses the fact that when many players use the same resources the resulting **congestion** increases the cost for everyone.

A congestion game is an exact potential game with the following potential:

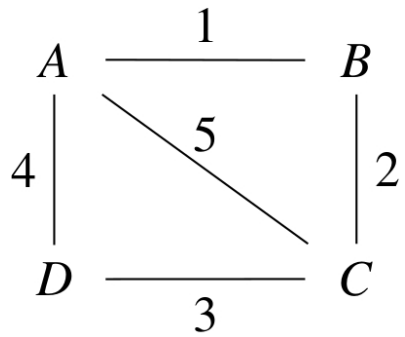
$$\phi(\gamma) := \sum_{r \in \mathbb{R}} \sum_{k=1}^{|\mathcal{S}_r(\gamma)|} C_r(k), \qquad \forall \gamma_i \in \Gamma_i$$

Therefore, the NE of this game will be the directionally-local minima of this function.

Congestion Games

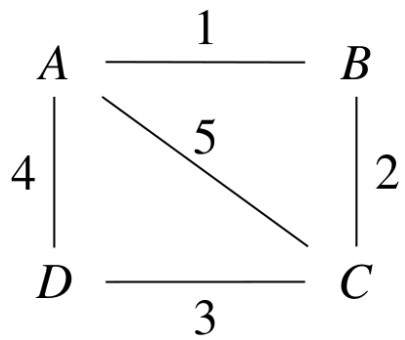
Example 13.2 (Homogeneous vehicle routing).

Consider a road network that connects the four cities A, B, C, D through five roads $1, 2, 3, 4, 5$:



Assuming that P_1 wants to route vehicles from city A to city D and player P_2 wants to route vehicles from city B to city D, their action spaces need to include all policies (i.e., sets of routes/resources) that allow them to route vehicles as desired.

Congestion Games



The action spaces for P_1 and P_2 are given, respectively, by

$$\Gamma_1 := \{\{4\}, \{5,3\}, \{1,2,3\}\} \qquad \Gamma_2 := \{\{1,4\}, \{2,3\}, \{1,5,3\}, \{2,5,4\}\}$$

Suppose that the cost $C_r(n)$ in

$$J_i(\gamma) := \sum_{r \in \gamma_i} C_r\left(|\mathcal{S}_r|\right), \qquad \gamma := (\gamma_1, \gamma_2, \ldots, \gamma_N)$$

is the time it takes to travel along road $r \in \mathbb{R} := \{1,2,3,4,5\}$ when $n \in \{1,2\}$ players use that road to route vehicles.

Congestion Games

Observation: if more vehicles use the same road they need to drive slower.

In this case, the **outcome** $J_i(\gamma)$

$$J_i(\gamma) := \sum_{r \in \gamma_i} C_r\left(|\mathcal{S}_r|\right), \qquad \gamma := (\gamma_1, \gamma_2, \ldots, \gamma_N)$$

for P_i would be the time it takes for P_i's vehicles to go from its start to its end city.

Note. For a congestion game, the time it takes to travel along a road must depend solely on how many players use that road to route vehicles and not on which specific players use it.

This carries the implicit homogeneity assumption that all players are equal.

Other Potential Games

Other Potential Games

Sudoku puzzle

Filling a 9×9 grid with digits 1 through 9 so that each row, each column, and each of the nine 3×3 sub-grids (also called blocks) contain all nine digits without repetitions.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Imagine that Sudoku is played by $N := 81$ players

- each placed at one square of the 9×9 grid.

The player at the ith square decides which digit to place in its own square.

Other Potential Games

Players associated with squares that are originally filled have action spaces with a single element

- the digit in their square

Players associated with empty squares all have the same action space consisting of all 9 digits:

$$\Gamma_i := \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Suppose that the outcome of P_i is given by

$$J_i(\gamma) := \sigma_i^{\text{row}}(\gamma) + \sigma_i^{\text{col}}(\gamma) + \sigma_i^{\text{block}}(\gamma), \qquad \gamma := \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$$

where $\sigma_i^{\text{row}}(\gamma)$, $\sigma_i^{\text{col}}(\gamma)$, and $\sigma_i^{\text{block}}(\gamma)$ denote the number of times that the digit γ_i selected by P_i appears elsewhere in P_i's row, column, and block, respectively.

Other Potential Games

The Sudoku puzzle is **solved** for every multiplayer **policy** γ that leads to a **zero outcome** for every player in

$$J_i(\gamma) := \sigma_i^{\text{row}}(\gamma) + \sigma_i^{\text{col}}(\gamma) + \sigma_i^{\text{block}}(\gamma), \qquad \gamma := \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$$

This multiplayer version of Sudoku is an exact potential game with potential

$$\phi(\gamma) := \frac{1}{2} \sum_{i=1}^{N} J_i(\gamma)$$

having a global minimum at 0 for every multiplayer policy γ that solves the Sudoku puzzle.

Attention: there may be NE that are directionally-local minima, but not global minima and thus do not solve the Sudoku puzzle.

Other Potential Games

Sudoku puzzle example illustrates how to use noncooperative game theory to construct a simple distributed multi-agent algorithm to solve a problem that starts as a **single-agent** centralized optimization.

For certain types of optimizations there are systematic procedures to construct multiplayer games whose NE correspond to global minima.

Distributed Resource Allocation

Distributed Resource Allocation

Scenario:

- N agents must share R resources
- each agent must decide which resources to use.

A **resource allocation policy** γ_i for agent A_i is a subset of the R resources that the agent wants to use, i.e.,

$$\gamma_i \subset \mathbb{R} := \{1, 2, \ldots, R\}$$

The **action space** Γ_i for agent A_i is the collection of all such sets that can be used to accomplish a particular goal for A_i.

In a **distributed resource allocation optimization**, the criteria depends on the sets of agents $\mathcal{S}_r(\gamma)$ that use each resource $r \in \mathbb{R}$, which are given by

$$\mathcal{S}_r(\gamma_1, \gamma_2, \ldots, \gamma_N) := \{i \in \{1, 2, \ldots, N\} : r \in \gamma_i\}$$

Distributed Resource Allocation

Goal: find a joint policy $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_N)$ for all the agents that minimizes a **global welfare cost** of the form

$$W(\gamma) := \sum_{r \in \mathcal{R}} W_r(\mathcal{S}_r(\gamma))$$

Function W_r is the **welfare cost for resource** r, and maps each possible set of agents using resource r to a numerical value.

Global welfare costs like $W(\gamma)$ (expressed by a sum of welfare costs for individual resources) are called separable.

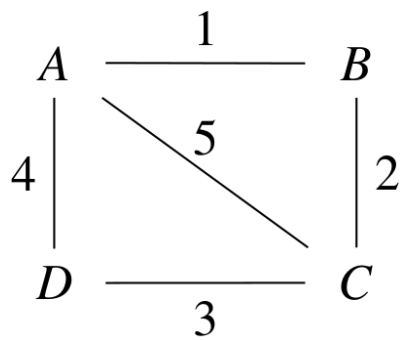
Examples: (sensor) Coverage Problems, and

Graph Coloring Problems: assign colors to the edges of a graph, while avoiding neighbors to have the same color.
We can view Sudoku as a very complex graph coloring problem.

Distributed Resource Allocation

Example 13.2 (Heterogeneous vehicle routing).

Consider a road network that connects four cities A, B, C, D through five roads $1, 2, 3, 4, 5$:



Assuming agent A_1 wants to route vehicles from city A to city D and agent A_2 wants to route vehicles from city B to city D, their action spaces are given, respectively, by

$$\Gamma_1 := \{\{4\}, \{5, 3\}, \{1, 2, 3\}\} \quad \Gamma_2 := \{\{1, 4\}, \{2, 3\}, \{1, 5, 3\}, \{2, 5, 4\}\}$$

Distributed Resource Allocation

Distributed resource allocation problem

Seek for a joint policy $\gamma = (\gamma_1, \gamma_2)$ that minimizes the total time that the vehicles spend on their trips. This can be written as

$$W(\gamma) := \sum_{r \in \mathcal{R}} W_r(\mathcal{S}_r(\gamma))$$

where

- $\mathcal{R} := \{1, 2, 3, 4, 5\}$; $\mathcal{S}_r(\gamma) \subset \{1, 2\}$: the set of agents using road $r \in \mathcal{R}$ under the policy γ
- $W_r(\mathcal{S})$: the total time that the vehicles from the agents in the set $\mathcal{S} \subset \{1, 2\}$ spend on the road $r \in \mathcal{R}$, when these agents share that road.

Distributed Resource Allocation

For **Distributed resource allocation**, the time it takes to travel along a road may depend on which agents' vehicles travel that road

This means that one can consider agents that are **heterogeneous** in the way that they **congest** the road.

Distributed Welfare Games

It is possible to construct outcomes $J_i(\gamma)$ for the agents so that if they are encouraged to behave as players in a noncooperative game defined by $J_i(\gamma)$, the minimum of the global welfare cost

$$W(\gamma) := \sum_{r \in \mathcal{R}} W_r(\mathcal{S}_r(\gamma))$$

will correspond to a NE.

Such games are called **distributed welfare games**.

There are several options for the construction of these games.

Distributed Welfare Games

1. The **Marginal Contribution** (MC)
   - or **Wonderful Life Utility** (WLU)

Corresponds to setting the outcome $J_i(\gamma)$ for each P_i equal to

$$J_i(\gamma) := \sum_{r \in \gamma_i} \left(W_r \left(\mathcal{S}_r(\gamma) \right) - W_r \left(\mathcal{S}_r(\gamma) \setminus \{i\} \right) \right)$$

Summation is taken over all resources r used by P_i's policy γ_i.

Each term accounts for the difference between the welfare cost associated with all players $\mathcal{S}_r(\gamma)$ using resource r, and the cost associated with all but the ith player using the same resource.

Distributed Welfare Games

The WLU leads to an exact potential game with potential equal to the global welfare cost

$$\phi(\gamma) := W(\gamma) = \sum_{r \in \mathcal{R}} W_r \left(\mathcal{S}_r(\gamma) \right)$$

whose minimum will be a NE.

Distributed Welfare Games

2. The **Shapley Value Utility** (SVU)

Corresponds to setting the outcome $J_i(\gamma)$ for each P_i equal to

$$J_i(\gamma) := \sum_{r \in \gamma_i} \sum_{\mathcal{S} \subset \mathcal{S}_r(\gamma) \setminus \{i\}} \frac{|\mathcal{S}|!(|\mathcal{S}_r(\gamma)| - |\mathcal{S}| - 1)!}{|\mathcal{S}_r(\gamma)|!} \left(W_r\left(\mathcal{S} \cup \{i\}\right) - W_r(\mathcal{S})\right)$$

where the summation is taken over all subsets $\mathcal{S}$ of $\mathcal{S}_r \setminus \{i\}$.

$W_r\left(\mathcal{S} \cup \{i\}\right) - W_r(\mathcal{S})$ compares the marginal cost of adding player P_i to the set of players $\mathcal{S}$ that use the resource r.

Distributed Welfare Games

The SVU leads to an exact potential game with potential

$$\phi(\gamma) := \sum_{r\in\gamma_i} \sum_{\mathcal{S}\subset\mathcal{S}_r(\gamma)\setminus\{i\}} \frac{1}{|\mathcal{S}|}\left(\sum_{r\in\mathcal{R}}\sum_{\mathcal{T}\subset\mathcal{S}}(-1)^{|\mathcal{S}|-|\mathcal{T}|}W_r(\mathcal{T})\right)$$

that typically does not match $W(\gamma)$.

In this case, there is no guarantee that the minimum of the global welfare cost is a NE.

Note 11.- **Budget balanced distributed welfare games**

The game obtained from the SVU has the advantage (over the WLU) that it is **budget balanced**.

Distributed Welfare Games

Budget balanced distributed welfare games

The outcomes of the players are of the form

$$J_i(\gamma) = \sum_{r \in \gamma_i} f_r(i, \mathcal{S}_r(\gamma))$$

with the **distribution rules** functions $f_r(\cdot)$ satisfying

$$\sum_{i \in \mathcal{S}} f_r(i, \mathcal{S}) = W_r(\mathcal{S})$$

for every: resource $r \in \mathcal{R}$, and set of players $\mathcal{S} \subset \{1, 2, \ldots, N\}$.

Budget balance is important when one needs to match the welfare cost $W_r(\mathcal{S})$ for resource r with the sum of the distribution rules for the players in $\mathcal{S}$.

Distributed Welfare Games

Scenario: when the distribution rules $f_r(i, \mathcal{S})$ is a price (or revenue) that P_i must pay (or receive) to use the resource r, while this resource is being shared by all the players in $\mathcal{S}$.

The budget balance

$$\sum_{i \in \mathcal{S}} f_r(i, \mathcal{S}) = W_r(\mathcal{S})$$

means that the total payments (or revenues) for each resource r match precisely the welfare cost (or revenue) of that resource.

Difficulty of SVU

Evaluation is computationally difficult for a large number of players since the summation over all subsets $\mathcal{S}$ of $\mathcal{S}_r(\gamma) \setminus \{i\}$ may include a large number of terms.

Price of Anarchy

For the WLU or for the SVU, there is no guarantee that noncooperative players will select policies that minimize the global welfare cost.

- WLU: this is because there may be directionally-local minima that are not the global minima.
- SVU: the global minimum may not even be a NE.

This observation motivates the following definition:

$$PoA := \frac{\max_{\gamma \in \Gamma_{\text{eq}}} W(\gamma)}{\min_{\gamma \in \Gamma} W(\gamma)}$$

where Γ_{eq} denotes the set of NE of the distributed welfare game.

Price of Anarchy

The ratio in PoA is known as the **price of anarchy**

PoA compares the value of the global welfare cost for the **worst** NE (i.e., the one corresponding to the largest cost) with the global minimum of the global welfare cost.

A large PoA means that when the players engage in the noncooperative game, they may end up at a NE that is significantly larger than the global minimum

This can be viewed as the price to pay for behaving as cooperative agents

For both the WLU and for the SVU, the PoA cannot exceed 2 under mild assumptions on the welfare costs.

Price of Anarchy

Lemma 13.1. Consider a distributed welfare game with outcomes given by $J_i(\gamma)$ for WLU or $J_i(\gamma)$ for SVU and assume that the welfare cost W_r of every resource $r \in \mathcal{R}$ is supermodular, then $PoA \leq 2$.

Note 12 (Supermodularity). A cost function W that maps subsets of $\mathcal{D}$ to real numbers is called supermodular if for every $i \in \mathcal{D}$ and subsets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{S}$, we have that

$$\mathcal{S}_1 \subset \mathcal{S}_2 \Rightarrow W(\mathcal{S}_1 \cup \{i\}) - W(\mathcal{S}_1) \leq W(\mathcal{S}_2 \cup \{i\}) - W(\mathcal{S}_2)$$

Supermodularity means that, as a set enlarges, the marginal cost associated with adding an element to the set increases

- or at least does not decrease

Computation of NE for Potential Games

Computation of Nash Equilibria for Potential Games

Attractive features of potential games

- they typically have at least one NE
- **it is straightforward to construct algorithms to compute such equilibria**

Algorithm for any potential game with finite action spaces.

Consider a game with N players $P_1, P_2, \ldots, P_N$, which select policies from the action spaces $\Gamma_1, \Gamma_2, \ldots, \Gamma_N$, respectively.

When each P_i uses a policy $\gamma_i \in \Gamma_i$, we denote by

$$J_i(\gamma), \qquad \gamma := (\gamma_1, \gamma_2, \ldots, \gamma_N) \in \Gamma := \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_N$$

the **outcome of the game** for the P_i, and all players want to minimize their own outcomes.

Computation of Nash Equilibria for Potential Games

A **(pure) path** is a sequence of multiplayer policies

$$\{\gamma^1, \gamma^2, \ldots\}, \qquad \gamma^k = (\gamma_1^k, \gamma_2^k, \ldots, \gamma_N^k) \in \Gamma := \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_N$$

where each $\gamma^{k+1} \in \Gamma$ differs from the previous $\gamma^k \in \Gamma$ by change in the policy of a single player P_i.

For every k, there exists a player $P_i, i \in \{1, 2, \ldots, N\}$ for which

$$\gamma^{k+1} = (\gamma_i, \gamma_{-i}^k), \quad \gamma_i \neq \gamma_i^k$$

Computation of Nash Equilibria for Potential Games

A path is said to be an **improvement path** with respect to a game with outcomes $J_i(\gamma)$ if, for every k

$$\gamma^{k+1} = (\gamma_i, \gamma^k_{-i}), \quad \gamma_i \neq \gamma^k_i \quad \Rightarrow \quad J_i(\gamma^{k+1}) < J_i(\gamma^k)$$

In words: if γ^{k+1} differs from γ^k due to a change in policy of player P_i, then this change in policy must result in a strict improvement of P_i's outcome.

Improvement paths can only terminate when no player can improve their outcome by changing policy.

A 4-step algorithm for generating improvement paths is shown next.

Computation of Nash Equilibria for Potential Games

Step 1: Pick an arbitrary initial multiplayer policy γ^1 and set $k = 1$.

Step 2: Find a player P_i for which there exists a policy $\gamma_i \in \Gamma_i$ such that

$$J_i(\gamma_i, \gamma_{-i}^k) < J_i(\gamma^k)$$

Step 3: If such a player does not exist, terminate the path.

Step 4: Otherwise, set

$$\gamma^{k+1} = (\gamma_i^{k+1}, \gamma_{-i}^k), \qquad \gamma_i^{k+1} \in \arg\min_{\gamma_i \in \Gamma_i} J_i(\gamma_i, \gamma_{-i}^k)$$

increase k by 1 and repeat from **Step 2**.

The policy γ_i^{k+1} in **step 4** can be viewed as P_i's best response against γ_{-i}^k.

Computation of Nash Equilibria for Potential Games

When the minimum is achieved at multiple policies γ_i, one can pick γ_i^{k+1} to be any of them.

For finite exact or ordinal potential games, improvement paths always terminate and lead to a NE.

Computation of Nash Equilibria for Potential Games

Proposition 13.1 (Finite improvement property).

Every finite improvement path terminates at a NE. Also, every improvement path of an exact or an ordinal potential game with finite action spaces is finite. Then it terminates at a NE.

Proof of Proposition 13.1.

A consequence of the fact that improvement paths are only allowed to terminate when no player can unilaterally improve their own outcome, i.e., if an improvement path terminates at a step k, then we must have that

$$J_i(\gamma_i^k, \gamma_{-i}^k) \leq J_i(\gamma_i, \gamma_{-i}^k) \qquad \forall \gamma_i \in \Gamma_i, \quad i \in \{1, 2, \ldots, N\}$$

since otherwise we could continue the path. This shows that $J_i(\gamma_i^k, \gamma_{-i}^k)$ is a NE.

Computation of Nash Equilibria for Potential Games

To show that every improvement path must terminate, note that for every γ^k along an improvement path, there exists a P_i such that

$$J_i(\gamma^{k+1}) < J_i(\gamma^k)$$

and therefore, either for an exact or an ordinal potential game with potential ϕ, we must have

$$\phi(\gamma^{k+1}) < \phi(\gamma^k)$$

i.e., the potential must be strictly decreasing along any improvement path. When the action spaces are finite, the potential function can only take a finite number of values and therefore can only decrease strictly a finite number of times. This means a finite improvement path must always be finite.

Computation of Nash Equilibria for Potential Games

MATLAB®Hint 4 (Improvement path).

Enumerate the actions of all P_is so that every action space can be represented by a numerical array. Construct the following

`actionSpaces` : cell array with N entries, each containing the action space for one P_i. `actionSpaces{i}`, $i \in \{1, 2, \ldots, N\}$ is a numerical array with all the actions available to P_i.

`A0` : matrix with N entries containing the actions of the P_is at the start of the path. `A0(i)`, $i \in \{1, 2, \ldots, N\}$ is the action of P_i at the start of the path.

`funJ` : MATLAB® function. Returns an N-element matrix with the P_is' costs for a given matrix of P_is' actions.
Given an N-element matrix `A` with the actions of all the P_i's, the costs' matrix is obtained using `funJ(A)`.

Computation of Nash Equilibria for Potential Games

MATLAB® function that returns an improvement path.

Randomization is used to make sure that, if it is called multiple times, it will likely result in distinct improvement paths and potentially distinct NE.

MATLAB® function returns two variables:

`A` : matrix with N entries with the players' actions at the end of the improvement path.
In view of **Proposition 13.1**, this must correspond to a NE.

`J`: matrix with N entries with the players' costs at the end of the improvement path.

Computation of Nash Equilibria for Potential Games

```
function [A,J] = improvementPath(actionSpaces,A0)

 N = prod(size(actionSpaces)); % number of players
 J = funJ(A0); % compute initial costs for every player
 A = A0;
 improved = true;

 while improved
  improved = false;

  for i = randperm(N) % range over players, random order
   actionSpace = setdiff(actionSpaces{i},A(i));
   % range over alternative actions for player i, random order

   for newAi = actionSpace(randperm(length(actionSpace)))
    oldAi = A(i); % save previous action
    A(i) = newAi; % try alternative action
    newJ = funJ(A);
```

Computation of Nash Equilibria for Potential Games

```
      if newJ(i) < J(i)
         % new action improves outcome for pi, keep it
         J = newJ;
         improved = true;
      else
         A(i) = oldAi; % new action does not improve, discard it
      end

    end % for new Ai = ...
    if improved
     break; % found improvement:  try all players again
    end

  end % for i = ...

 end % while improved

end
```

Fictitious Play

Fictitious Play

Game with N players $P_1, P_2, \ldots, P_N$ that select probability distributions from mixed action spaces $\mathcal{Y}^1, \mathcal{Y}^2, \ldots, \mathcal{Y}^N$.

When each P_i uses a mixed policy $y_i \in \mathcal{Y}^i$, we denote by

$$J_i(y), \qquad y := (y_1, y_2, \ldots, y_N) \in \Gamma := \mathcal{Y}^1 \times \mathcal{Y}^2 \times \ldots \times \mathcal{Y}^N$$

the **outcome of the game** for P_i, and all players want to minimize their own outcomes.

A **mixed path** is a sequence of multiplayer policies

$$\{y^1, y^2, \ldots\}, \qquad y^k = (y_1^k, y_2^k, \ldots, y_N^k) \in \mathcal{Y} := \mathcal{Y}^1 \times \mathcal{Y}^2 \times \cdots \times \mathcal{Y}^N$$

that the players use over consecutive repetitions of the game.

Fictitious Play

Players construct **beliefs** regarding the mixed policies used by the remaining P_is as the path $\{y^1, y^2, \ldots, y^N\}$ progresses. To accomplish this, they keep track of the empirical distributions of the mixed policies used by the other P_is up to the current time.

The **belief** at time k regarding the mixed policy used by P_i is the time-average of the mixed policies used by P_i up to time k:

$$\hat{y}_i^k := \frac{1}{k} \sum_{\ell=1}^{k} y_i^\ell, \qquad \forall k \geq 1$$

Players assume that **beliefs** are correct, i.e., remaining players will use a mixed policy that is exactly equal to these beliefs. Then, every player selects a mixed policy that is a **best response** to these beliefs.

Fictitious Play

Specifically, P_i selects a mixed policy y_i^{k+1} for the next time step that satisfies

$$J_i(y_i^{k+1}, \hat{y}_{-i}^k) \leq J_i(y_i, \hat{y}_{-i}^k), \qquad \forall y_i \in \mathcal{Y}^i$$

or equivalently, y_i^{k+1} is the best response against $\hat{y}_{-i}^k$:

$$y_i^{k+1} \in \arg\min_{y_i \in \mathcal{Y}^i} J_i(y_i, \hat{y}_{-i}^k)$$

Assumption that the beliefs precisely match the mixed policies that the other P_i's will use is **not correct**

- because all P_is keep adjusting their own mixed policies.

Nevertheless, for potential games, fictitious play leads to beliefs $\hat{y}^k$ that converge to NE.

Fictitious Play

Theorem 13.1 (Fictitious play for potential games).

For every (exact) potential game with mixed policies, the fictitious play belief sequence converges to the set of mixed NE for the game, in the sense that each limit point of the belief sequence $\hat{y}^k$

$$\{y^1, y^2, \ldots\}, \qquad \hat{y}^k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_N^k) \in \mathcal{Y} := \mathcal{Y}^1 \times \mathcal{Y}^2 \times \cdots \times \mathcal{Y}^N$$

is a NE.

The use of fictitious play goes much beyond potential games, as one can conclude from the following two results:

Fictitious Play

Theorem 13.2 (Fictitious play for zero-sum games).

For every finite ZS game, the fictitious play belief sequence

$$\{y^1, y^2, \ldots\}, \qquad \hat{y}^k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_N^k) \in \mathcal{Y} := \mathcal{Y}^1 \times \mathcal{Y}^2 \times \cdots \times \mathcal{Y}^N$$

always converges to a saddle-point equilibrium.

Theorem 13.3 (Fictitious play for general games). For any finite game, if the belief sequence

$$\{y^1, y^2, \ldots\}, \qquad \hat{y}^k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_N^k) \in \mathcal{Y} := \mathcal{Y}^1 \times \mathcal{Y}^2 \times \cdots \times \mathcal{Y}^N$$

converges, that limit of the belief sequence is a Nash equilibrium for the game.

Fictitious Play

Fictitious play converges for many games, but it does not converge for every game, including the bimatrix game

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

for which fictitious play does not converge, e.g., starting from the following initial beliefs

$$\hat{y}_1^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \hat{y}_2^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Fictitious Play

MATLAB®Hint 5 (Fictitious play).

Construct an $(N+1)$-dimensional tensor `A` that defines the pure game outcomes.

Entry `A(i1,i2,...,iN,i)` contains the outcome for P_i, when

Player P_1 selects the pure policy `i1`,

Player P_2 selects the pure policy `i2`,

...

Player P_N selects the pure policy `iN`.

Fictitious Play

The following MATLAB® function plays `K` rounds of fictitious play for the game described by the tensor `A`

Function returns an N-dimensional cell array `belief` with the players' beliefs at the end of the path.

The entry `belief{i}` is the belief for player P_i, which is a probability distribution over the probability simplex of dimension equal to `size(A,i)`.

It also returns a vector with the value of the game for the beliefs at the end of the path.

Fictitious Play

```
function [belief,value] = fictitiousPlay(A,K)

    N = length(size(A))-1;

    %% compute random initialization for the mixed path
    belief = cell(N,1);

    for i = 1:N
       if 1
          belief{i}= rand(size(A,i),1); % random belief;
       else
          belief{i} = ones(size(A,i),1); % uniform belief;
       end
       belief{i} = belief{i}/sum(belief{i});
       % normalize to get distribution
    end
```

Fictitious Play

```
%% iterate fictitious play
for k = 1:K
   %% compute probability - weighted outcomes
   Ay = A;

   for i = 1:N
      reps = size(A);
      reps(i) = 1;
      shape = ones(1,N+1);
      shape(i) = length(belief{i});
      yi = reshape(belief{i}, shape);
      yi = repmat(yi,reps);
      % for player i's outcomes do not multiply by its own
      % belief to eventually compute the best response
      str =['yi(',repmat(':,',1,N),num2str(i),')=1;'];
      eval(str);
      Ay = Ay.*yi;
   end % for i = 1:N
```

Fictitious Play

```
%% compute best response
y = cell(N,1);

for i = 1:N
   % average outcomes over actions of other players
   str = ['Ay(',repmat(':,',1,N),num2str(i),')'];
   S = eval(str);

   for j =1:N
      if j ~= i
         S = sum(S,j);
      end
   end

   % compute best responses
   [~,j] = min(S,[],i); % pure best response
   y{i} = zeros(size(S,i),1);
   y{i}(j)=1; % mixed best response
end % for i = 1:N
```

Fictitious Play

```
    %% update belief
    for i = 1:N
       belief{i} = (k*belief{i} + y{i})/(k+1);
    end % for i = 1:N
end % for k = 1:K
%% compute final value
value = nan(N,1);
% compute probability - weighted outcomes
Ay = A;
for i = 1:N
    reps = size(A);
    reps(i) = 1;
    shape = ones(1,N +1);
    shape(i) = length(belief{i});
    yi = reshape(belief{i},shape);
    yi = repmat(yi,reps);
    Ay = Ay.*yi;
end % for i = 1:N
```

Fictitious Play

```
    for i = 1:N
        % average outcomes over actions of all players
        str = ['Ay(',repmat(':,',1,N),num2str(i),')'];
        S = eval(str);
        for j = 1:N
           S = sum(S,j);
        end
        value(i) = S;
     end % for i = 1:N
end
```

Practice Exercises

Practice Exercises

13.1. Verify that

$$\phi(\gamma) = \sum_{i=1}^{N} H_i(\gamma_i), \qquad \forall \gamma_i \in \Gamma_i$$

is an exact potential for the game with outcomes given by

$$H_i(\gamma_i, \gamma_{-i}) = H_i(\gamma_i, \bar{\gamma}_{-i}) = H_i(\gamma_i),$$
$$\forall \gamma_i \in \Gamma_i, \gamma_{-i}, \bar{\gamma}_{-i} \in \Gamma_{-i}, i \in \{1, 2, \ldots, N\}$$

Solution to Exercise 13.1.
For every $\gamma_i, \bar{\gamma}_i \in \Gamma_i, \gamma_{-i} \in \Gamma_{-i}$, and $i \in \{1, 2, \ldots, N\}$, we have that

$$J_i(\gamma_i, \gamma_{-i}) - J_i(\bar{\gamma}_i, \gamma_{-i}) = H_i(\gamma_i) - H_i(\bar{\gamma}_i) = \phi(\gamma_i, \gamma_{-i}) - \phi(\bar{\gamma}_i, \gamma_{-i})$$

which confirms ϕ is an exact potential function for the game.

Practice Exercises

13.2 (Sudoku). Consider the multi-player version of Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

with the outcome of player P_i given by

$$J_i(\gamma) := \sigma_i^{\text{row}}(\gamma) + \sigma_i^{\text{col}}(\gamma) + \sigma_i^{\text{block}}(\gamma) \qquad \gamma := \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$$

where $\sigma_i^{\text{row}}(\gamma)$, $\sigma_i^{\text{col}}(\gamma)$, $\sigma_i^{\text{block}}(\gamma)$: number of times that digit γ_i selected by P_i appears elsewhere in its row, column, block.

Show that this defines an exact potential game with potential

$$\phi(\gamma) := \frac{1}{2} \sum_{j=1}^{N} J_j(\gamma)$$

Practice Exercises

Solution to Exercise 13.2.

We can view this game as the sum of three games:

- rows game: outcomes given by $\sigma_i^{\text{row}}(\gamma)$
- columns game: outcomes given by $\sigma_i^{\text{col}}(\gamma)$
- blocks game: outcomes given by $\sigma_i^{\text{block}}(\gamma)$

We will show that each of these games is an exact potential game with potentials given by

$$\phi^{\text{row}}(\gamma) := \frac{1}{2}\sum_{j=1}^{N}\sigma_j^{\text{row}}(\sigma), \quad \phi^{\text{col}}(\gamma) := \frac{1}{2}\sum_{j=1}^{N}\sigma_j^{\text{col}}(\sigma), \quad \phi^{\text{block}}(\gamma) := \frac{1}{2}\sum_{j=1}^{N}\sigma_j^{\text{block}}(\sigma)$$

Then, it follows from **Proposition 12.4** that

$$\phi(\gamma) := \phi^{\text{row}}(\gamma) + \phi^{\text{col}}(\gamma) + \phi^{\text{block}}(\gamma) = \frac{1}{2}\sum_{j=1}^{N} J_j(\gamma)$$

is indeed a potential for the sum game.

Practice Exercises

Show the rows game is an exact potential game with potential ϕ^{row}. Pick $\gamma_i \neq \bar{\gamma}_i \in \Gamma_i$, $\gamma_{-i} \in \Gamma_{-i}$, $i \in \{1, 2, \ldots, N\}$, compute

$$\phi^{\text{row}}(\gamma_i, \gamma_{-i}) - \phi^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}) = \frac{1}{2}\sum_{j=1}^{N}(\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}))$$

$$= \frac{1}{2}(\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i})) + \frac{1}{2}\sum_{j\neq 1}(\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}))$$

Terms in RHS sum for which $\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}) \neq 0$, must correspond to players P_j, $j \neq i$ in the same row as P_i, since for all others the value of σ_j^{row} is not affected by a change in the digit selected by P_i. Among those players P_j, $j \neq i$, we have

$$\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}) = \begin{cases} +1 & P_j \text{ selected a digit } \gamma_j = \gamma_i \neq \bar{\gamma}_i \\ -1 & P_j \text{ selected a digit } \gamma_j = \bar{\gamma}_i \neq \gamma_i \\ 0 & P_j \text{ neither selected } \gamma_j \notin \{\gamma_i, \bar{\gamma}_i\} \end{cases}$$

Practice Exercises

Therefore the summation

$$\sum_{j \neq 1} (\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}))$$

is simply equal to the number of players P_j, $j \neq i$ in the same row as P_i that selected γ_i minus the number of players that selected $\bar{\gamma}_i$, which means that

$$\sum_{j \neq 1} (\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i})) = \sigma_i^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_i^{\text{row}}(\bar{\gamma}_i, \gamma_{-i})$$

This shows that the two terms in the right-hand side of

$$\phi^{\text{row}}(\gamma_i, \gamma_{-i}) - \phi^{\text{row}}(\bar{\gamma}_i, \gamma_{-i})$$
$$= \frac{1}{2}(\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i})) + \frac{1}{2} \sum_{j \neq 1} (\sigma_j^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_j^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}))$$

are equal to each other.

Practice Exercises

Therefore

$$\phi^{\text{row}}(\gamma_i, \gamma_{-i}) - \phi^{\text{row}}(\bar{\gamma}_i, \gamma_{-i}) = \sigma_i^{\text{row}}(\gamma_i, \gamma_{-i}) - \sigma_i^{\text{row}}(\bar{\gamma}_i, \gamma_{-i})$$

which establishes that ϕ^{row} is an exact potential for the rows game. The same argument can be used to conclude that ϕ^{col} and ϕ^{block} are exact potentials for the column and block games, respectively, by looking at columns and blocks instead of rows.

Practice Exercises

13.5 (Sudoku). Write a MATLAB® script to solve the puzzle

5	3			7				
6			1	9	5			
	9	8					6	
8		9		6				3
4			8	5	3			1
7			9	2			5	6
	6					2	8	
			4	1	9		3	5
				8			7	9

by computing improvement paths for the multi-player version of the Sudoku game considered in **Section 13.5.**

Hint: Make use of the code in MATLAB® Hint 4.

Practice Exercises

Solution. To use the code in Hint 4, construct a 9×9 cell array with the action spaces for all the P_i's. Encode the Sudoku board layout into a 9×9 matrix with all the digits in the layout and zeros where the layout has an empty slot.

The matrix is:

```
board = [5 3 0   0 7 0   0 0 0 ;
         6 0 0   1 9 5   0 0 0 ;
         0 9 8   0 0 0   0 6 0 ;

         8 0 9   0 6 0   0 0 3 ;
         4 0 0   8 5 3   0 0 1 ;
         7 0 0   9 2 0   0 5 6 ;

         0 6 0   0 0 0   2 8 0 ;
         0 0 0   4 1 9   0 3 5 ;
         0 0 0   0 8 0   0 7 9 ] ;
```

Practice Exercises

We will need a MATLAB® function to construct the cell array with the action spaces:

- if a digit appears in the board matrix, the player has a single action which is precisely that digit; and
- if a 0 appears in the board matrix, the player can play any digit that does not appear in its row, column, and block.

Practice Exercises

```
function actionSpaces = sudoku_actionspaces(board);
    actionSpaces = cell(size(board));
    for row = 1:9
        for col = 1:9
            if board(row,col) ~= 0
                actionSpaces{row,col} = board(row,col);
            else
                actionSpaces{row,col} = 1:9;
                % remove digits in same row
                actionSpaces{row,col} = setdiff(actionSpaces{row,col},board(row,:));
                % remove digits in same column
                actionSpaces{row,col} = setdiff(actionSpaces{row,col},board(:,col));
                % remove digits in same block
                rowBlk = 3*floor((row-1)/3)+(1:3);
                colBlk = 3*floor((col-1)/3)+(1:3);
                actionSpaces{row,col} = setdiff(actionSpaces{row,col},board(rowBlk,colBlk));
            end
        end
    end
end
```

Practice Exercises

We also need a function `funJ` that, given a 9×9 matrix `A` with the players' actions, returns a 9×9 matrix with the corresponding players' costs.

```
function function J = funJ(A)
   J = zeros(size(A));
   for row = 1:9
      for col = 1:9
         rowBlk =3*floor((row-1)/3)+(1:3);
         colBlk =3*floor((col-1)/3)+(1:3);
         J(row,col) = sum(A(row,:)  == A(row,col))+...
         sum(A(:,col) == A(row,col))+...
         sum(sum(A(rowBlk,colBlk) == A(row,col)))-3;
      end
   end
end
```

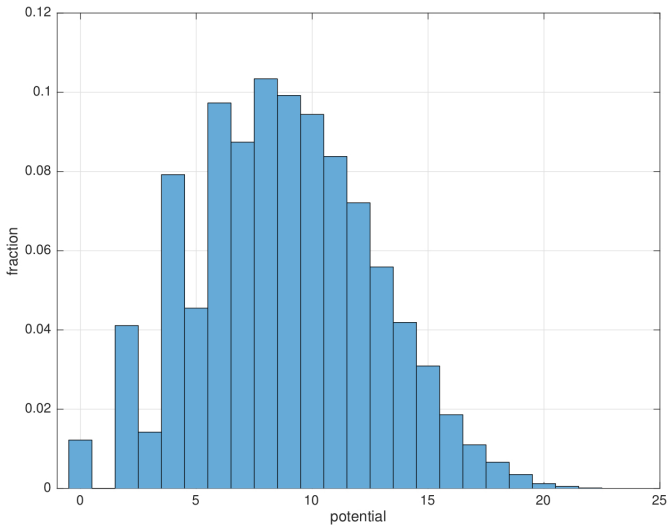
Practice Exercises

Finally, the following MATLAB® code calls the function `improvementPath` defined in MATLAB® Hint 4 multiple times to try to obtain a NE that is a global minimum of the potential.

```
actionSpaces = sudoku_actionspaces(board);
nRepeats = 1000;
for k = 1:nRepeats
   % compute random initial actions
   A0 = nan(size(actionSpaces));
   for i = 1:N
      A0(i) = actionSpaces{i}(randi(length(actionSpaces{i})));
   end
   [A,J] = improvementPath(actionSpaces,A0);
   if sum(J(:))  == 0
      fprintf('Solution found:\n');
      disp(A);
      break;
   end
```

Practice Exercises

Histogram of the values of the potential at the NE obtained by the code, by computing 10,000 improvement paths.



Number of times that we obtain a zero potential - the global minimum - is relatively small (about 1.22%).

This means that generally we must compute many improvement paths until we find a solution to the Sudoku puzzle.

Practice Exercises

13.6 (Fictitious play). Use the fictitious play algorithm in MATLAB® Hint 5 to compute NE policies for the games:

1. Prisoners' dilemma with

$$A = \underbrace{\left[\begin{array}{cc} 2 & 30 \\ 0 & 8 \end{array}\right]}_{P_2 \text{ choices}} \Big\} P_1 \text{ choices} \qquad B = \underbrace{\left[\begin{array}{cc} 2 & 0 \\ 30 & 8 \end{array}\right]}_{P_2 \text{ choices}} \Big\} P_1 \text{ choices}$$

2. Battle of the sexes with

$$A = \underbrace{\left[\begin{array}{cc} -2 & 0 \\ 3 & -1 \end{array}\right]}_{P_2 \text{ choices}} \Big\} P_1 \text{ choices} \qquad B = \underbrace{\left[\begin{array}{cc} -1 & 3 \\ 0 & -2 \end{array}\right]}_{P_2 \text{ choices}} \Big\} P_1 \text{ choices}$$

Practice Exercises

3. Rock-paper-scissors with

$$A = \underbrace{\left[\begin{array}{ccc} 0 & +1 & -1 \\ -1 & 0 & +1 \\ +1 & -1 & 0 \end{array}\right]}_{P_2 \text{ choices}} \Big\} P_1 \text{ choices} \qquad B = \underbrace{\left[\begin{array}{ccc} 0 & -1 & +1 \\ +1 & 0 & -1 \\ -1 & +1 & 0 \end{array}\right]}_{P_2 \text{ choices}} \Big\} P_1 \text{ choices}$$

Practice Exercises

Solution to Exercise 13.6.

1. For the prisoners' dilemma, we use the code

```
A = zeros(2,2,2);
A(:,:,1) = [2,30;0,8];
A(:,:,2) = [2,0;30,8];
K = 10000;
[belief,value ]= fictitiousPlay(A,K);
```

which leads to

```
belief{1} = [0.0000;1.0000];
belief{2} = [0.0001;0.9999];
value = [8.0005;8.0009];
```

Practice Exercises

2. For the battle of the sexes, we use the code

```
A = zeros(2,2,2);
A(:,:,1) = [-2,1;0,-1];
A(:,:,2) = [-1,3;2,-2];
K = 10000;
[belief,value ]= fictitiousPlay(A,K);
```

which leads to

```
belief{1} = [1.0000;0.0000];
belief{2} = [1.0001;0.0000];
value = [-1.9999;-0.9999];
```

Practice Exercises

3. For the rock paper scissors, we use the code

```
A = zeros(3,3,2);
A(:,:,1) = [0,+1,-1;-1,0,+1;+1,-1,0];
A(:,:,2) = -A(:,:,1);
K = 10000;
[belief,value ]= fictitiousPlay(A,K);
```

which leads to

```
belief{1} = [0.3365;0.3353;0.3282];
belief{2} = [0.3317;0.3293;0.3390];
value = [-2.892e-05;2.892e-05];
```

End of Lecture

13 - Classes of Potential Games

Questions?