Aristotle University of Thessaloniki
Faculty of Engineering
Department of Electrical and Computer Engineering

Microscopic Analysis of Evolutionary Games

Group 8 Project for the Game Theory Course

Ioannis Konstantakis
(inkonstant@ece.auth.gr)

Ilias Mastrogiannis
(mastilia@ece.auth.gr)

Supervisor: Athanasios Kehagias
(kehagiat@ece.auth.gr)

May, 2025

Contents

Chapter 1

Introduction

1.1 Description of the Problem

1.1.1 The Classical Iterated Prisoner's Dilemma

The Classical Iterated Prisoner's Dilemma (CIPD) is one of the most studied models in evolutionary game theory, serving as a fundamental framework to understand the evolution of cooperation among self-interested agents, each of whom can either cooperate for mutual benefit or betray their partner ("defect") for individual gain. The classical version of the game involves two artificial agents, each faced with two possible actions:

- Cooperate (C): behave "nicely"
- Defect (D): acting selfishly or "naughtily"

The payoff for each agent is determined by the combination of actions chosen. These outcomes are governed by four central values:

- **R** (Reward): Both players cooperate → each receives 3
- **T** (Temptation): One defects while the other cooperates → defector receives 5
- **S** (Sucker's payoff): Cooperator receives 0 (exploited by the defector)
- **P** (Punishment): Both defect → each receives 1

This results in the following payoff matrix:

		Opponent	
		Cooperate	**Defect**
Player	**Cooperate**	$(R = 3,\ R = 3)$	$(S = 0,\ T = 5)$
	Defect	$(T = 5,\ S = 0)$	$(P = 1,\ P = 1)$

The dilemma arises from the inequality $T > R > P > S$, which implies that while mutual cooperation is collectively optimal, individual rationality leads players to defect—resulting in mutual punishment. This tension captures the conflict between the Nash equilibrium (both defect) and the Pareto optimum (both cooperate) [1].

In the one-shot version of the Prisoner's Dilemma, rational agents playing pure strategies are expected to defect, since defection strictly dominates cooperation. However, the *Classical Iterated Prisoner's Dilemma* (CIPD) alters the strategic environment: players interact repeatedly, and total payoff is the sum of rewards over multiple rounds. This repetition enables the use of history-dependent strategies—where current decisions are influenced by past behavior.

To foster cooperation in the CIPD, a commonly imposed constraint is:

$$S + T < 2R$$

This ensures that mutual cooperation yields a higher average payoff than alternating between exploiting and being exploited.

Over repeated interactions, a wide range of strategies becomes viable. Below are several simple yet influential ones (some of which are used in the simulations)

- **ALL-C**: always cooperates.
- **ALL-D**: always defects.
- **Tit for Tat (TFT)**: starts with cooperation and mimics the opponent's previous move.
- **Grim Trigger**: cooperates unless the opponent defects; then defects forever.
- **PER-CD**: alternates between cooperation and defection (pattern: C, D, C, D, ...).
- **PER-DDC**: cycles through the sequence D, D, C.
- **PER-CCCD**: cycles through the sequence C, C, C, D.
- **Soft Majority (SoftMajo)**: plays the opponent's most frequent move; defaults to cooperation in case of a tie.
- **Prober**: plays D, C, C initially; if the opponent cooperated on moves 2 and 3, switches to always defecting; otherwise, behaves like Tit for Tat.

1.1.2 Evolutionary Iterated Prisoner's Dilemma

An evolutionary game of IPD consists of the following components:

1. a *population* of players of the base game.
2. a *pool of available strategies*; each player adopts one of these strategies when playing the base game
3. a *match*, which is a play of the base game between two players.
4. a *meeting*, which consists of all possible matches between the players.

Given the above components, an evolutionary game of IPD is a sequence of meetings combined with some evolution dynamics. In the first meeting, played between the first generation of players, the adoption of strategies by the players can be arbitrary (e.g. 100 players playing TFT, 100 players playing ALL-C, 100 players playing ALL-D). At the end of each meeting, a new generation of players is produced, in which the

strategies are adopted by the players in some manner (evolutionary dynamic) such that higher performing strategies have a higher representation in the population (e.g. 80 players playing TFT, 80 players playing ALL-C, 140 players playing ALL-D).

It is expected that, as the number of generations increases, the consistently better performing strategies will dominate; often (but not always) after a sufficiently long number of generations, a single strategy survives. More complicated behaviors (oscillations, chaos) can also be observed.

The evolution dynamics is the mechanism by which strategies are propagated to the next generation. In this report, we study two dynamics (via simulations in MATLAB):

1. *Fitness dynamics*. The number of players who adopt a strategy in a generation is proportional to the performance (measured by total payoff) of the strategy.
2. *Imitation dynamics*. In every generation, a number of players adopt the strategy used by one of the best-performing strategies (or players) of the previous generation.

The key challenge in CIPD research lies not only in identifying successful strategies but also in understanding the dynamics of agent populations using these strategies. These dynamics reveal how certain strategies dominate, coexist, or vanish over time, shedding light on the conditions that favor cooperation in evolving systems.

1.2 Related Work

Despite its simplicity, the Iterated Prisoner's Dilemma (IPD) has become a powerful theoretical model for analyzing the evolution of cooperation in populations subject to Darwinian selection. This model, originating in classical game theory, has been widely adopted across disciplines such as mathematics, biology, computer science, sociology, and zoology [2].

The foundational work of Axelrod and Hamilton demonstrated how cooperative behavior can emerge even in environments where defection is the dominant Nash strategy [1]. Their findings were later expanded by Axelrod and Dion, who explored the conditions under which cooperation could persist or break down [3]. These contributions laid the groundwork for a large body of work in evolutionary game theory, particularly focused on strategy dynamics in repeated games such as the IPD.

In the IPD, players repeatedly decide whether to cooperate or defect, with long-term payoffs depending on both players' previous actions. Classic strategies such as Always Cooperate (AllC), Always Defect (AllD), and Tit-for-Tat (TFT) have been studied extensively. TFT, for instance, was the winning strategy in Axelrod's original tournaments and inspired decades of theoretical and experimental research. However, it is known to be vulnerable to noise and neutral drift, motivating the development of more robust strategies such as Win-Stay, Lose-Shift (WSLS).

Much of the early work on evolutionary dynamics was based on deterministic models, especially replicator dynamics [2]. These models assume infinite populations and continuous strategy updates, but recent work has emphasized the importance

of stochastic effects and finite population models, especially in the presence of mutation, drift, and imitation dynamics. These advances incorporate methods from statistical physics and Markov processes to analyze fixation probabilities, convergence, and stability of strategies in evolving populations.

Our work builds directly on the influential research of Mathieu, Beaufils, and Delahaye, who investigated the dynamics of small strategy sets in the IPD using a blend of theoretical analysis and computer simulation [4]. The present report reproduces and extends several of their experiments, including comparisons between theoretical predictions and empirical simulations.

Chapter 2

Quick Start

This project is implemented as an open-source MATLAB toolbox designed to simulate and analyze evolutionary dynamics in the Classical Iterated Prisoner's Dilemma. It allows users to explore both Fitness Dynamics (see Chapter 3) and Imitation Dynamics (see Chapter 4) across a variety of predefined strategies. After installation, users can immediately reproduce key results from the report as well as explore custom scenarios of their liking (changing initial population numbers, number of rounds, etc.).

2.1 Key Features

Some core capabilities include:

- Simulates population evolution with custom strategies.
- Supports both theoretical and simulation-based computation of evolutionary outcomes.
- Generates payoff matrices, state transition graphs (see 4.2 Experiments), and time evolution plots.
- Includes predefined strategy sets (e.g., ALL-D, ALL-C, Tit-for-Tat, Grim, periodic strategies, and more), but can also be extended with custom payoff matrices, strategies, or simulation parameters.
- Reproduces all 12 figures from Delahaye [4] (see 3.2 Experiments)

Note: For detailed function documentation and advanced usage, see Appendix B.

2.2 Installation

1. Obtain the toolbox source code by cloning or downloading the repository from GitHub:

   `https://github.com/inkonstant/EvolutionaryGamesToolbox`

For instance, on a Windows system, open a Command Prompt or PowerShell and execute:

```
git clone https://github.com/inkonstant/EvolutionaryGamesToolbox
```

2. If the GIT repository is downloaded as a ZIP file, extract its contents to a directory of your choice.
3. Open MATLAB and run the provided `setup.m` script to add all necessary paths.

2.3 Running Examples

After installation, you can immediately run over 20 ready-to-run scripts in the `Examples` directory to generate all the results presented in both Fitness Dynamics and Imitation Dynamics

Chapter 3

Fitness Dynamics

3.1 The Process

3.2 Fitness Dynamics

In fitness dynamics, the transition between player generations follows a proportional update rule:

- The new generation's strategy distribution is determined by each strategy's relative performance.
- Specifically, a strategy's representation grows in proportion to its payoff relative to the total payoff across all strategies.

Similar to imitation dynamics, the true state space consists of all possible strategy compositions. However, fitness dynamics fundamentally operates on:

- A continuous space of strategy proportions.

The standard analytical framework (which is adopted here) focuses on:

- The dynamics of strategy proportions rather than individual player transitions.
- Application of dynamical systems theory to study the evolutionary trajectories.
- Continuous approximation of the discrete population dynamics.

3.2.1 Theoretical Approach

Our analysis builds upon the evolutionary framework proposed by [4], adopting their population dynamics formulas (Equations 3-5). However, since we lacked the payoff matrix $V(X|Y)$ representing scores when strategy X meets Y, we generated these values empirically through Axelrod tournament simulations.

The Axelrod tournament provided:

- Pairwise payoff values $V(\cdot|\cdot)$ for all strategy combinations.
- Empirical results under controlled conditions (T moves per match).

- Consistent measurement across all strategy pairings.

This approach maintains the theoretical framework while grounding the payoff values in actual strategy interactions, with $W_n(X)$ representing the population proportion of strategy X at generation n, and $\mathbf{g}_n(X)$ denoting the fitness score.

The above protocol can be described by the following pseudocode:

Algorithm 1 TourTheFit

Function TourTheFit(B, Strategies[1..S], POP0[1..S], T, J)
$V \leftarrow S \times S$ matrix of payoffs over T rounds
$\Pi \leftarrow$ total number of players
for j from 1 to J **do**
  $W \leftarrow$ current population vector (length S)
  for x from 1 to S **do**
    ▷ Compute fitness $g[x]$ for each strategy x
    $g[x] \leftarrow \left(\sum_{y=1}^{S} W[y] \cdot V[x, y]\right) - V[x, x]$
  end for
  FIT[j] $\leftarrow g$
  $t \leftarrow \sum_{i=1}^{S} W[i] \cdot g[i]$
    ▷ Update population proportions
  for i from 1 to S **do**
    $\text{POP}[j][i] \leftarrow \dfrac{\Pi \cdot W[i] \cdot g[i]}{t}$
  end for
  BST[j] $\leftarrow$ index_of_max(POP[j]) ▷ Best strategy this generation
end for
return POP, BST, FIT

3.2.2 Simulation Approach

The simulation implementation follows the same theoretical framework, but with two key methodological differences in the player interactions:

- **Discrete Population:** We model individual players rather than continuous population proportions.
- **Match-Based Payoffs:** Players compete in pairwise tournaments, with the V matrix calculated empirically through:

$$V(X|Y) = \texttt{calculateV}(X, Y)$$

  where `calculateV` aggregates results from repeated head-to-head matches.

This approach maintains mathematical consistency while better reflecting discrete evolutionary dynamics in finite populations.

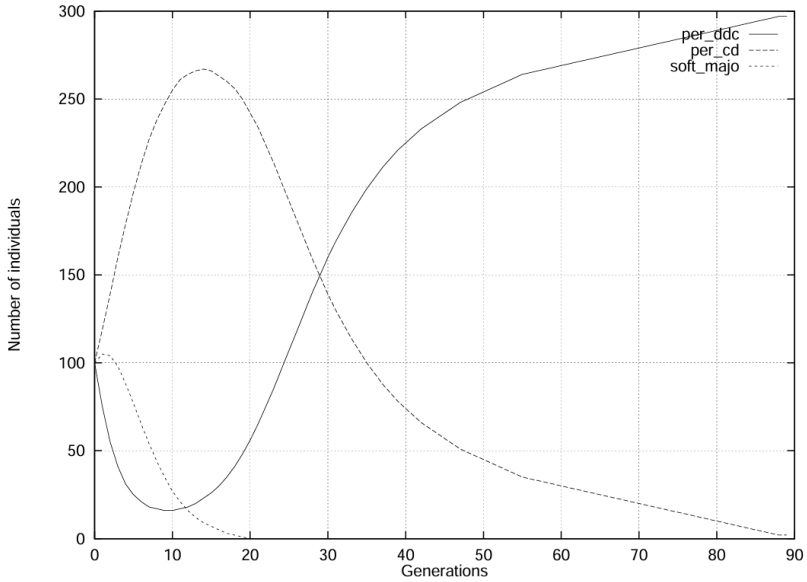
The above methodology can be described by the following pseudocode:

Algorithm 2 TourSimFit

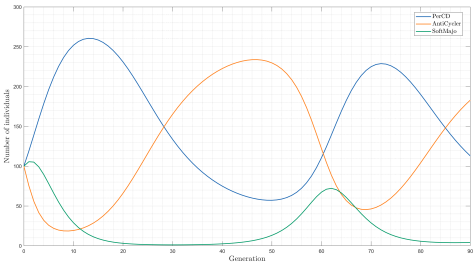
Function TourSimFit(B, Strategies[1..S], POP0[1..S], T, J)
$V \leftarrow S \times S$ matrix of payoffs over T rounds
$\Pi \leftarrow$ total number of players
for j from 1 to J **do**
    $W \leftarrow$ current population vector (length S)
    ▷ Expand population into individual indices
    strat_idx $\leftarrow$ concatenate s repeated $W[s]$ times for $s = 1$ to S
    ▷ Compute per-type score
    for x from 1 to S **do**
        perTypeScore[x] $\leftarrow \left(\sum_{y=1}^{S} W[y] \cdot V[x, y]\right) - V[x, x]$
    end for
    ▷ Assign scores and aggregate by type
    for i from 1 to Π **do**
        scores[i] $\leftarrow$ perTypeScore[strat_idx[i]]
    end for
    for s from 1 to S **do**
        totByType[s] $\leftarrow \sum_{\text{strat\_idx}[i]=s}$ scores[i]
        $g[s] \leftarrow \dfrac{\text{totByType}[s]}{\sum_{i=1}^{\Pi} \text{scores}[i]}$
    end for
    FIT[j] $\leftarrow g$
    ▷ Update population counts using floor and distribute remainder
    for s from 1 to S **do**
        $W_{\text{next}}[s] \leftarrow \lfloor \Pi \cdot g[s] \rfloor$
    end for
    rem $\leftarrow \Pi - \sum_{s=1}^{S} W_{\text{next}}[s]$
    if rem > 0 **then**
        order $\leftarrow$ indices of g sorted in descending order
        for k from 1 to rem **do**
            $W_{\text{next}}[\text{order}[k]] \leftarrow W_{\text{next}}[\text{order}[k]] + 1$
        end for
    end if
    POP[j] $\leftarrow W_{\text{next}}$
    BST[j] $\leftarrow$ index_of_max(W_{next})
end for
return POP, BST, FIT

3.3 Experiments

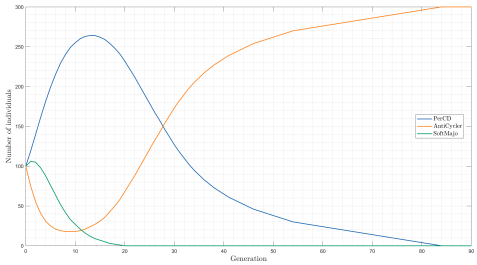
Our primary goal is to reproduce and extend the experimental findings in [4], specifically *Figures* 1 – 12, providing both theoretical and empirical insight into how simple strategies interact over time under evolutionary pressures.



(a) MBD original figure

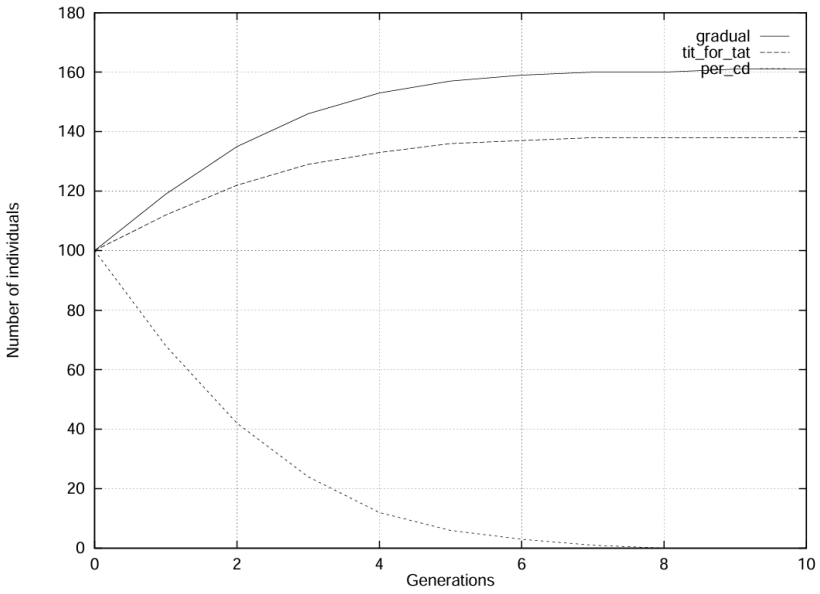


(b) Our theoretical analysis

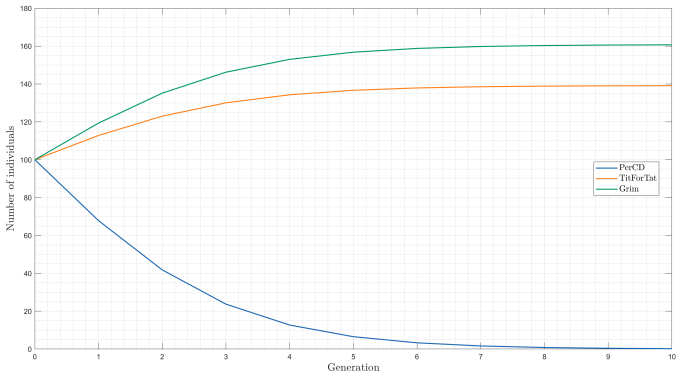


(c) Our simulation result

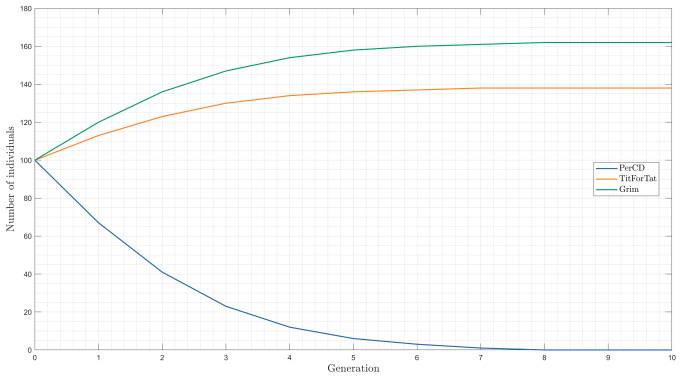
Figure 3.1: Comparison of results for Fig. 1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_1.m`), and (c) simulation output (see script `demo_fit_sim_1.m`).



(a) MBD original figure

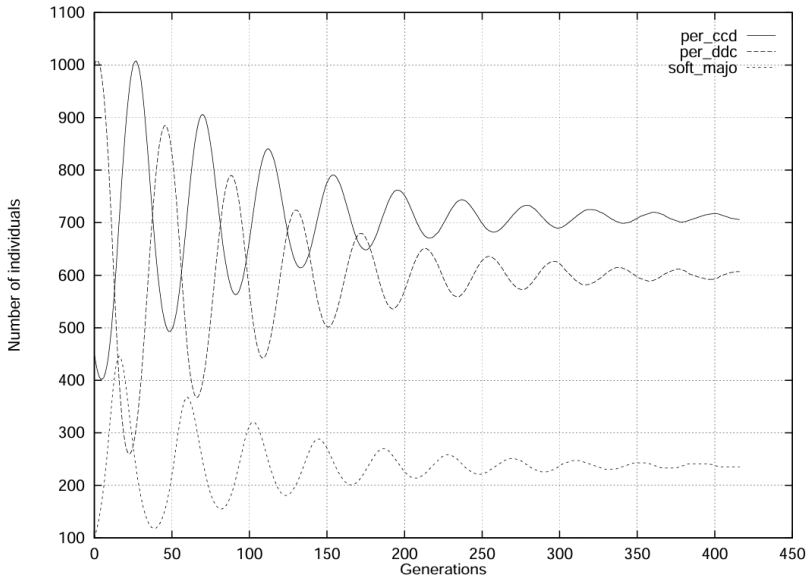


(b) Our theoretical analysis

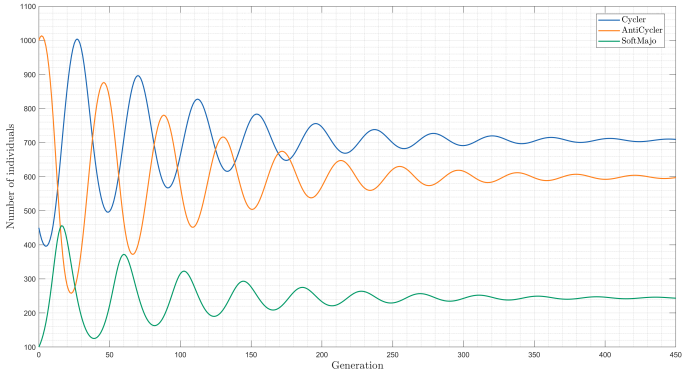


(c) Our simulation result

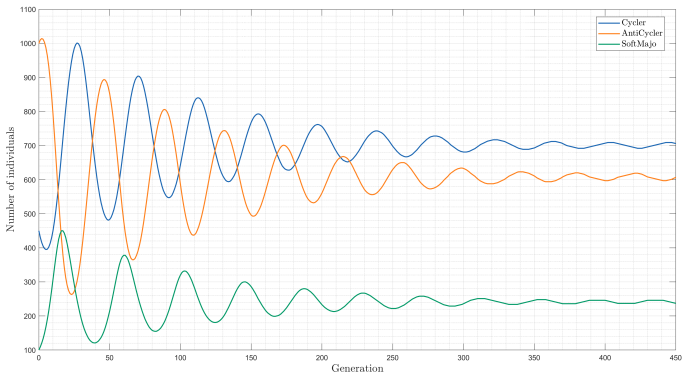
Figure 3.2: Comparison of results for Fig. 2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_2.m`), and (c) simulation output (see script `demo_fit_sim_2.m`).



(a) MBD original figure

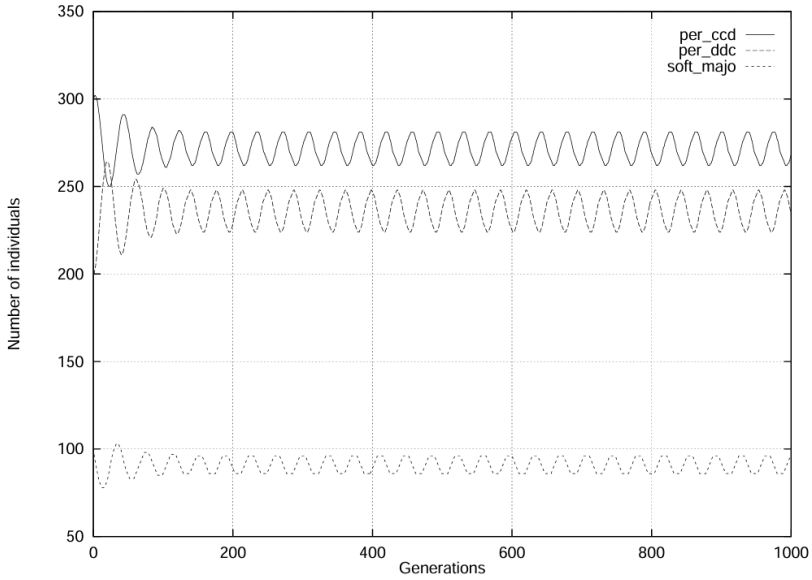


(b) Our theoretical analysis

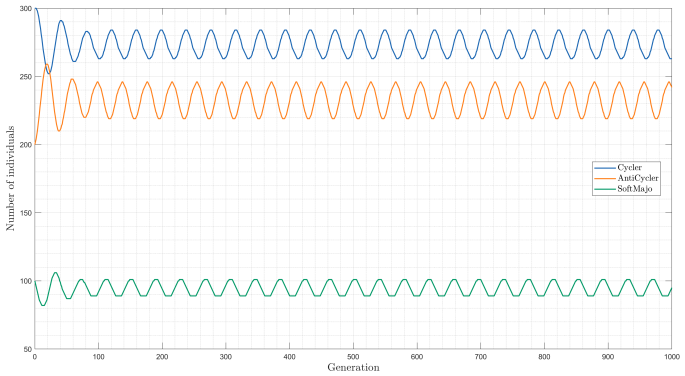


(c) Our simulation result

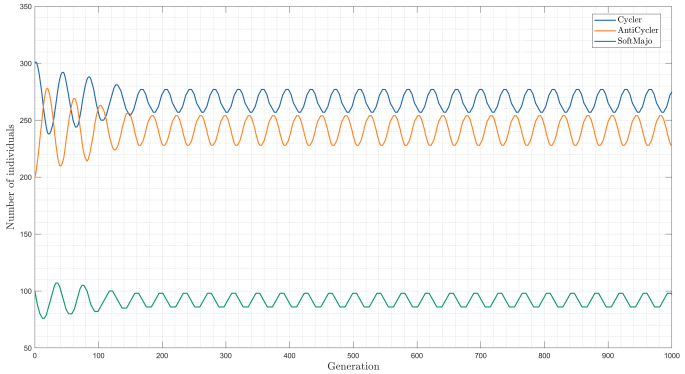
Figure 3.3: Comparison of results for Fig. 3 (a) from [4], (b) our theoretical model (see script `demo_fit_the_3.m`), and (c) simulation output (see script `demo_fit_sim_3.m`).



(a) MBD original figure

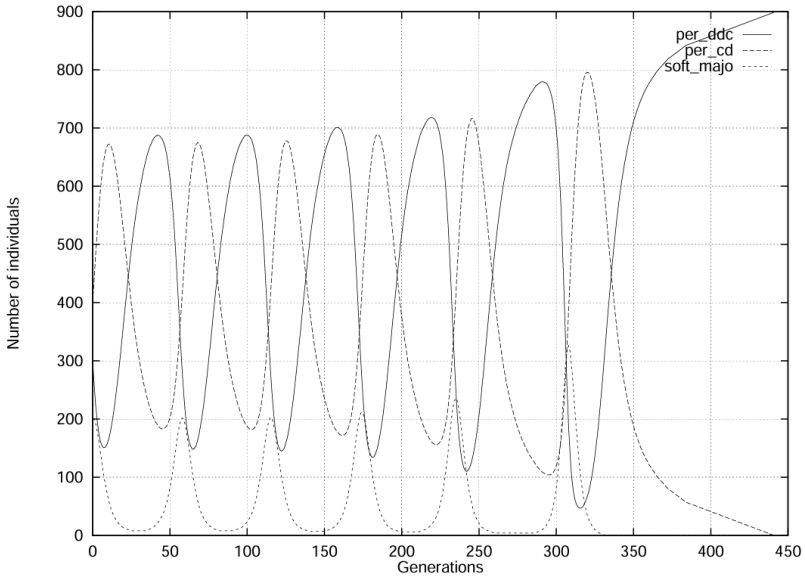


(b) Our theoretical analysis

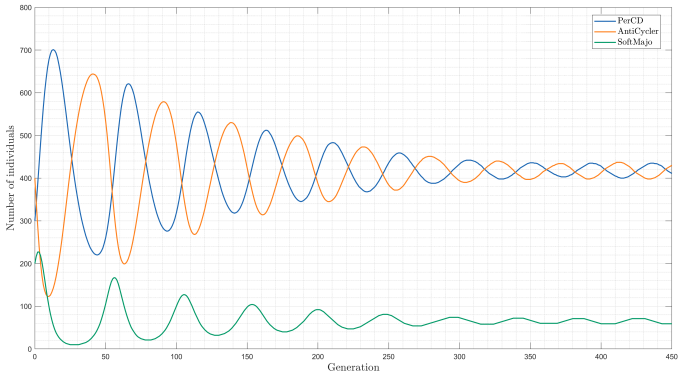


(c) Our simulation result

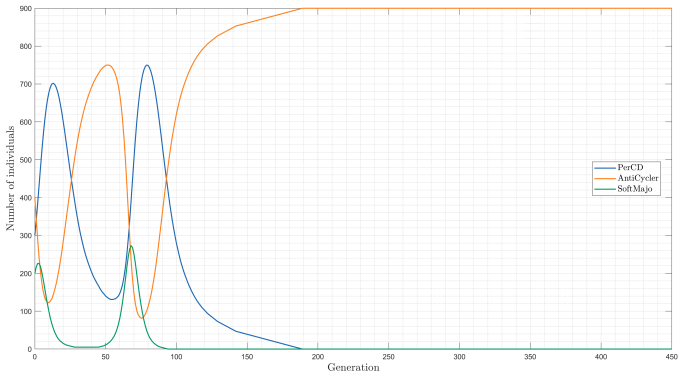
Figure 3.4: Comparison of results for Fig. 4 (a) from [4], (b) our theoretical model (see script `demo_fit_the_4.m`), and (c) simulation output (see script `demo_fit_sim_4.m`).



(a) MBD original figure

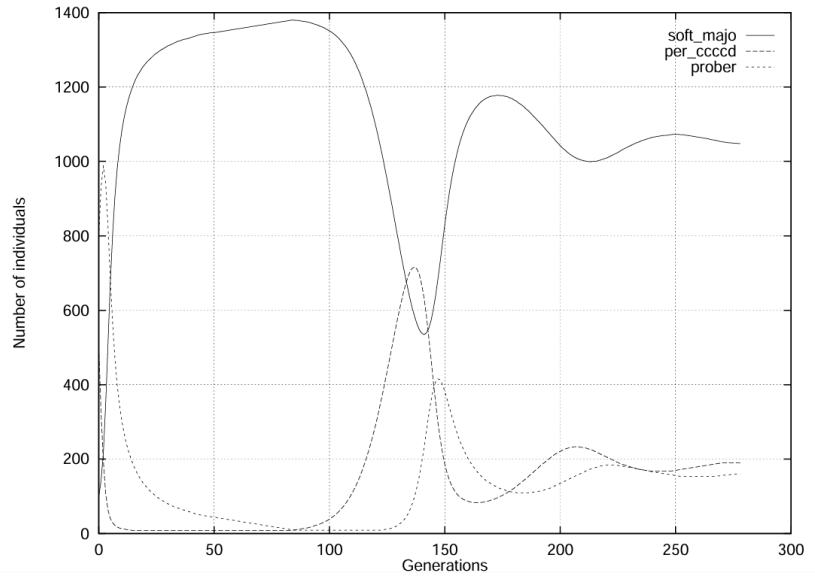


(b) Our theoretical analysis

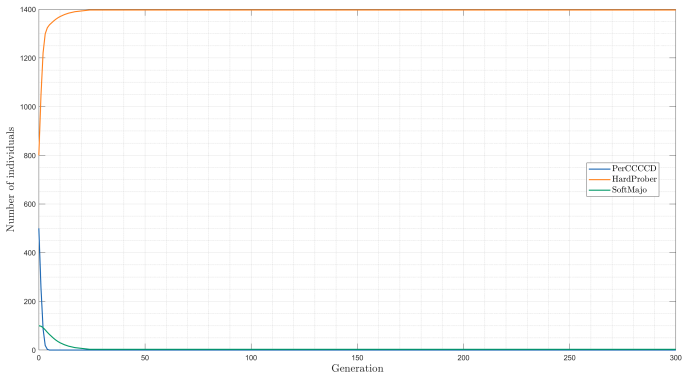


(c) Our simulation result

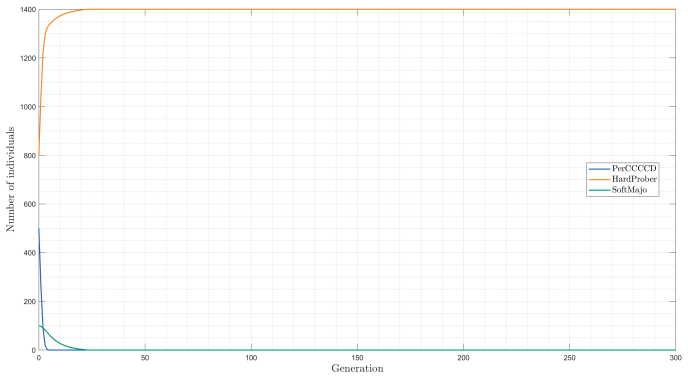
Figure 3.5: Comparison of results for Fig. 5 (a) from [4], (b) our theoretical model (see script `demo_fit_the_5.m`), and (c) simulation output (see script `demo_fit_sim_5.m`).



(a) MBD original figure

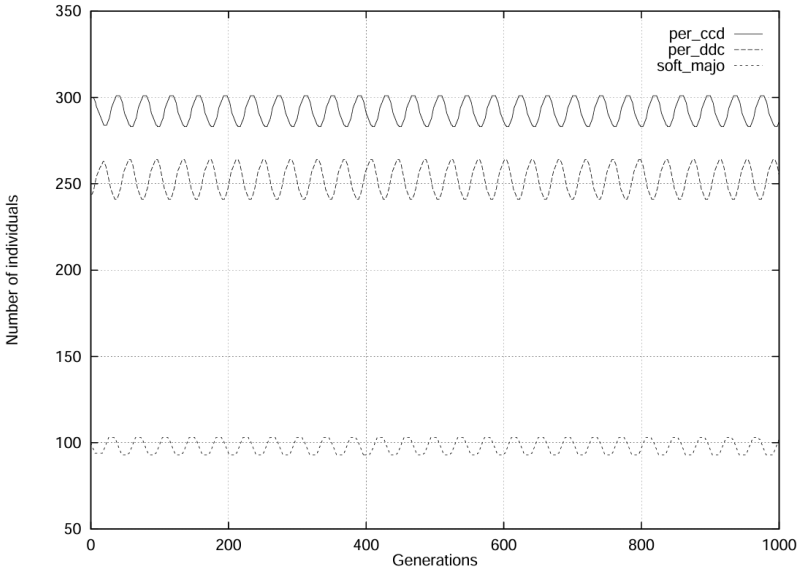


(b) Our theoretical analysis

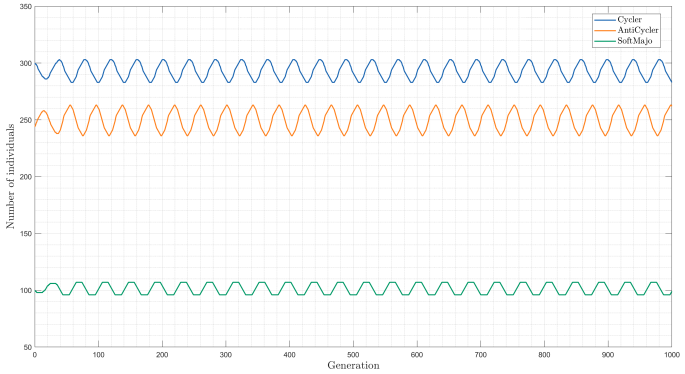


(c) Our simulation result

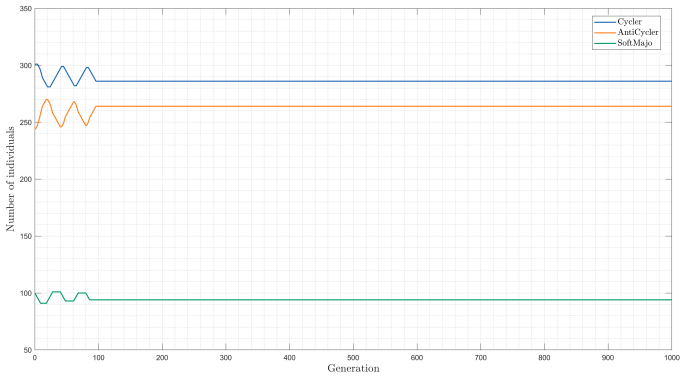
Figure 3.6: Comparison of results for Fig. 6 (a) from [4], (b) our theoretical model (see script `demo_fit_the_6.m`), and (c) simulation output (see script `demo_fit_sim_6.m`).



(a) MBD original figure

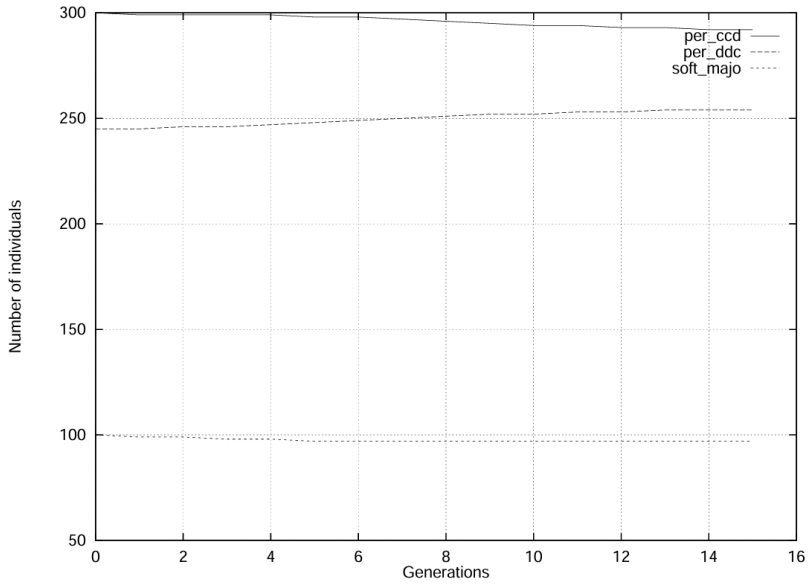


(b) Our theoretical analysis

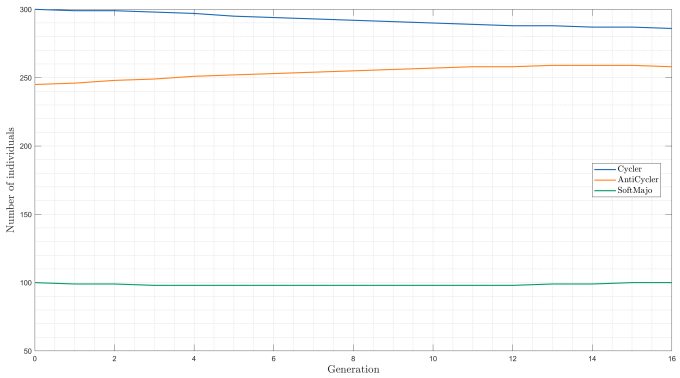


(c) Our simulation result

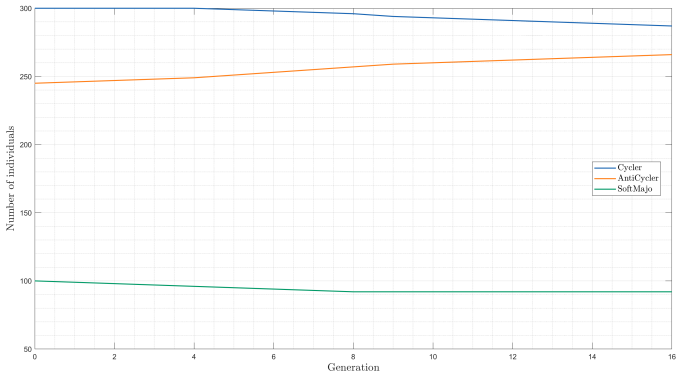
Figure 3.7: Comparison of results for Fig. 7.1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_7_1.m`), and (c) simulation output (see script `demo_fit_sim_7_1.m`).



(a) MBD original figure

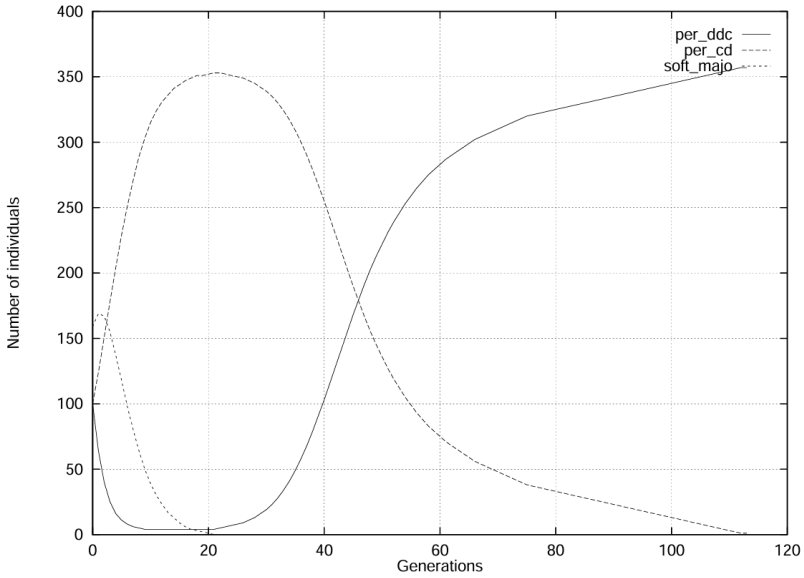


(b) Our theoretical analysis

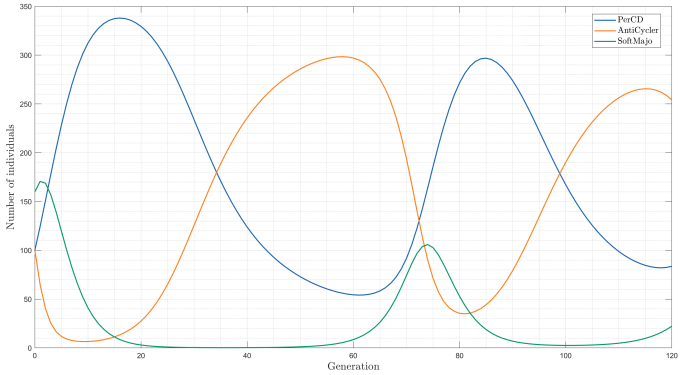


(c) Our simulation result

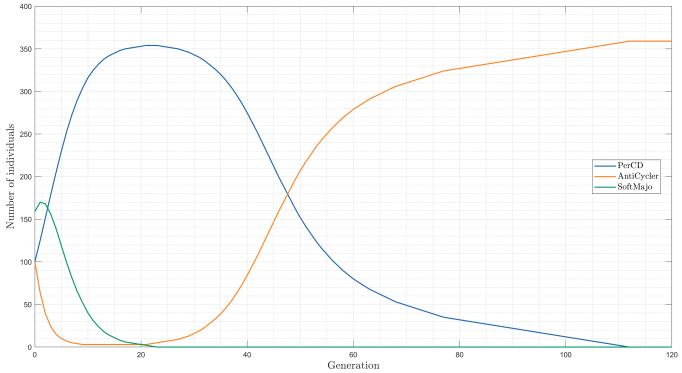
Figure 3.8: Comparison of results for Fig. 7.2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_7_2.m`), and (c) simulation output (see script `demo_fit_sim_7_2.m`).



(a) MBD original figure

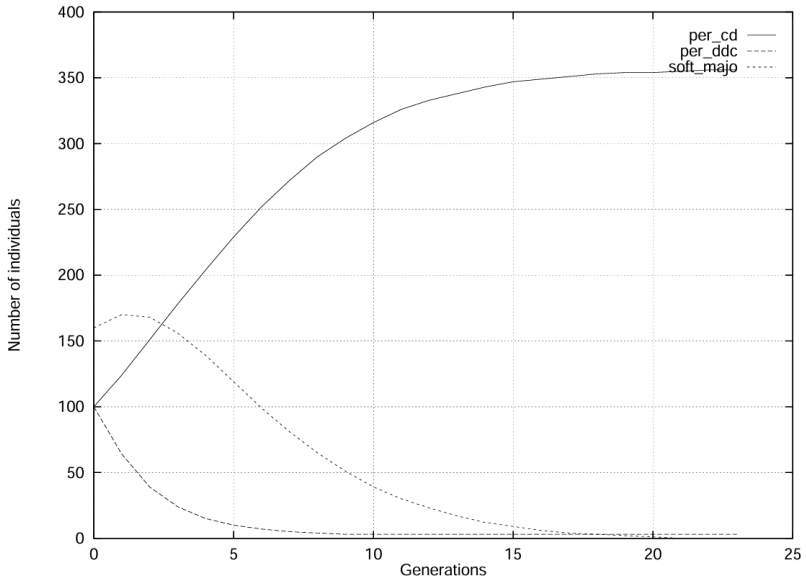


(b) Our theoretical analysis

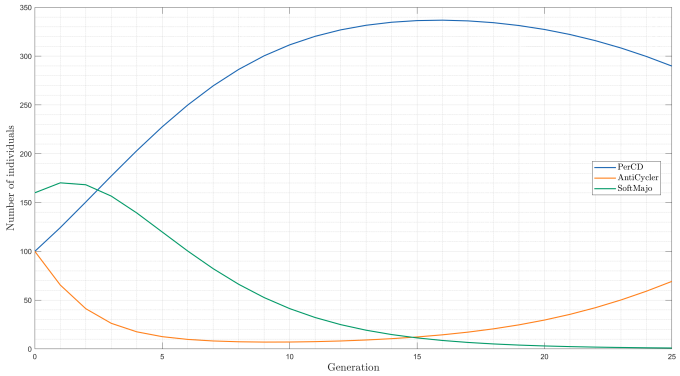


(c) Our simulation result

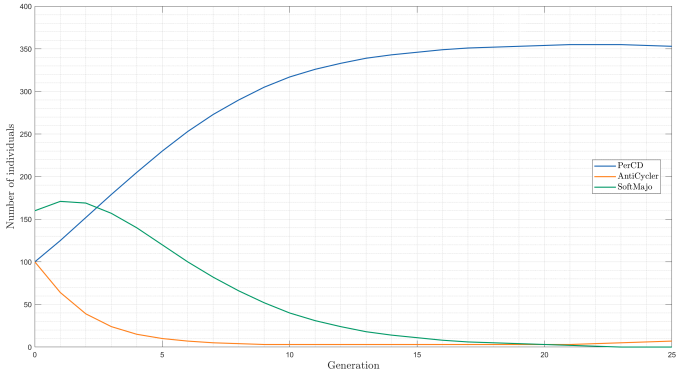
Figure 3.9: Comparison of results for Fig. 8.1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_8_1.m`), and (c) simulation output (see script `demo_fit_sim_8_1.m`).



(a) MBD original figure

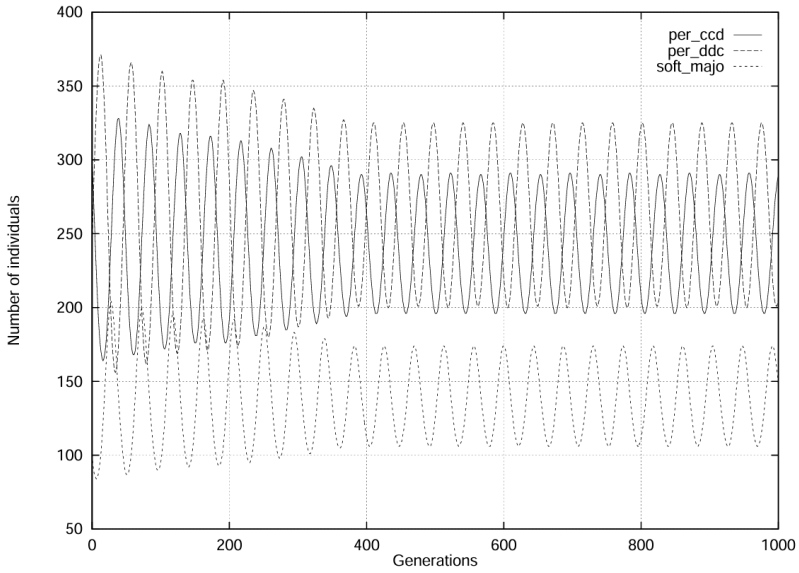


(b) Our theoretical analysis

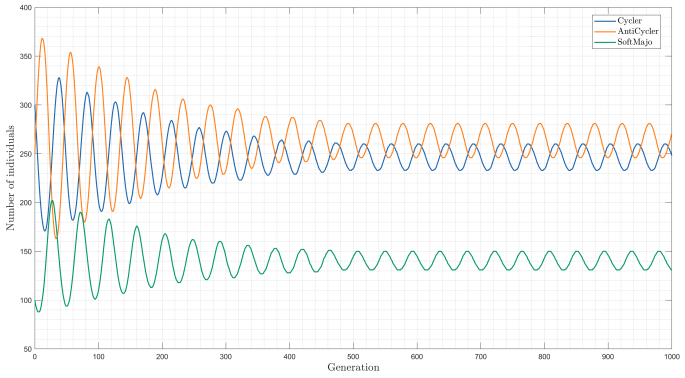


(c) Our simulation result

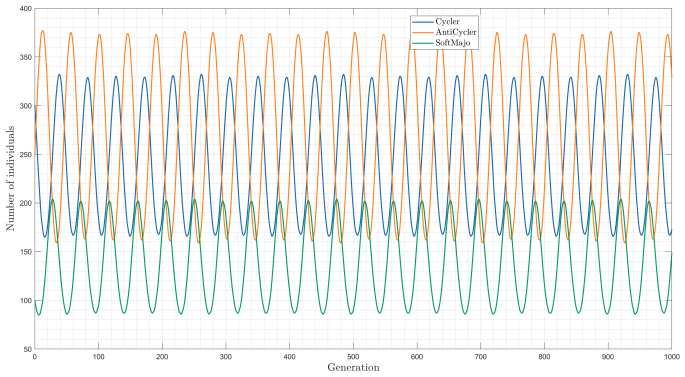
Figure 3.10: Comparison of results for Fig. 8.2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_8_2.m`), and (c) simulation output (see script `demo_fit_sim_8_2.m`).



(a) MBD original figure

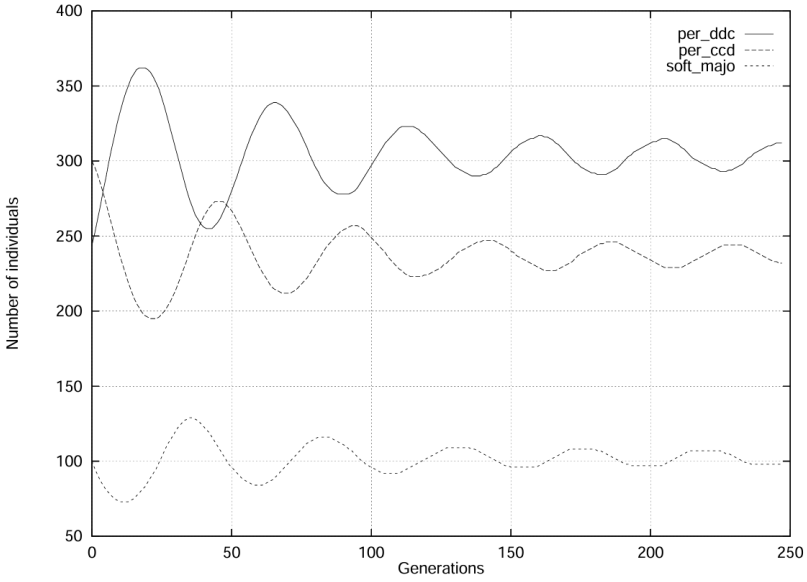


(b) Our theoretical analysis

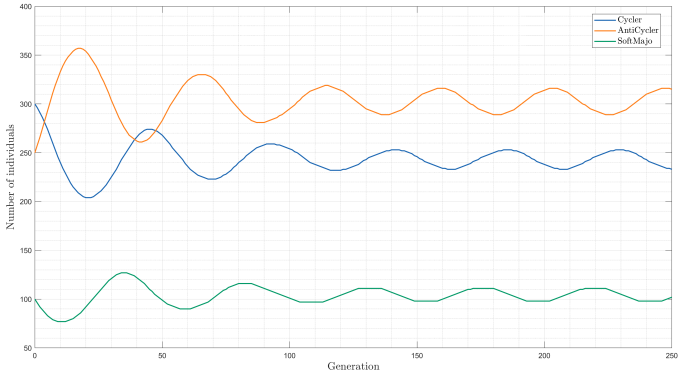


(c) Our simulation result

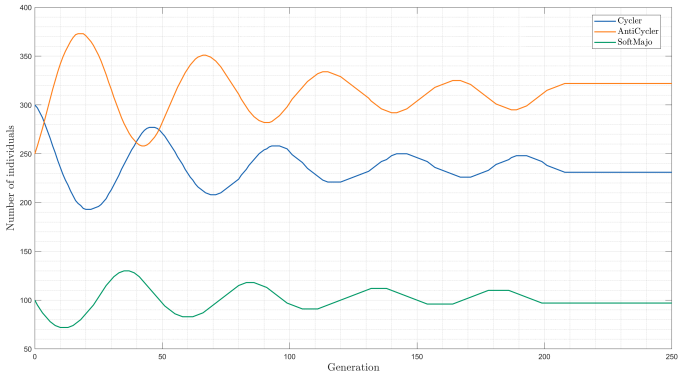
Figure 3.11: Comparison of results for Fig. 9.1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_9_1.m`), and (c) simulation output (see script `demo_fit_sim_9_1.m`).



(a) MBD original figure

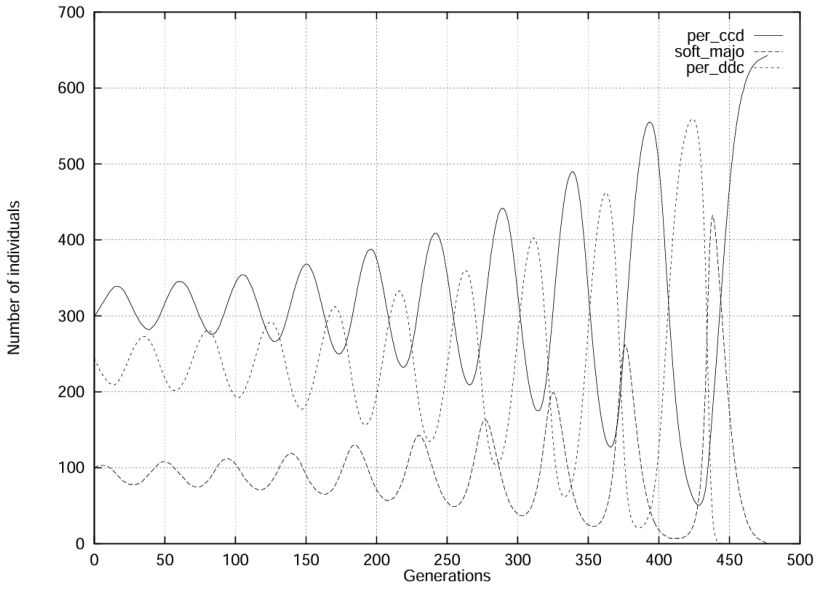


(b) Our theoretical analysis

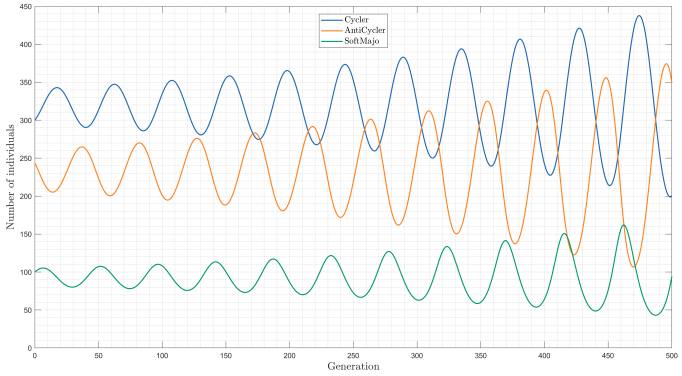


(c) Our simulation result

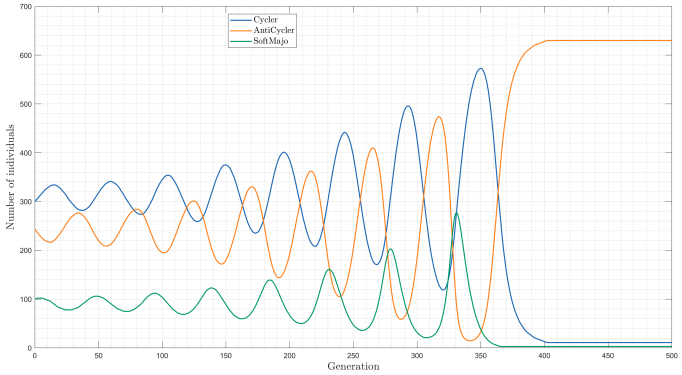
Figure 3.12: Comparison of results for Fig. 9.2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_9_2.m`), and (c) simulation output (see script `demo_fit_sim_9_2.m`).



(a) MBD original figure

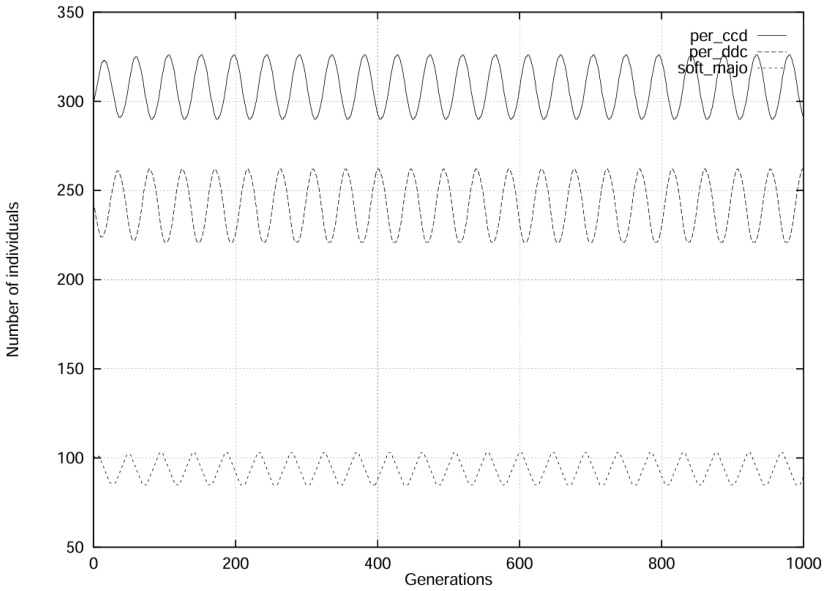


(b) Our theoretical analysis

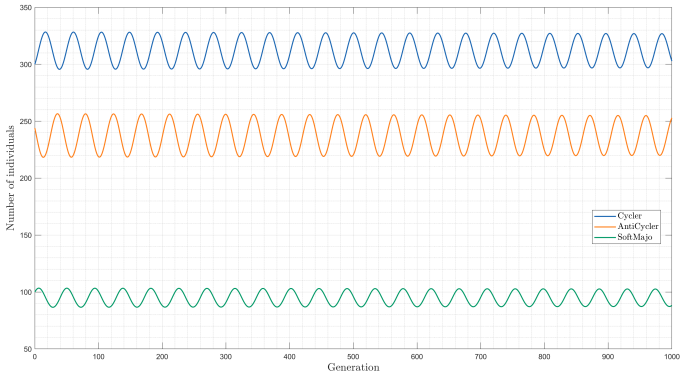


(c) Our simulation result

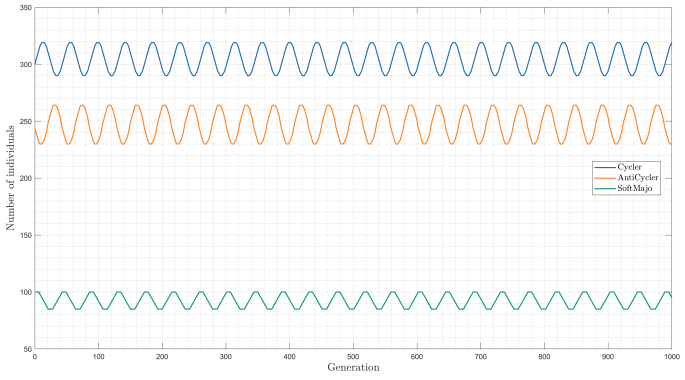
Figure 3.13: Comparison of results for Fig. 10.1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_10_1.m`), and (c) simulation output (see script `demo_fit_sim_10_1.m`).



(a) MBD original figure

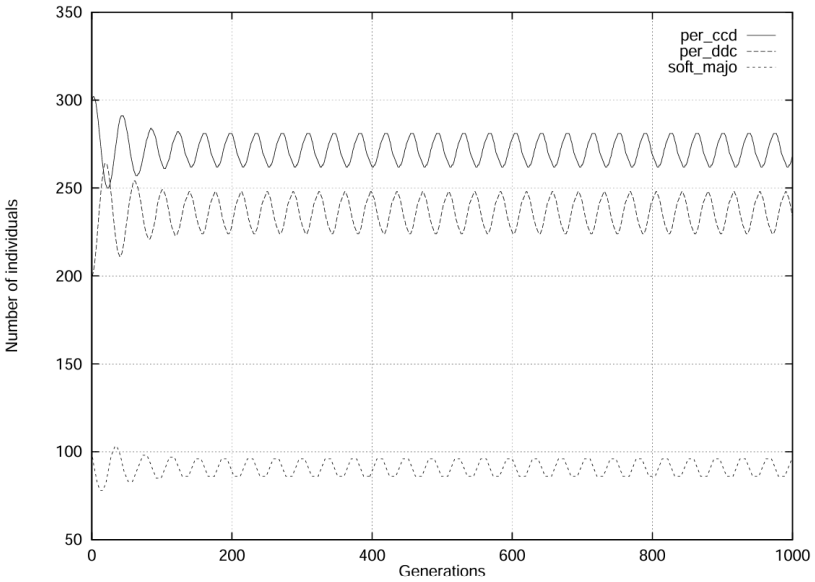


(b) Our theoretical analysis

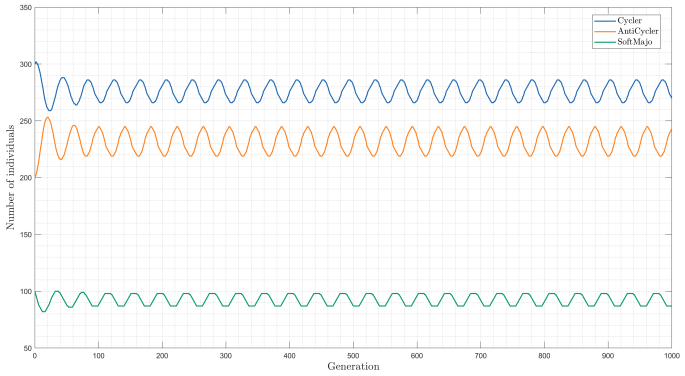


(c) Our simulation result

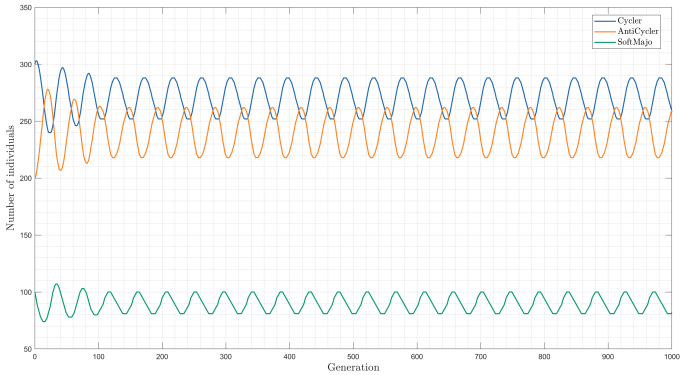
Figure 3.14: Comparison of results for Fig. 10.2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_10_2.m`), and (c) simulation output (see script `demo_fit_sim_10_2.m`).



(a) MBD original figure

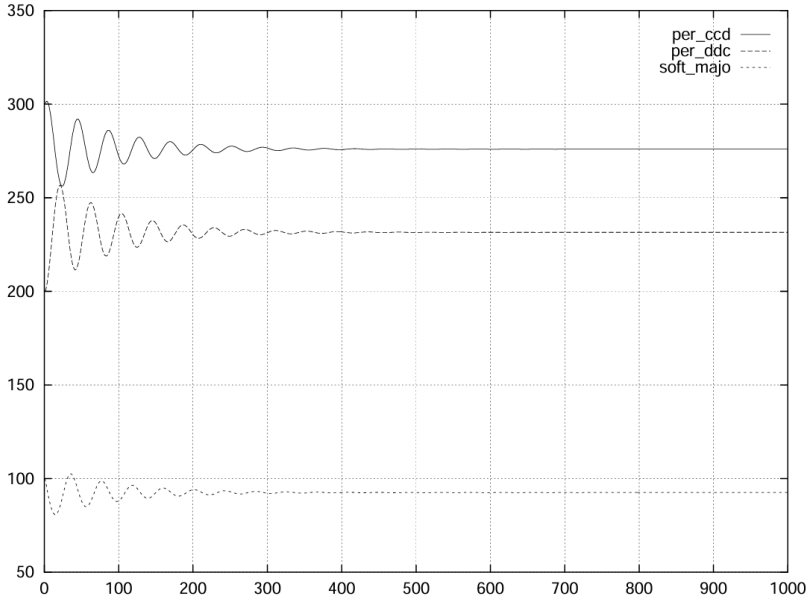


(b) Our theoretical analysis

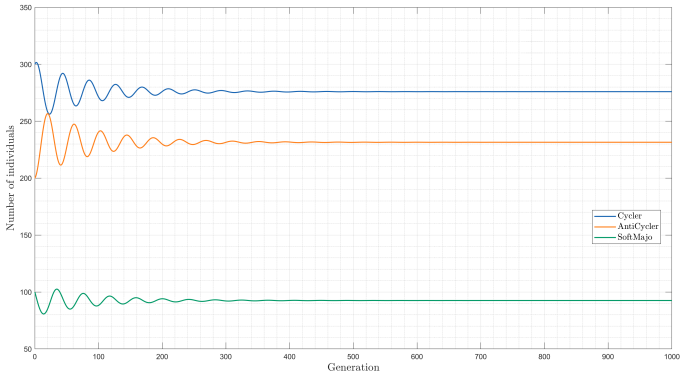


(c) Our simulation result

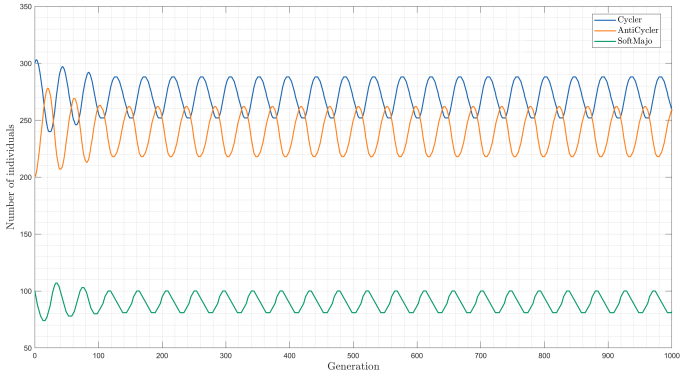
Figure 3.15: Comparison of results for Fig. 11.1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_11_1.m`), and (c) simulation output (see script `demo_fit_sim_11_1.m`).



(a) MBD original figure

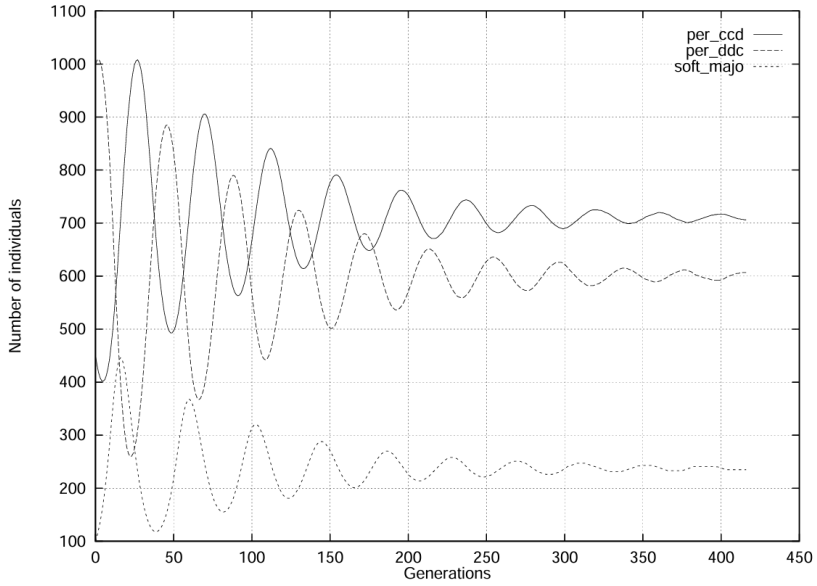


(b) Our theoretical analysis

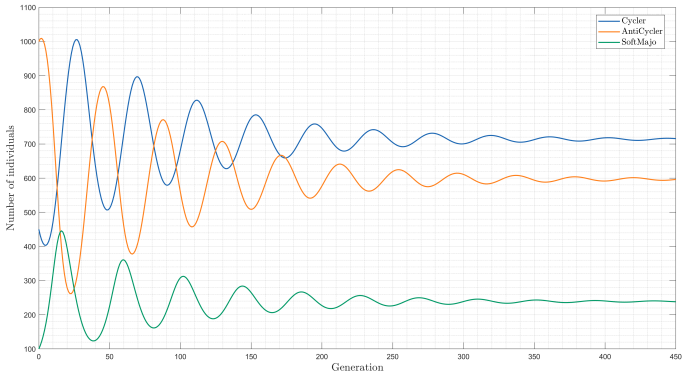


(c) Our simulation result

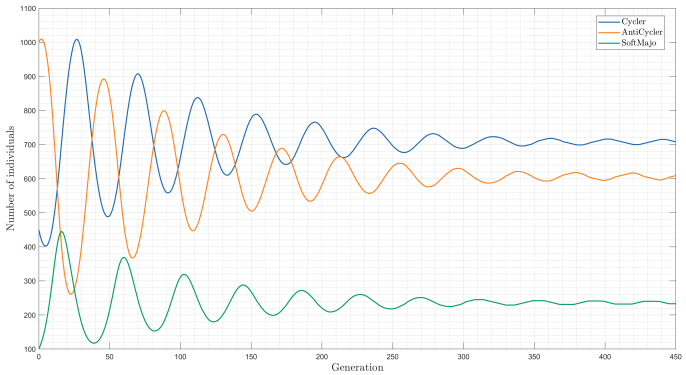
Figure 3.16: Comparison of results for Fig. 11.2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_11_2.m`), and (c) simulation output (see script `demo_fit_sim_11_2.m`).



(a) MBD original figure

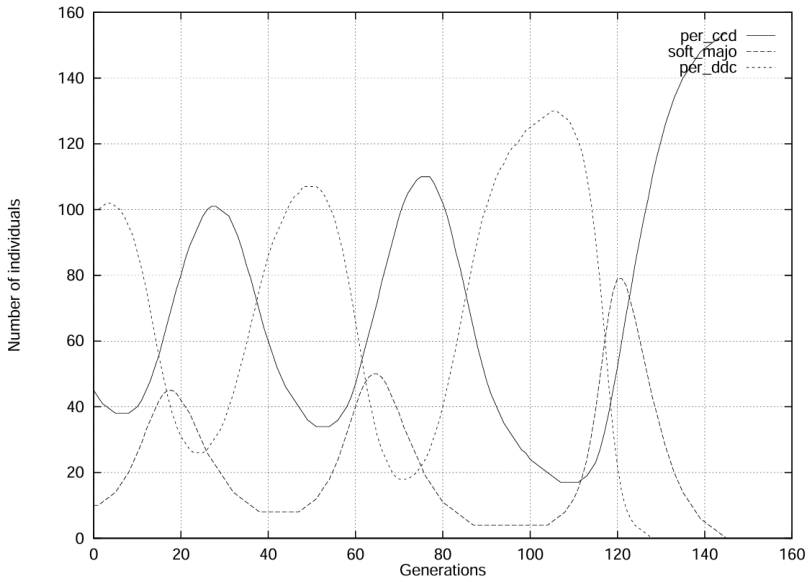


(b) Our theoretical analysis

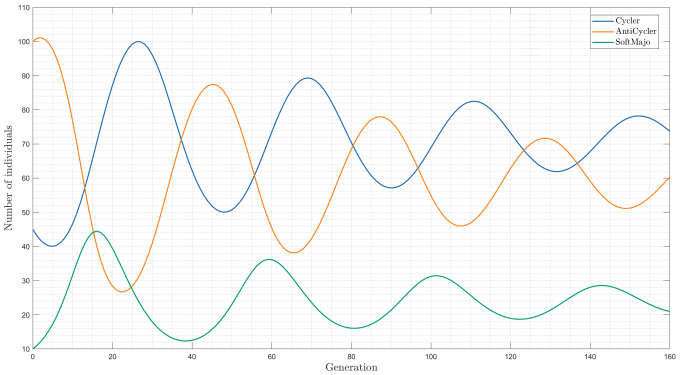


(c) Our simulation result

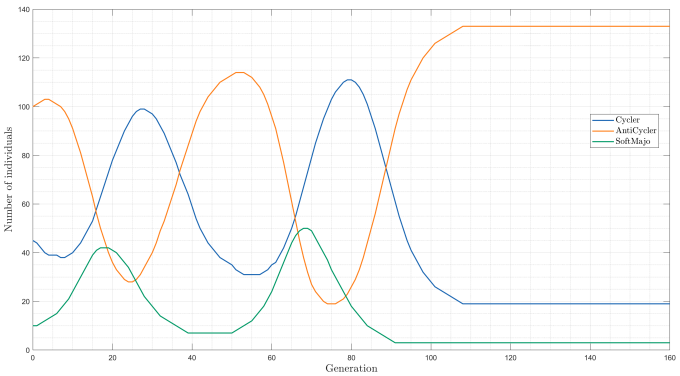
Figure 3.17: Comparison of results for Fig. 12.1 (a) from [4], (b) our theoretical model (see script `demo_fit_the_12_1.m`), and (c) simulation output (see script `demo_fit_sim_12_1.m`).



(a) MBD original figure



(b) Our theoretical analysis



(c) Our simulation result

Figure 3.18: Comparison of results for Fig. 12.2 (a) from [4], (b) our theoretical model (see script `demo_fit_the_12_2.m`), and (c) simulation output (see script `demo_fit_sim_12_2.m`).

Chapter 4

Imitation Dynamics

4.1 The Process

In the context of imitation dynamics, the transition between generations of players occurs through the following process:

1. Each player in the current generation produces exactly one offspring
2. In the new generation, $N - K$ players inherit their parent's strategy while K players adopt one of the top-performing strategies from the previous generation

This generational transition can be interpreted in two equivalent ways:

(a) As *literal* offspring where each generation consists of children of the previous generation's players.

(b) As the *same* set of players who update their strategies in every generation depending on the performance of all strategies.

The "best performing" strategies are those that achieved the highest payoffs in the previous generation. Importantly:

- Strategy performance is generation-dependent - a top strategy in one generation might perform differently when the strategy composition changes.
- Multiple strategies may simultaneously qualify as best performers when their users achieve identical maximum payoffs.
- In such cases of tied performance, strategy-changing players select randomly among the top-performing options.

4.1.1 Theoretical Approach

The *State Transition Matrix* represents the probabilities of moving between different states in a Markov process. In our visualization:

- Rows represent current states
- Columns represent next possible states
- Matrix entries show transition probabilities

This matrix form enables analysis of system dynamics using Markov chain properties, where future states depend only on the current state.

The following pseudo-code explains the process of its visualization.

Algorithm 3 TourTheImi

Function TourTheImi(B, Strategies$[1..M]$, POP0$[1..M]$, K, T, J)
$N \leftarrow \sum$ POP0
$M \leftarrow$ number of strategies
states $\leftarrow$ all possible population vectors summing to N
$S \leftarrow$ number of rows in states
$P \leftarrow$ zero matrix of size $S \times S$ ▷ Transition-probability matrix
for i from 1 to S **do**
    $s \leftarrow$ states$[i]$ ▷ Current population state
    payoffs $\leftarrow$ ComputePayoffs(B, s, Strategies, T)
    best_strats $\leftarrow$ indices of strategies achieving max(payoffs)
    for each *from* such that $s[from] > 0$ **do**
        for each *to* in best_strats **do**
            $s_{\text{new}} \leftarrow s$
            $s_{\text{new}}[from] \leftarrow s_{\text{new}}[from] - 1$
            $s_{\text{new}}[to] \leftarrow s_{\text{new}}[to] + 1$
            $j \leftarrow$ index of s_{new} in states
            ▷ Probability of transition: select individual from 'from' and imitate 'to'
            $p \leftarrow \left(\frac{s[from]}{N}\right) \cdot \left(\frac{1}{\text{length(best\_strats)}}\right)$
            $P[i,j] \leftarrow P[i,j] + p$
        end for
    end for
end for
return P

4.1.2 Simulation Approach

In the simulation execution, the expected payoff for each strategy is first computed—using expressions analogous to those in the Fitness Dynamics model. Following this, K players from the set of underperforming strategies (the "losers") are selected to imitate the most successful strategy (the "winner"). This process is illustrated in the following pseudocode:

Algorithm 4 TourSimImi

Function TourSimImi(B, Strategies[1..S], POP0[1..S], K, T, J)
$S \leftarrow$ number of strategies
POP[0] $\leftarrow$ POP0
BST $\leftarrow$ empty list of length J
$V \leftarrow$ payoff matrix between all strategies
for j from 1 to J **do**
    $W \leftarrow$ POP[j]
                                ▷ Compute expected payoff for each strategy
    for x from 1 to S **do**
        $g[x] \leftarrow \sum_{y=1}^{S} W[y] \cdot V[x, y]$
    end for
    best $\leftarrow$ any index achieving max(g)
    $W_{\text{next}} \leftarrow W$
    $k_{\text{remaining}} \leftarrow K$
    while $k_{\text{remaining}} > 0$ **do**
        Candidates $\leftarrow \{i \mid i \neq \text{best} \wedge W_{\text{next}}[i] > 0\}$
        if Candidates is empty **then**
            break
        end if
        fromIndex $\leftarrow$ random element from Candidates
        W_{next}[fromIndex] $\leftarrow W_{\text{next}}$[fromIndex] $- 1$
        $k_{\text{remaining}} \leftarrow k_{\text{remaining}} - 1$
    end while
    W_{next}[best] $\leftarrow W_{\text{next}}$[best] $+ (K - k_{\text{remaining}})$
    POP[j] $\leftarrow W_{\text{next}}$
    BST[j] $\leftarrow$ best
end for
return POP, BST

4.2 Experiments

4.2.1 Simulation Figures

In Figures 4.1 – 4.12, we explore the Imitation Evolution Dynamics with changes to:

- the *pool of available strategies*
- the *Reward R* and the *Punishment P* of the payoff matrix
- the *number of players K* "mimicking" the winning strategy in each generation
- the initial *Population* of each strategy

In Figure 4.1, an initial simulation is created, in order to make comparisons.

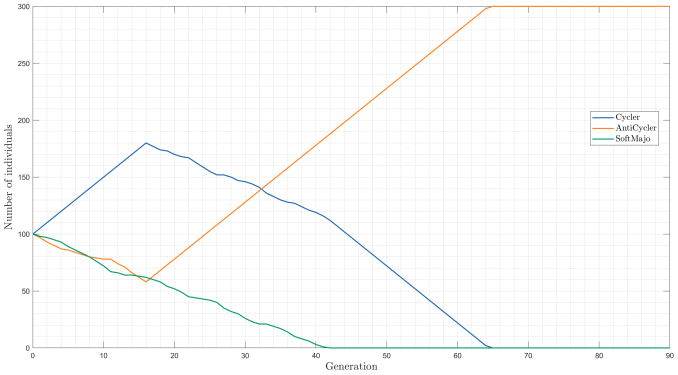


Figure 4.1: See script `demo_imi_sim_1.m`

In Figure 4.2, a monotonous convergence is observed, with change of strategies.

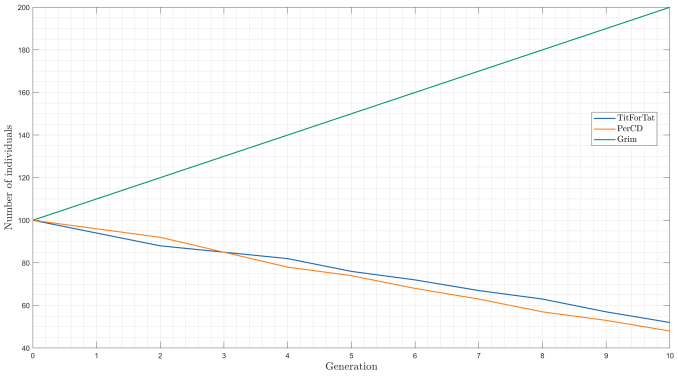


Figure 4.2: See script `demo_imi_sim_2.m`

In Figure 4.5, compared to Figure 4.1, the *Reward* R for mutual cooperation is bigger ($R' = 4$, instead of $R = 3$), while still abiding to the CIPD constraints (see Introduction:

- $T > R > P > S$
- $S + T < 2R$

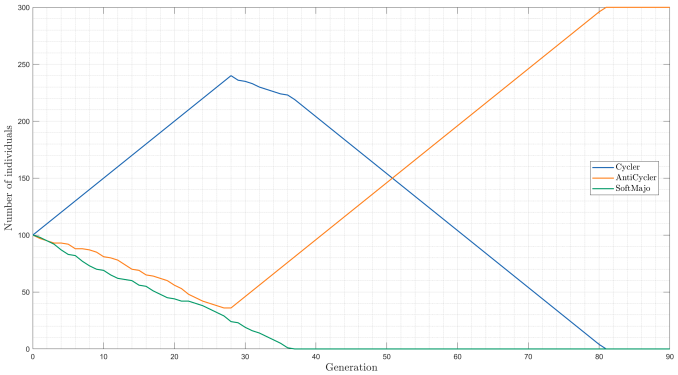


Figure 4.3: See script `demo_imi_sim_3.m`

In Figure 4.4, compared to Figure 4.1, more players change strategy between generations, which, as expected, accelerates the process.

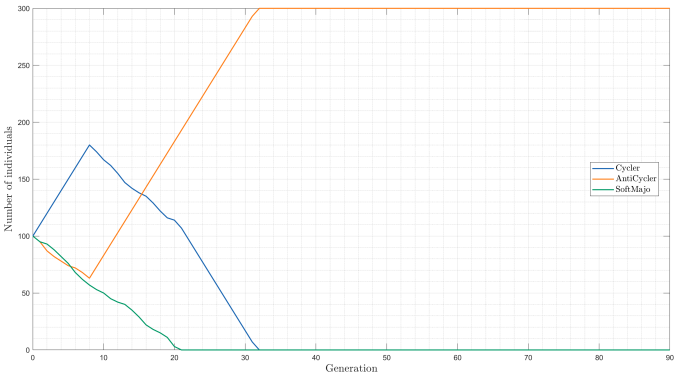


Figure 4.4: See script `demo_imi_sim_4.m`

In Figures 4.5 – 6, the *RewardR* for mutual cooperation increases.

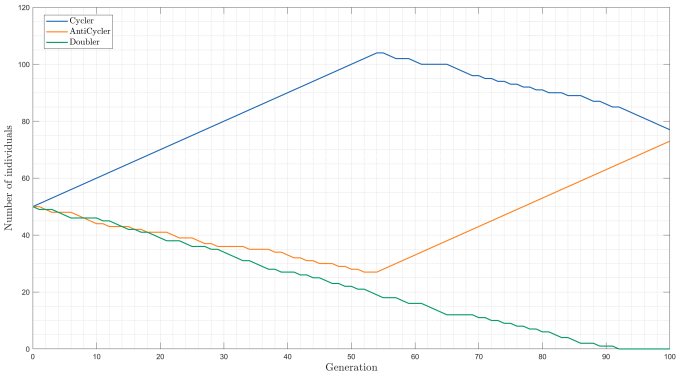


Figure 4.5: See script `demo_imi_sim_5.m`

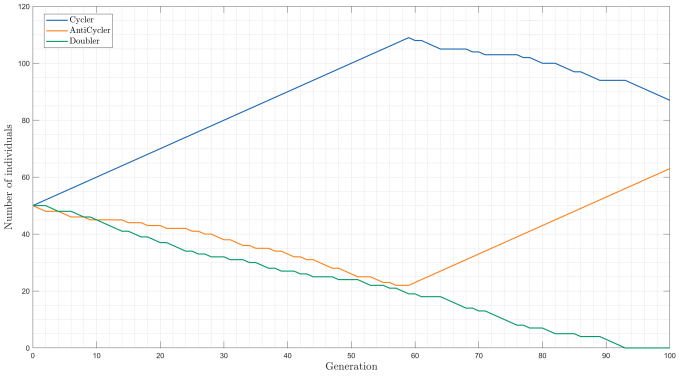


Figure 4.6: See script `demo_imi_sim_6.m`

In Figures 4.7 – 8, as the value of K decreases (from Figure 4.7 to Figure 4.8), not all individuals in the population ultimately adopt the LevelPunisher strategy.

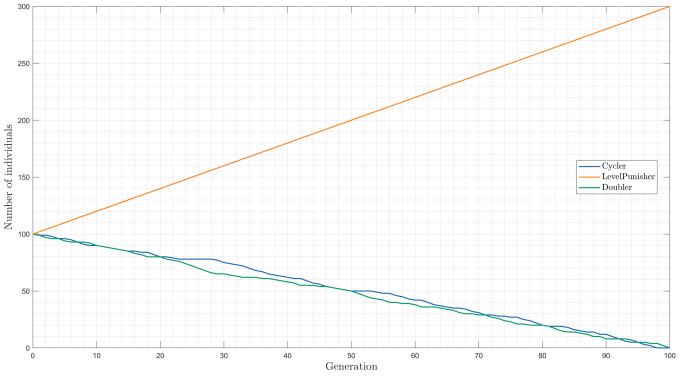


Figure 4.7: See script `demo_imi_sim_7.m`

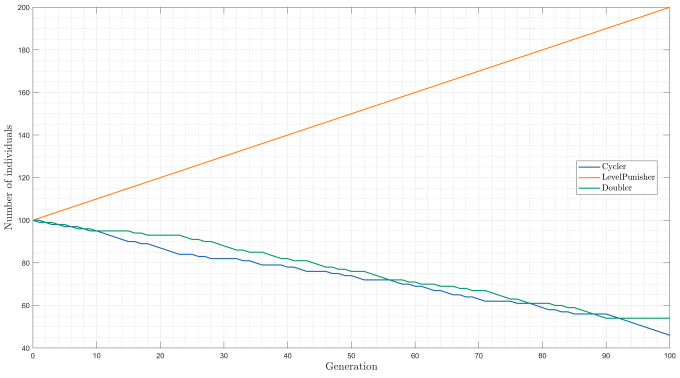


Figure 4.8: See script `demo_imi_sim_8.m`

In Figure 4.9, oscillations are observed, for a different setting of available strategies and initial population distribution.

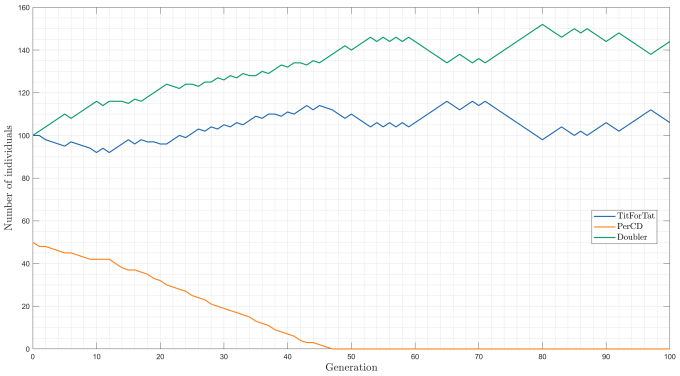


Figure 4.9: See script `demo_imi_sim_9.m`

In Figure 4.10, oscillations are again observed, but this time *TFT* takes advantage of the increase in the number of *PerCD* players.

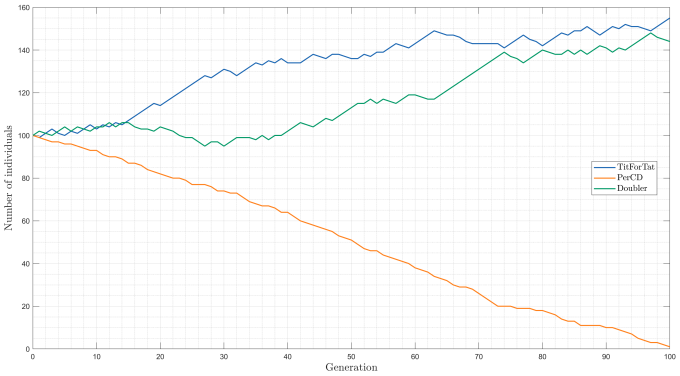


Figure 4.10: See script `demo_imi_sim_10.m`

In Figures 4.11 – 12, the *Punishment P* for mutual defection becomes stronger (that is, a a smaller value of P)

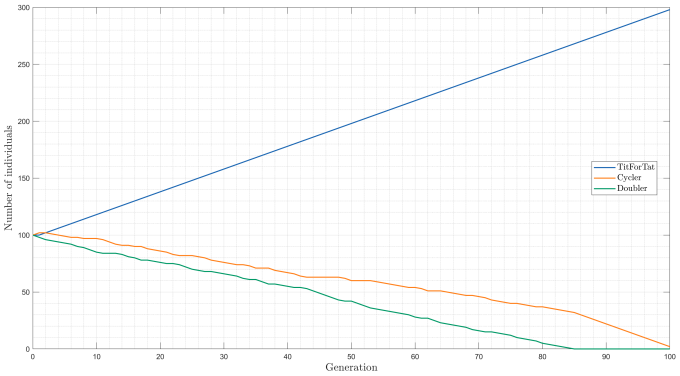


Figure 4.11: See script `demo_imi_sim_11.m`

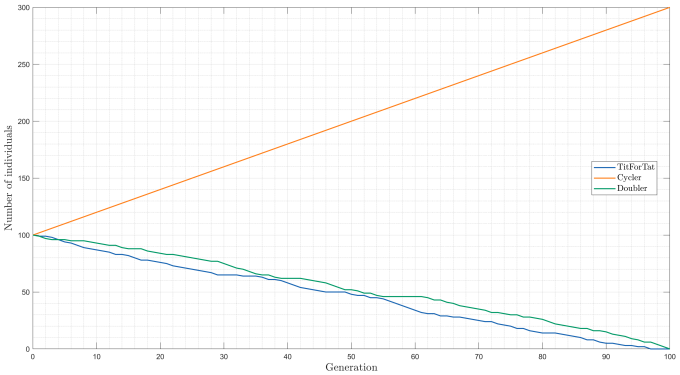


Figure 4.12: See script `demo_imi_sim_12.m`

4.2.2 Markov Chains

In Figures 4.13 – 4.22, the *State Transition Matrix* is visualized in the context of Markovian Analysis. Here, the parameter R, representing mutual cooperation, is gradually increased for $R \in [3, 5)$, and the *Markov Chains* are depicted.

3 Players

For the simple case of $N = 3$ players, we have the 10 states:

$$S = (0,0,3),(0,1,2),(0,2,1),(0,3,0),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)$$

Here, the parameter R, representing mutual cooperation, is gradually increased. Markov chains remain unchanged for $R \in [3, 4]$, indicating no observable behavioral shift. For illustration:

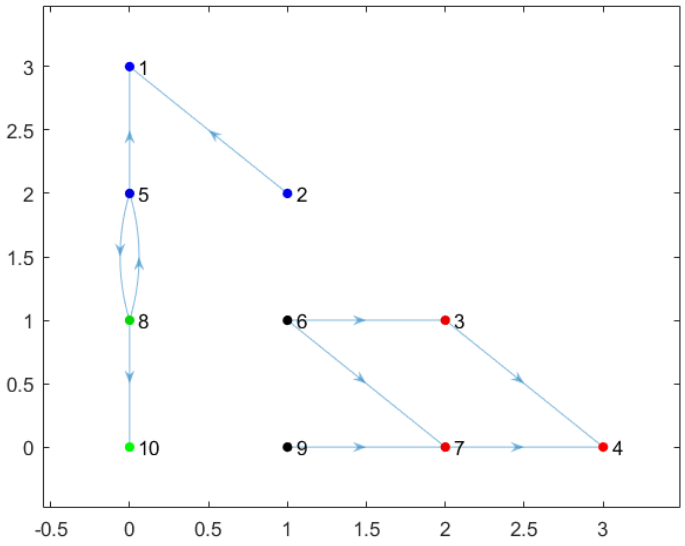


Figure 4.13: See script `demo_imi_the_3Players_1.m`

5 Players

For the case of $N = 5$ players, we have the 21 states. Here, a change in the *Markov Chain* is observed for the values of R: $R = 3$ (Figure 4.14), $R = 3,5$ (Figure 4.15) and $R = 3.75$ (Figure 4.16). The Markov chain appears to split into two disconnected regions.

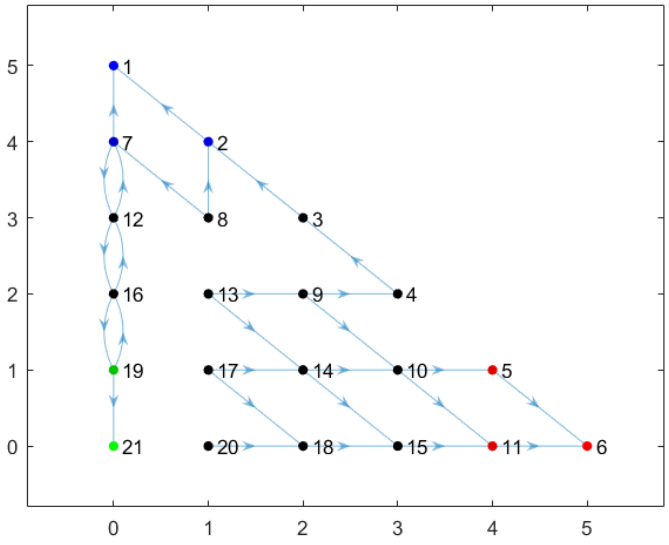


Figure 4.14: See script `demo_imi_the_5Players_1.m`

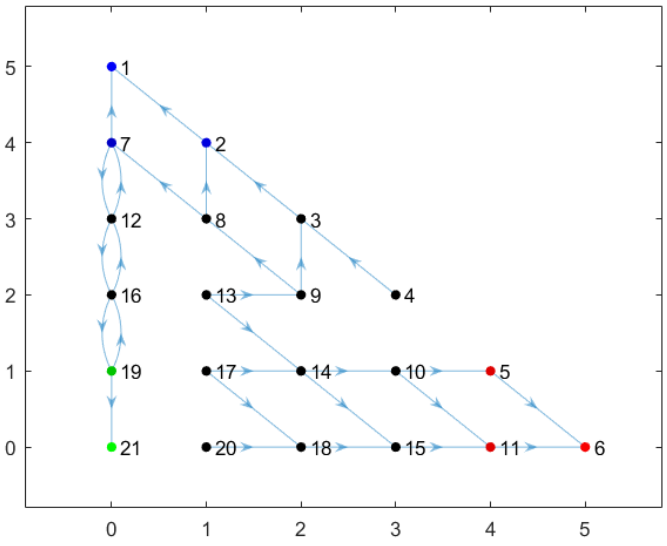


Figure 4.15: See script `demo_imi_the_5Players_3.m`

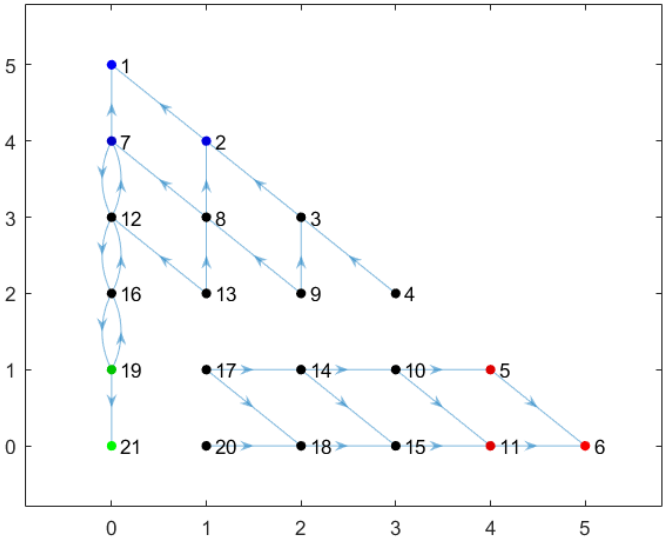


Figure 4.16: See script `demo_imi_the_5Players_4.m`

10 Players

For the case of $N = 5$ players, we have 66 states. Here, a change in the *Markov Chain* is observed for every value of R tested. This time, the Markov chain appears to remain connected (that is there is always a "road" connecting all states), only up until the value $R = 4.9$ (Figure 4.24)

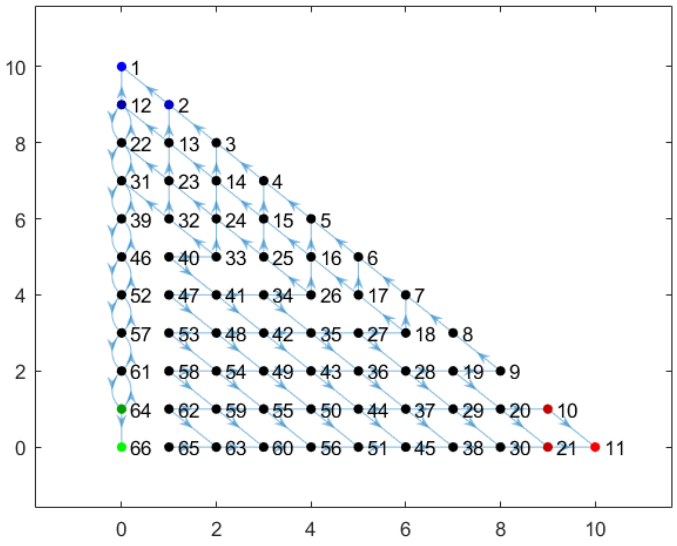


Figure 4.17: See script `demo_imi_the_10Players_1.m`

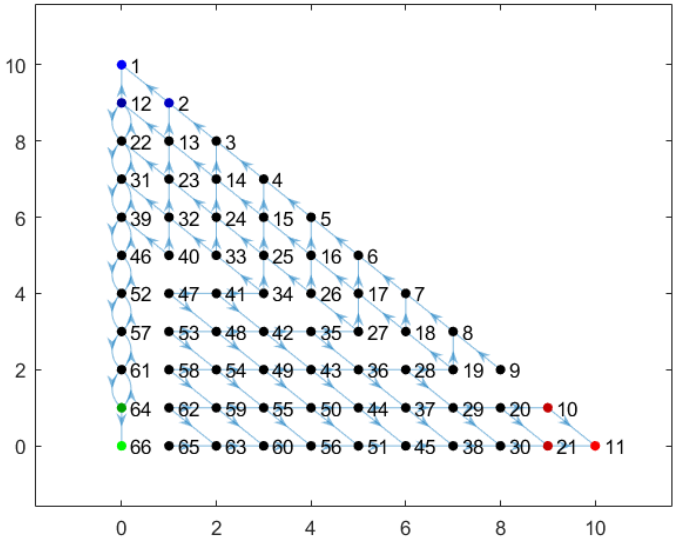


Figure 4.18: See script `demo_imi_the_10Players_2.m`

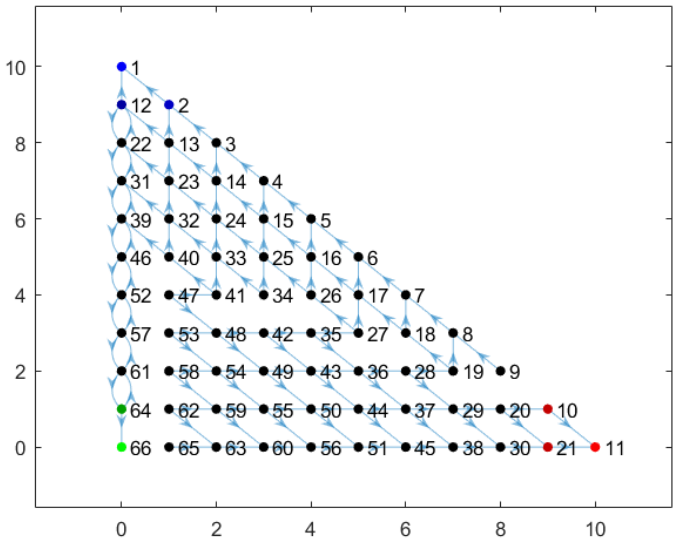


Figure 4.19: See script `demo_imi_the_10Players_3.m`

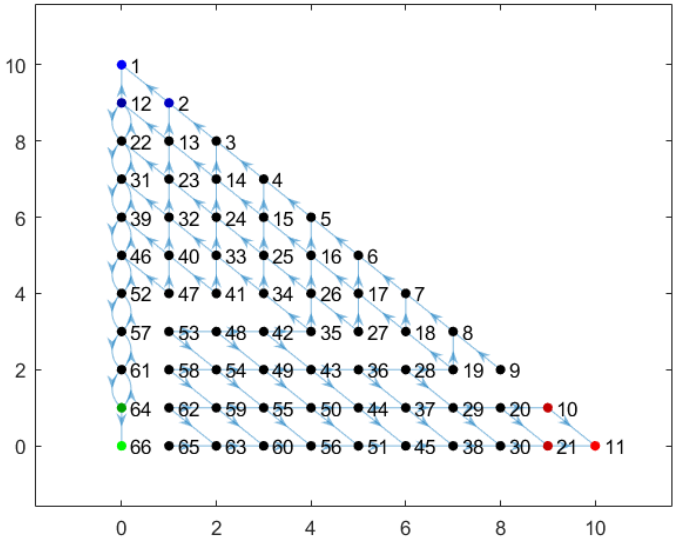


Figure 4.20: See script `demo_imi_the_10Players_4.m`

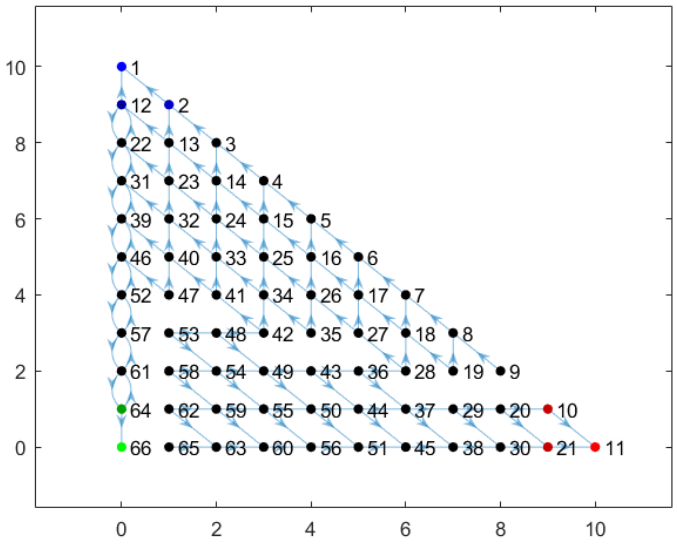


Figure 4.21: See script `demo_imi_the_10Players_5.m`

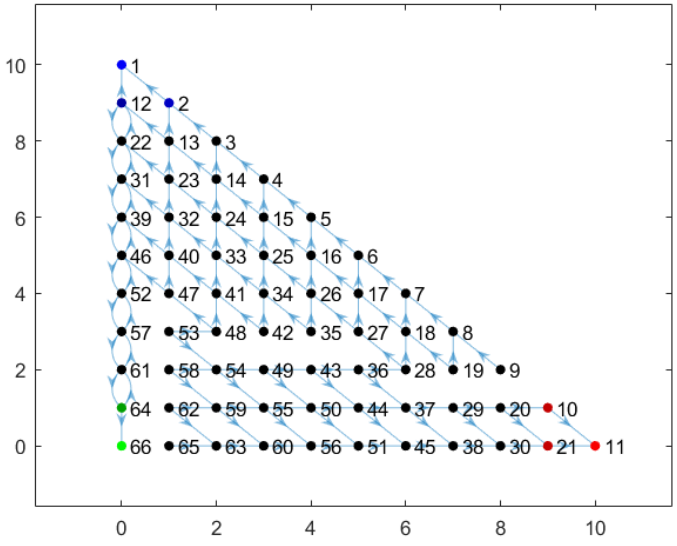


Figure 4.22: See script `demo_imi_the_10Players_6.m`

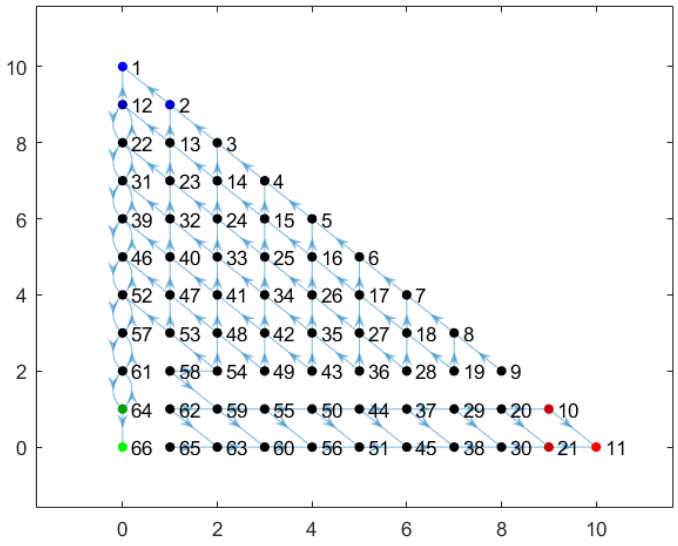


Figure 4.23: See script `demo_imi_the_10Players_7.m`

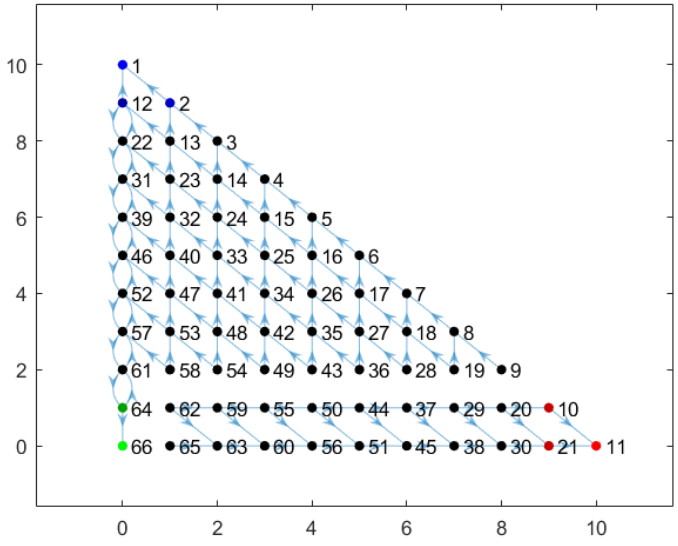


Figure 4.24: See script `demo_imi_the_10Players_8.m`

Note: for more Figures and Examples, see the "Examples" and the "Report/Figures" directories in

`https://github.com/inkonstant/EvolutionaryGamesToolbox`

Chapter 5

Discussion

Our comparative analysis of fitness and imitation dynamics reveals fundamental differences in how populations evolve under these two selection mechanisms. Both approaches share the common framework of the Iterated Prisoner's Dilemma but demonstrate distinct behavioral patterns and convergence properties.

5.1 Key Differences

- **Selection Pressure:**
  - Fitness dynamics exhibit gradual proportional changes, where strategies grow according to relative performance
  - Imitation dynamics feature punctuated transitions, with K players abruptly switching to top performers
- **Convergence Speed:**
  - Fitness dynamics show smoother, slower convergence (Figs. 3.1-3.18)
  - Imitation dynamics reach stable states faster but with more variance (Figs. 4.1-4.12)
- **Population Representation:**
  - Fitness dynamics maintain all strategies at non-zero proportions longer
  - Imitation dynamics tend to eliminate intermediate performers quickly

5.2 Practical Implications

- **Fitness Dynamics** better model:
  - Biological evolution with reproductive success metrics
  - Economic systems with market share dynamics
- **Imitation Dynamics** better capture:
  - Social learning processes

- Technology adoption scenarios

These differences suggest that the choice of dynamics should align with the modeled system's characteristics. Fitness dynamics provide more nuanced evolutionary trajectories, while imitation dynamics better reflect systems where agents consciously adopt successful behaviors.

Appendix A

Documentation

The Documentation is available at:

https://github.com/inkonstant/EvolutionaryGamesToolbox/tree/main/Documentation

Appendix B

Github Repo

The Github Repo is available at:

`https://github.com/inkonstant/EvolutionaryGamesToolbox`

Bibliography

[1] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *Science*, vol. 211, no. 4489, pp. 1390–1396, 1981.

[2] J. M. Alexander, *Evolutionary Game Theory*. Cambridge University Press, 2023.

[3] R. Axelrod and D. Dion, "The further evolution of cooperation," *Science*, vol. 242, no. 4884, pp. 1385–1390, 1988.

[4] P. Mathieu, B. Beaufils, and J.-P. Delahaye, "Studies on dynamics in the classical iterated prisoner's dilemma with few strategies," in *European Conference on Artificial Evolution*. Berlin Heidelberg: Springer, 1999.