Population Dynamics Toolbox

Carlos Barreto

September 21, 2014

Abstract

This is set of tools to implement evolutionary dynamics with multiple populations. We consider both small and large populations. For finite populations, we implement some revision protocols to model random interactions between agents. On the other hand, the evolution of a society with large populations is approximated by means of dynamical equations.

The toolbox is designed to facilitate the implementation of any game, as well as the implementation of different evolutionary dynamics and/or revision protocols. In particular, our attempt is to make an efficient implementation of the algorithms to compute the dynamical evolution of the society. Also, the toolbox counts with some functions to plot the state of the system and the evolution of each strategy.

Contents

1 Introduction

Let us define a society of $\mathcal{P} = \{1, \ldots, P\}$ composed by $P \geq 1$ populations. Each population consist of a large number of agents, which conform a mass $m^p > 0$, with $p \in \mathcal{P}$. Let $S^p = \{1, \ldots, n^p\}$ be the set of actions (or pure strategies) available for each agent of the p^{th} population. Each agent selects a pure strategy and the resulting state of the population is the usage proportion of each strategy. The set of population states is defined as $X^p = \{x^p \in \mathbb{R}_+^{n^p} : \sum_{i \in S^p} x_i^p = m^p\}$, where the i^{th} component of the state, denoted by $x_i^p \in \mathbb{R}_+$, is the mass of players that select the i^{th} strategy of the population p.

Population games (or large games) capture some properties of the interactions of many economic agents:

1. large number of agents.
2. Continuity: The actions of an agent has small impact on the payoff of other agents.
3. Anonymity: means that the utility of each agent only depends on the aggregated actions of the other agents.

In particular, an economic agent decides whether to modify or not its strategy according to the available information. In this respect, we assume that the agent's behavior satisfies both inertia and myopia properties. On the one hand, inertia is the tendency to remain at the status-quo, unless there exist motives to do that. Also, this implies that the strategy adjustment events are rare events. On the other hand, myopia means that the information used to make decisions is limited, e.g., each user makes decisions based on the current state of the population and do not estimate future actions. These two properties are based on the population games theoretical framework [8] and behavioral economics [4].

To accomplish the inertia property, the time between two successive updates of one agent's strategy is modeled with an exponential distribution (this distribution is used to model the occurrence of rare events). Thus, strategy actualization events could be characterized by means of stochastic alarm clocks. Particularly, a rate R_i Poisson alarm clock produces time among rings described by a rate R_i exponential distribution. The whole actualization events in the population can be considered as a rate $R = \sum_{j \in \mathcal{V}} R_j$ Poisson alarm clock. Therefore, the average number of events in a given time interval is R and the probability of selecting the i^{th} agent in a given time instant is $\frac{R_i}{R}$ [8].

At each update opportunity (revision opportunity), the i^{th} agent might compare the average profit of its strategy with the average profit of other strategies. Particularly, an agent might change its strategy with rate ρ_{ij}.

The rate of change ρ_{ij} is determined by a revision protocol, which defines the procedure used by each user to decide whether to change or not its strategy. The scalar $\rho_{ij}(\pi^p, x^p)$ is the *conditional switch rate* from strategy i to strategy j in function of a given payoff vector π and a population state x^p.

Using the law of large numbers we can approximate the evolution of the society's state to a dynamical equation defined by

$$\dot{x}_i^p = \sum_{j \in S^p} x_j^p \rho_{ij}(\pi^p, x^p) - x_i^p \sum_{j \in S^p} \rho(\pi^p, x^p).$$

The previous equation is known as the *mean dynamic*, which is used to define some of the dynamics in the next section.

2 Revision Protocols and Evolutionary Dynamics

In this section we introduce four revision protocols, that lead to the evolutionary dynamics *logit dynamics* (Logit), *replicator dynamics* (RD), *Brown-von Neumann-Nash dynamics* (BNN), and *Smith dynamics* (Smith). These dynamics belong to the families of *perturbed optimization*, *imitative dynamics*, *excess payoff dynamics*, and *pairwise comparison dynamics*, respectively [5, 8].

2.1 Pairwise Proportional Imitation (Replicator Dynamics)

The i^{th} agent observes an opponent j at random. Then it might change its strategy if its opponent has a greater fitness. The rate change is

$$\rho_{ij}^p(\pi^p, x^p) = \frac{1}{m^p}[\pi_j^p - \pi_i^p]_+$$

This protocol lead to the replicator dynamics defined as

$$\dot{x}_i^p = x_i^p \, \hat{F}_i^p \, (x) \, ,$$

where $\hat{F}_i^p$ is the excess payoff to strategy $i \in S^p$, which is defined as

$$\hat{F}_i^p(x) = F_i^p(x) - \bar{F}^p(x),$$

and $\bar{F}^p(x)$ is the average payoff the population p, i.e.,

$$\bar{F}^p(x) = \frac{1}{m^p} \sum_{j \in S^p} x_j^p F_j^p(x).$$

2.2 Comparison to the Average Payoff (Brown-von Neumann-Nash Dynamics (BNN))

The i^{th} agent selects a strategy at random and might switch to it if that strategy has a payoff above the average. The agent switch strategy with probability proportional to the excess payoff

$$\rho_{ij}^p(\pi^p, x^p) = \left[\pi_j^p - \frac{1}{m^p} \sum_{k \in S^p} x_k^p \pi_k^p \right]_+$$

This protocol lead to BNN dynamics, defined as

$$\dot{x}_i^p = \left[\hat{F}_i^p \, (\boldsymbol{x}) \right]_+ - x_i^p \sum_{j \in S^p} \left[\hat{F}_j^p \, (\boldsymbol{x}) \right]_+ .$$

2.3 Pairwise Comparison (Smith Dynamics)

The i^{th} agent selects a strategy at random. If the opponent has a higher fitness, the the agent switch strategy with probability proportional to

$$\rho_{ij}(\pi, x) = [\pi_j - \pi_i]_+$$

This protocol leads to Smith dynamics that are defined as

$$\dot{x}_i^p = \sum_{\gamma \in S^p} x_\gamma^p \left[F_i^p(\boldsymbol{x}) - F_\gamma^p(\boldsymbol{x})\right]_+ - x_i^p \sum_{\gamma \in S^p} \left[F_\gamma^p(\boldsymbol{x}) - F_i^p(\boldsymbol{x})\right]_+. \quad (1)$$

2.4 Logit Choice

The i^{th} agent selects a strategy at random and change its strategy with a probability proportional to

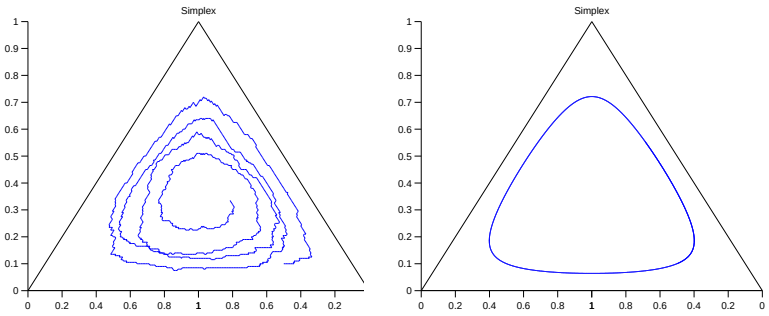
$$\rho_{ij}(\pi) = \frac{\exp(\pi_j \eta^{-1})}{\sum_{k \in S} \exp(\pi_k \eta^{-1})}$$

This protocol belong to target dynamics and with a large population results in the following dynamics

$$\dot{x}_i^p = \frac{\exp\left(\eta^{-1} F_i^p(\boldsymbol{x})\right)}{\sum_{\gamma \in S^p} \exp\left(\eta^{-1} F_\gamma^p(\boldsymbol{x})\right)}, \ \eta > 0,$$

known ad Logit dynamics.

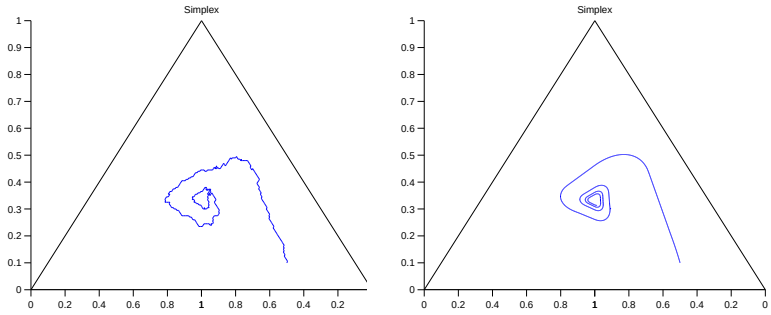
2.5 Examples



(a) Small population. (b) Large popultion.

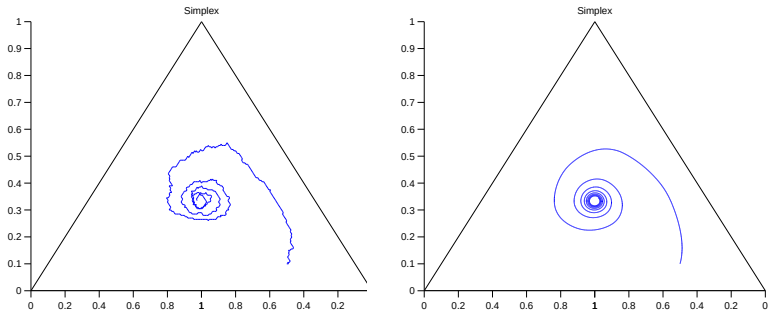
Figure 1: Rock-paper-scissors game with a) proportional imitation revision protocol and b) replicator dynamics.

In this section, we implement rock-paper-scissors game with both revision protocols and evolutionary dynamics presented above, to observe the behavioral differences between a society with small number of agents and its approximation to a dynamical system. The game has only one population with three strategies,



(a) Small population. (b) Large popultion.

Figure 2: Rock-paper-scissors game with a) comparison to average revision protocol and b) BNN dynamics.



(a) Small population. (b) Large popultion.

Figure 3: Rock-paper-scissors game with a) pairwise comparison revision protocol and b) Smith dynamics.

denoted $x = [x_1, x_2, x_3]^\top$. The fitness function is defined as $F(x) = Ax$, where A is equal to

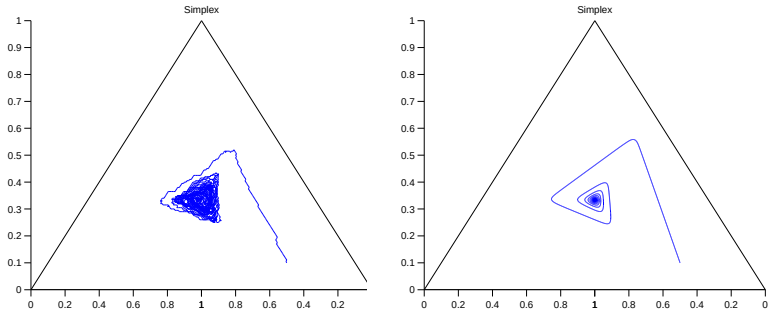
$$A = \begin{pmatrix} 2 & 1 & 3 \\ 3 & 2 & 1 \\ 1 & 3 & 2 \end{pmatrix}$$

Note that we modify the payoff matrix proposed in the literature to ensure positive payoffs. Fig. 1 to 4 show the evolution of the society with each revision protocol and its approximation to differential equations. We set the initial condition $x_0 = [0.2, 0.7, 0.1]^\top$. The small population cases are made with 200 agents and 10000 iterations. The dynamical cases are run during 30 time units.

3 Implementation

3.1 Parameters of the Implementation

The toolbox uses a structure that contains all the parameters required to run the simulations. The parameters of a population game are defined in Table 1. The following is an example to define a game with one population and three strategies per population:



(a) Small population. (b) Large popultion.

Figure 4: Rock-paper-scissors game with a) logit choice revision protocol and b) Logit dynamics with $\eta = 0.02$.

```
G = struct('n', 3, 'f', @fitness1, 'dynamics', {rd} , '
    ode', 'ode113', 'x0', [0.2 .7 0.1 ]', 'time', 60);
```

n defines the number of strategies per population, f is a function handler that calculates the fitness of the strategies in each population, and *dynamics* defines the name of the evolutionary dynamics that we want to use. The simulations are run using the ordinal differential equation (ODE) solver called *ode*113, with initial condition $x0 = [0.2, 0.7, 0.1]^\top$ during 60 time units. Note that the number of populations and the mass of each population are defined by default to one. The simulation can be started by executing

```
G.run()
```

On the other hand, the following structure is used to define a population game with small number of agents per population:

```
G = struct('N', 200, 'n', 3, 'f', @fitness1, 'x0', [0.2
    .7 0.1 ]', 'ode', 'ode113', 'time', 10000, 'eta',
    0.02, 'revision_protocol', @proportional_imitation);
```

The finite population case uses the same parameters than the dynamical implementation, except for the dynamical model. However, it is necessary to define the revision protocol and the number of agents N per population. Table 2 contains the list of parameters required to run the revision protocol. The simulation of the revision protocol can be started by executing

```
G.run_finite()
```

The functions `G.graph()` and `G.graph_evolution()` are used to graph the simplex and the state evolution of the society for both cases.

3.2 Algorithms

In this section we introduce the algorithms used to implement each evolutionary dynamic, as well as some boundaries of their running time.

Field	Description	Default value
P	Number of populations	1
n	Number of pure strategies per population	-
S	Vector of pure strategies in each population	ones(G.P, 1) * G.n
m	Vector with the mass of each population	ones(G.P, 1)
x0	Initial state of the society	random
f	Function that returns a vector with the fitness of each strategy of a population	-
ode	ODE solver for the evolutionary dynamics	'ode45'
dynamics	Evolutionary dynamic. Current version support combinations of 'rd', 'maynard_rd', 'bnn', 'smith', 'logit'	'rd'
gamma	Defines the weight given to each dynamic when using combined dynamics	$\sum \gamma(i) = 1$

Table 1: Parameters of the dynamical implementation.

Field	Description	Default value
N	Number of agents	100
R	Rate of the Poisson clock	1
revision_protocol	Revision protocol. The current version support one of the following: comparison2average, pairwise_comparison, logit_choice, proportional_imitation.	'proportional_imitation'

Table 2: Parameters of the revision protocol.

3.2.1 Replicator Dynamics

input : Society's state x
output: State update $\dot{x}$

for $p \leftarrow 1$ **to** P **do**
  $F^p \leftarrow fitness(x, p)$;
  $\bar{F}^p \leftarrow \frac{1}{m^p}(F^p)^\top x^p$;
  $\hat{F}^p \leftarrow F^p - \mathbf{1}\bar{F}^p$;
  $\dot{x}^p \leftarrow \text{diag}(\hat{F}^p)x^p$;
end

The running time of the algorithm is $T_{rd}(n, P) = O(P(T_f(n, P)+n))$, where $T_f(n, P)$ is the time required to calculate the fitness vector of the p^{th} population.

3.2.2 BNN

input : Society's state x
output: State update $\dot{x}$

for $p \leftarrow 1$ **to** P **do**
  $F^p \leftarrow fitness(x, p)$;
  $\bar{F}^p \leftarrow \frac{1}{m^p}(F^p)^\top x^p$;
  $\hat{F}^p \leftarrow \max\{F^p - \mathbf{1}\bar{F}^p, \mathbf{0}\}$;
  $\dot{x}^p \leftarrow \hat{F}^p - (\mathbf{1}^\top \hat{F}^p)x^p$;
end

The running time is $T_{BNN}(n, P) = O(P(T_f(n, P) + n))$.

3.2.3 Smith Dynamics

In this case we need to order the strategies first and the calculate the difference between fitness functions in a matrix $\Delta_{ij} = [F_i^p(x) - F_j^p(x)]_+$. The running time of this algorithm is $T_{smith}(n, p) = O(P(T_f(n, P) + n^2))$.

input : Society's state x
output: State update $\dot{x}$

for $p \leftarrow 1$ **to** P **do**
  $F^p \leftarrow fitness(x, p)$;
  $B \leftarrow$ Strategies ordered in ascending order by their fitness ;
  for $i \leftarrow 2$ **to** n^p **do**
    for $j \leftarrow 1$ **to** $i - 1$ **do**
      $\alpha \leftarrow B[i]$;
      $\beta \leftarrow B[j]$;
      $\Delta_{ij} \leftarrow F_\alpha^p - F_\beta^p$;
    end
  end
  for $i \leftarrow 1$ **to** n^p **do**
    $\alpha \leftarrow B[i]$;
    for $j \leftarrow 1$ **to** $i - 1$ **do**
      $\beta \leftarrow B[j]$;
      $\Gamma_a^p[\alpha] \leftarrow \Gamma_a^p[\alpha] + x_\beta^p \Delta_{ij}^p$;
    end
    for $j \leftarrow i + 1$ **to** n^p **do**
      $\Gamma_b^p[\alpha] \leftarrow \Gamma_b^p[\alpha] + \Delta_{ij}^p$;
    end
  end
  $\dot{x}^p \leftarrow \Gamma_a^p - \mathrm{diag}(\Gamma_b^p)x^p$;
end

3.2.4 Logit dynamics

input : Society's state x
output: State update $\dot{x}$

for $p \leftarrow 1$ **to** P **do**
  $F^p \leftarrow fitness(x, p)$;
  $\bar{F}^p \leftarrow \frac{1}{m^p}(F^p)^\top x^p$;
  $\tilde{F}^p \leftarrow \exp(F^p \eta^{-1})$;
  $\Gamma \leftarrow \mathbf{1}^\top \tilde{F}^p$;
  $\dot{x}^p \leftarrow \frac{\tilde{F}^p}{\Gamma} - x^p$;
end

The running time is $T_{BNN}(n, P) = O(P(T_f(n, P) + n))$.

4 Combined Dynamics

It is possible to define a set of dynamics to run a combination of the dynamics. The resulting dynamic is defined as

$$\dot{x} = \sum_{d \in \mathcal{D}} \gamma_d V_d(x),$$

where $\mathcal{D} = \{Logit, RD, Smith, BNN\}$ denotes the set of available dynamics, $V_d()$ is the differential equation of the d^{th} dynamic and γ_d is the weight assigned to it. The dynamics should be defined in a cell array, e.g.,

```
dynamics = {'bnn', 'rd'};
```

The combination is made making a linear combination between each dynamic listed in the cell array. The weight assigned to each dynamic is defined in the vector `gamma`. In this case we assign

```
gamma = [.25, .75];
```

Fig. 5 shows an example of the combined dynamics for the rock-paper-scissors game. Note that the evolution of the system is not confined to a limit cycle, as happened with the replicator dynamics in Fig. 1.

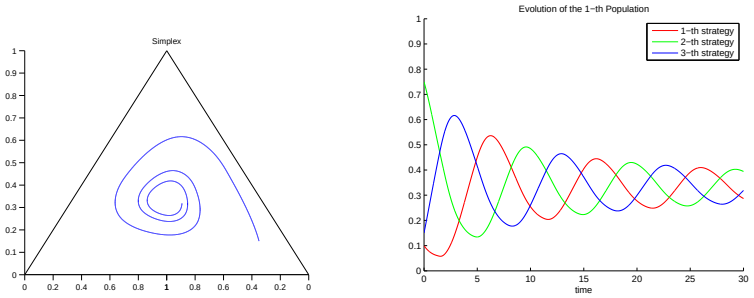
5 Multi-population Games

5.1 Matching pennies

We implement a matching pennies game defining a society $\mathcal{P} = \{p_1, p_2\}$ with two populations and two strategies per population, namely *heads* and *tails*. First, note that the payoff of the game in normal form is

2, 1	1, 2
1, 2	2, 1

Now, the fitness vector of the population p_j can be expressed as $F^{p_j}(x^{p_k}) = A^{p_j} x^{p_k}$, for $p_j, p_k \in \mathcal{P}$ and $p_j \neq p_k$. That is, the payoff of a population is



(a) Simplex. (b) Evolution of the strategies in time.

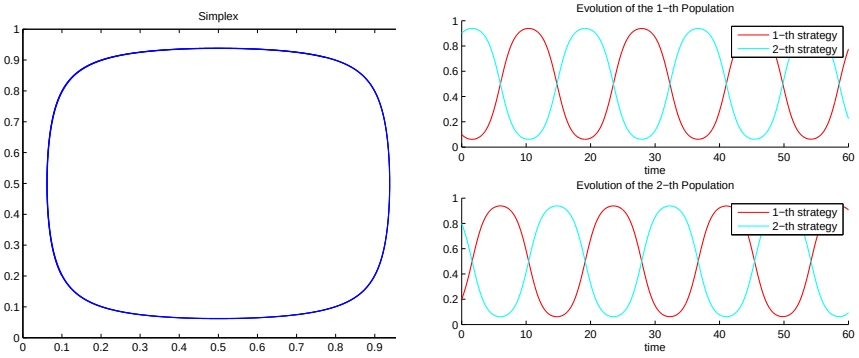
Figure 5: Evolution of the combination of replicator dynamics and BNN dynamics.

affected only by the state of the opponent population. The payoff matrices are defines as follows

$$A^1 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

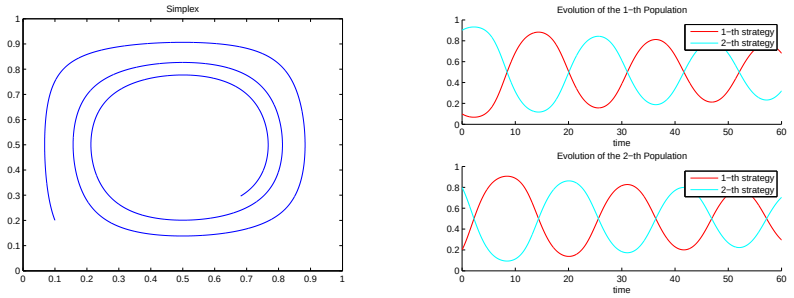
$$A^2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

Fig. 6 to 10 show the evolution of the social state with the evolutionary dynamics presented in Section 2.



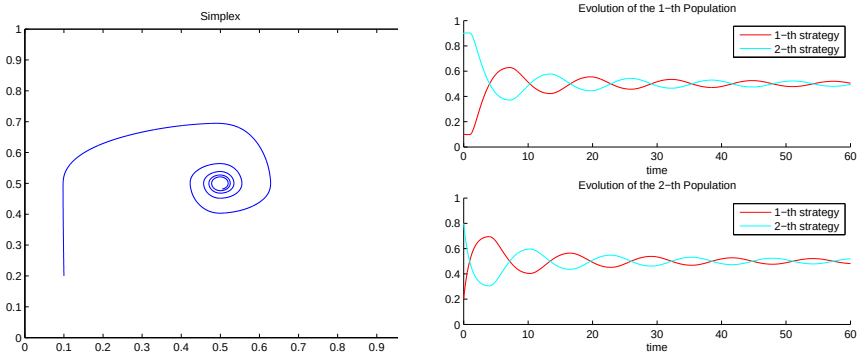
(a) Simplex. (b) Evolution of the strategies in time.

Figure 6: Matching pennies game with replicator dynamics.



(a) Simplex. (b) Evolution of the strategies in time.

Figure 7: Matching pennies game with Maynard replicator dynamics.



(a) Simplex. (b) Evolution of the strategies in time.

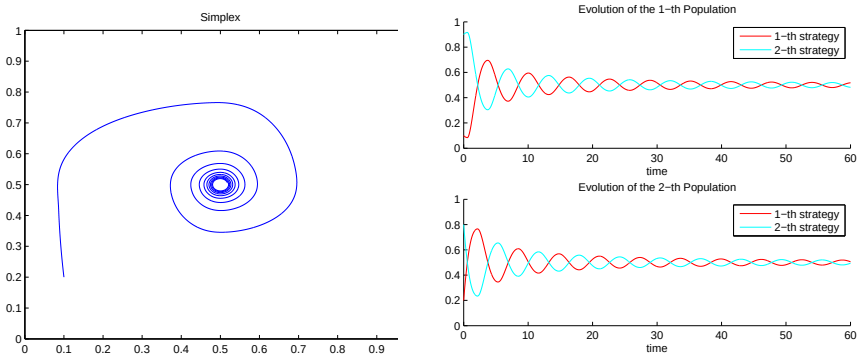
Figure 8: Matching pennies game with BNN dynamics.

5.2 Demand response programs

This is an example of multiple populations used to implement demand response programs in smart grids [1, 2]. In this case, we assume that each user must decide how to distribute its electricity usage along a day. Particularly, agents might have conflicting interests because they might impose externalities on the society through the price signals, i.e., the aggregated demand might affect the profit of agents. This conflict can be seen as a game between agents, in which each agent is selfish and endeavors to maximize independently its own welfare.

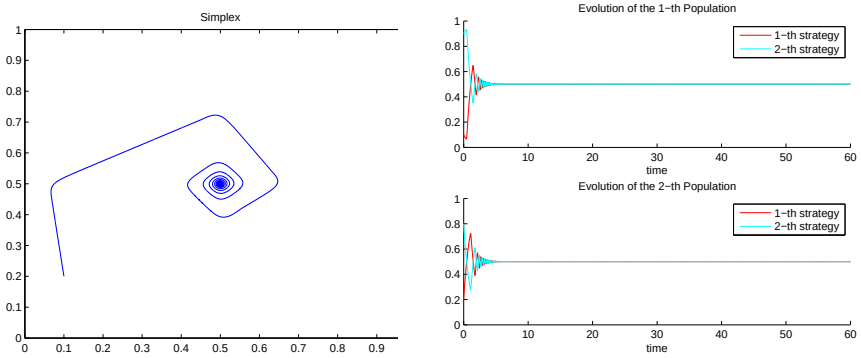
In this problem we model the daily electricity allocation problem as a multi-population game with nonlinear fitness functions. Particularly, each agent can implement an evolutionary dynamic to find the best distribution of resources. Note that when implemented locally by each user, the evolutionary dynamics lead to the global efficient equilibrium.

A particular feature of this problem is that the Nash equilibrium of the system is inefficient. Hence, we introduce an incentives scheme (indirect revelation mechanism) to maximize the aggregated surplus of the population. The main feature of this mechanism is that it does not require private information from users, and employs a one dimensional message space to coordinate the demand profile of agents. These properties facilitate the distributed implementation of



(a) Simplex. (b) Evolution of the strategies in time.

Figure 9: Matching pennies game with Smith dynamics.



(a) Simplex. (b) Evolution of the strategies in time.

Figure 10: Matching pennies game with Logit dynamics with $\eta = 0.02$.

the mechanism. The mechanism entrusts the computation tasks among users, who should maximize its own utility function based the aggregated demand (that is calculated and broadcasted by a central agent). Thus, users avoid revelation of private information (e.g., preferences), but are required to report the aggregated consumption of their appliances during some time periods.

5.2.1 Problem Formulation

We consider a population composed by N consumers defined as $\mathcal{V} = 1, \ldots . N$. Also, let us divide a period of 24 hours in a set of T time intervals denoted $\tau = \{\tau_1, \ldots, \tau_T\}$. Formally, we define the set τ as a partition of $[0, 24)$, where $\cup_{t\in\{1,\ldots,T\}}\tau_t = \tau$ and $\cap_{t\in\{1,\ldots,T\}}\tau_t = \varnothing$. Let q_i^t be the electricity consumption of the i^{th} user in the t^{th} time interval. The daily electricity consumption of the i^{th} user is represented by the vector $\boldsymbol{q}_i = [q_i^1, \ldots, q_i^T]^\top \in \Re^T_{\geq 0}$. The population consumption at a given time t is defined by the vector $\boldsymbol{q}^t = [q_1^t, , q_2^t \ldots, q_N^t]^\top \in \Re^N_{\geq 0}$. On the other hand, the joint electricity consumption of the whole population is denoted by $\boldsymbol{q} = [\boldsymbol{q}_1^\top, \ldots, \boldsymbol{q}_N^\top]^\top$. Without loss of generality, we assume that the electricity consumption of the i^{th} user satisfies $q_i^t \geq 0$, in each time instant t. A *valuation function* $v_i^t(q_i^t)$ models the *valuation* that the i^{th} user gives to an elec-

tricity consumption of q_i^t units in the t^{th} time interval. Finally, let $p(\cdot) : \Re \to \Re$ be the price of electricity charged to consumers. The aggregated consumption at a given time t is defined as $||\boldsymbol{q}^t||_1 = \sum_{j=1}^{N} q_j^t$. Moreover, a daily valuation is $v_i(\boldsymbol{q}_i) = \sum_{t=1}^{T} v_i^t(q_i^t)$, where $t \in \{1, \ldots, T\}$.

Now, assuming that the electricity generation cost is the same for all t, we can express the profit function of each individual as

$$U_i(\boldsymbol{q}) = v_i(\boldsymbol{q}_i) - \sum_{t=1}^{T} q_i^t p\Big(\left\|\boldsymbol{q}^t\right\|_1 \Big),$$

where $p : \Re_+ \to \Re_+$ is the unitary price function. The consumers welfare function is maximized by solving [6]

$$\begin{aligned} \underset{\boldsymbol{q}}{\text{maximize}} \quad & \sum_{i=1}^{N} U_i(\boldsymbol{q}) = \sum_{i=1}^{N} \left(v_i(\boldsymbol{q}_i) - \sum_{t=1}^{T} q_i^t p\left(\left\|\boldsymbol{q}^t\right\|_1\right) \right) \\ \text{subject to} \quad & q_i^t \geq 0, i = \{1, \ldots, N\}, t = \{1, \ldots, T\}. \end{aligned} \tag{2}$$

5.2.2 Incentives

The solution of the optimization problem in Eq. (2) is inefficient in a strategic environment, i.e., when individuals are rational and selfish [1,6]. In such cases, the analysis of strategic interactions among rational agents is made using game theory [3]. In particular, the Nash equilibrium (a solution concept in game theory) is sub-optimal, however, we can show that if we consider an added incentive to the individual cost function of each player, the Nash equilibrium of the game with incentives can be made efficient in the sense of Pareto [1,2].

In particular, our DR scheme with incentives models the case when all agents keep their valuation of electricity to themselves, and have autonomous control their consumption. However, in order to incentive the agents to modify their behavior for the good of the population, the central entity sends them an incentive (e.g., a price signal or reward) to indirectly control their load.

Consider the new cost function for the i^{th} agent:

$$W_i(q_i, \boldsymbol{q}_{-i}) = v_i(q_i) - q_i p\left(\left\|\boldsymbol{q}^t\right\|_1\right) + I_i(\boldsymbol{q}).$$

where incentives are of the form:

$$I_i(\boldsymbol{q}) = \left(\left\|\boldsymbol{q}_{-i}^t\right\|_1\right) \left(h_i(\left\|\boldsymbol{q}_{-i}\right\|) - p\left(\left\|\boldsymbol{q}^t\right\|_1\right)\right).$$

The form of this incentive is inspired in the Vickrey-Clarke-Groves mechanism and the Clarke pivot rule [7]. We assign incentives according to the contribution made by an agent to the society. In particular, the function $h_i : \Re \to \Re$ is a design parameter that estimates the externalities introduced by each individual. It can be shown that these incentives can lead to an optimal equilibrium in a strategic environment. In this DR approach we consider that the utility sends a two dimensional signal to each customer, namely (q, I_i) and each customer responds with some consumption q_i. Note that the incentives modify the price paid by each user according to their relative consumption. However, two different users receive different incentives as long as their consumption are different.

5.2.3 Simulations

In this section, we illustrate some ideas of efficiency and the decentralized implementation of the incentives mechanism. We select some functions used previously in the literature. On the one hand, we define the family of valuation functions as

$$v(\boldsymbol{q}^k, \alpha_i^k) = v_i^k(q_i^k) = \alpha_i^k \log(1 + q_i^k)$$

where $\alpha_i^k > 0$ is the parameter that characterizes the valuation of the i^{th} agent at the k^{th} time instant. On the other hand, the generation cost function is defined as

$$C(\|\boldsymbol{q}\|_1) = \beta(\|\boldsymbol{q}\|_1)^2 + b\|\boldsymbol{q}\|_1,$$

and the unitary price function is

$$p(\|\boldsymbol{q}\|_1) = \frac{C(\|\boldsymbol{q}\|_1)}{\|\boldsymbol{q}\|_1} = \beta\|\boldsymbol{q}\|_1 + b.$$

Note that the generation cost only depends on the aggregated consumption, not on the time of the day. Furthermore, the fitness function of the system with incentives is

$$F_i^k(\boldsymbol{q}^k) = \frac{\alpha_i^k}{1 + q_i^k} - 2\beta \left(\sum_{j=1}^{N} q_j^k \right).$$

The evolution of utility, demand, and incentives for different dynamics is shown in Figs. 11 and 12. Note that despite using the same initial condition, the evolution of the system is different with each dynamical model. In particular, BNN and Smith dynamics converge faster to the optimum, in contrast with the Logit and replicator dynamics. This is achieved by means of a fast decrease in the power consumption.

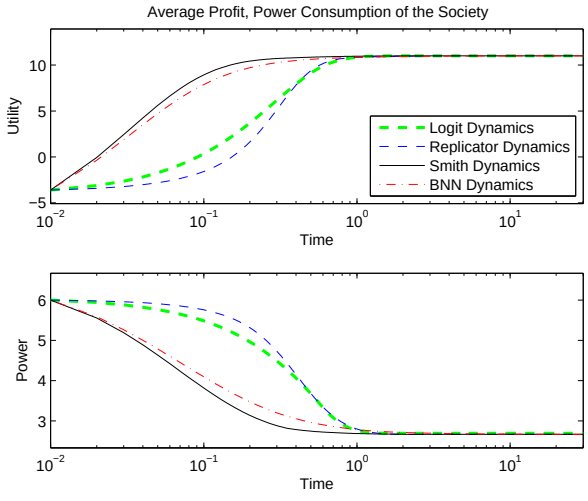


Figure 11: Evolution of profit and costs for four different dynamics.

Incentives in Fig. 12 show that, in the long run, all dynamics converge to the same level of incentives. Particularly, Smith dynamics requires more incentives

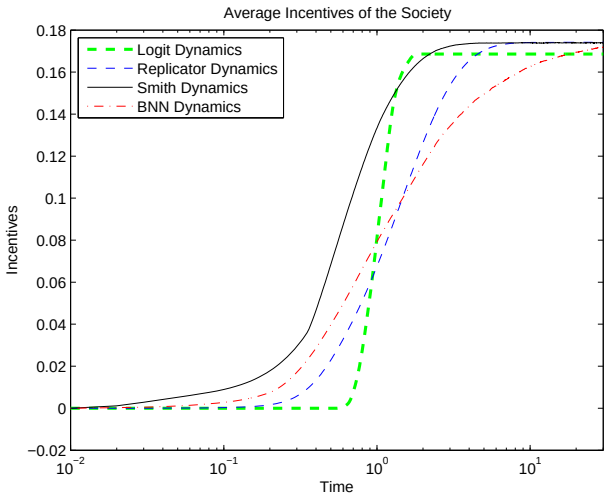


Figure 12: Evolution of the incentives with four different dynamics.

during all time, except for logit dynamics, which has a sudden increase in the incentives close to the equilibrium point.

In Fig. 12 it is not clear which dynamical model moves the state of the system to the optimal equilibrium using less resources. To answer this question, we simulate the total amount of incentives used by each model. Thus, let us define the aggregated incentives in a society in a particular time t as

$$I_d(t) = \sum_{i \in \mathcal{P}} \frac{1}{|S|} \sum_{k \in S} I_i \left(\boldsymbol{q}^k(t) \right).$$

Now, the total accumulated incentives from t_0 to t is defined as

$$\Phi_d(t) = \int_{t_0}^{t} I_d(\tau) d\tau.$$

Thus, $\Phi_d(t)$ gives a measurement of the total amount subsidies required by the system with dynamic d, in the time interval $[t_0, t]$. In this case we do not have a reference to compare the subsidies requirements of each evolutionary dynamic. Hence, we compare the subsidies requirements with the average requirements of all the dynamics implemented. In order to see which dynamic requires more resources, we plot the cumulative resources required by each dynamic relative to the average. Hence, we define the cumulative incentives as

$$CI_d = \frac{\Phi_d(t)}{\sum_{d \in \mathcal{D}} \Phi_d(t)}.$$

Fig. 13 shows the results of the simulation of the relative subsidies required by each model of evolutionary dynamics.

Smith dynamics requires much more resources during all the time stamp, but is particularly high during the first stages, while logit has the lower incentives requirements. However, BNN has the lower incentives in long run.

Fig. 14 shows the final demand profile of each agent. Note that the final state corresponds to the state of each population at the equilibrium.

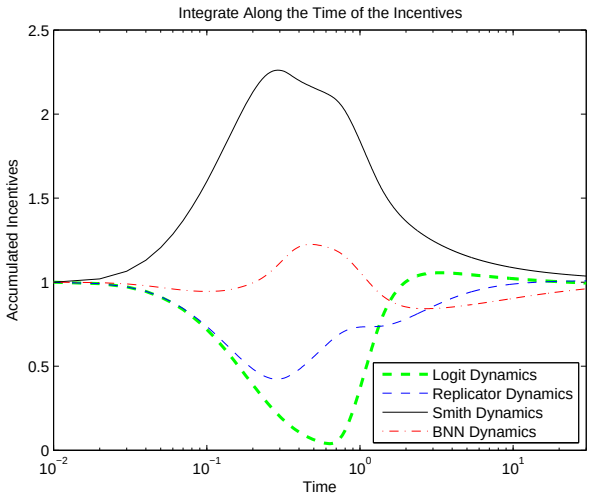


Figure 13: Accumulated incentives during the evolution of the algorithm.

References

[1] Carlos Barreto, Eduardo Mojica-Nava, and Nicanor Quijano. Design of mechanisms for demand response programs. In *Proceedings of the 2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 1828–1833, 2013.

[2] Carlos Barreto, Eduardo Mojica-Nava, and Nicanor Quijano. Incentives-based mechanism for efficient demand response programs. *arXiv preprint arXiv:1408.5366*, 2014.

[3] Drew Fudenberg and David K. Levine. *The Theory of Learning in Games*, volume 1 of *MIT Press Books*. The MIT Press, April 1998.

[4] David Gal. A psychological law of inertia and the illusion of loss aversion. *Judgment and Decision Making*, 1(1):23–32, July 2006.

[5] Josef Hofbauer. From nash and brown to maynard smith: equilibria, dynamics and ess. *Selection*, 1(1):81–88, 2001.

[6] Ramesh Johari and John N. Tsitsiklis. Efficiency of scalar-parameterized mechanisms. *Oper. Res.*, 57(4):823–839, July 2009.

[7] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 32 Avenue of the Americas, New York, NY 10013-2473, USA, 2007.

[8] William H. Sandholm. *Population Games and Evolutionary Dynamics (Economic Learning and Social Evolution)*. The MIT Press, 1 edition, January 2011.

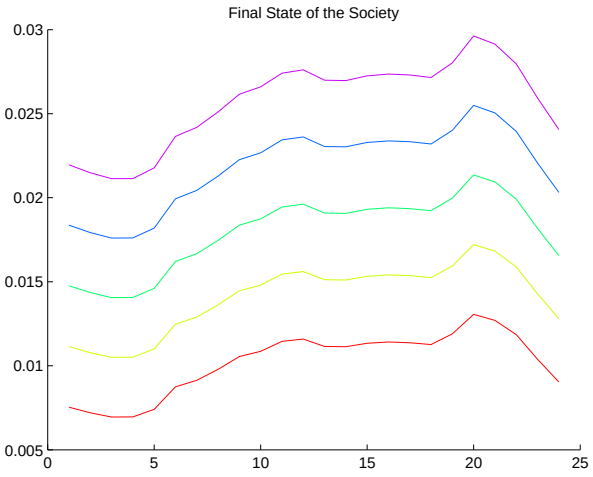


Figure 14: Final demand profile of each agent.