Aristotle University of Thessaloniki
Group Project for the Game Theory Course
Group 01

Imitation and Fitness Dynamics on Iterated Prisoner's Dilemma

Ιωάννης Λαζαρίδης
10431
ilazarit@ece.auth.gr

Κωνσταντίνος Βλαχάκος
10403
kvlachak@ece.auth.gr

Ιωάννης Κοσμάς Βαρδάκης
10498
ivardakis@ece.auth.gr

May 2025

Contents

Κεφάλαιο 0

Εισαγωγή

Η παρούσα εργασία πραγματεύεται την υλοποίηση του εξελικτικού πρωταθλήματος *Axelrod*, το οποίο έχει ως θεωρητική βάση το επαναλαμβανόμενο δίλημμα του φυλακισμένου (Iterated Prisoner's Dilemma - IPD). Η εργασία αυτή αποτελεί το πέμπτο παραδοτέο στο πλαίσιο του μαθήματος της Θεωρίας Παιγνίων, και ακολουθεί προηγούμενες εργασίες στις οποίες έχουν υλοποιηθεί:(α) το περιβάλλον του παιχνιδιού,(β) οι βασικές στρατηγικές που χρησιμοποιούνται στα πειράματα. Για την καλύτερη κατανόηση, είναι σημαντικό να εξηγηθεί πρώτα τι είναι το δίλημμα του φυλακισμένου και το πρωτάθλημα Axelrod.

Το Δίλημμα του Φυλακισμένου

Το κλασικό δίλημμα του φυλακισμένου είναι ένα υποθετικό σενάριο συνεργασίας και προδοσίας μεταξύ δύο παικτών. Κάθε παίκτης έχει δύο επιλογές: να συνεργαστεί (**C**) ή να προδώσει (**D**). Οι αποδόσεις (**payoffs**) κάθε παίκτη εξαρτώνται από τον συνδυασμό των επιλογών τους και μπορούν να περιγραφούν με τέσσερις βασικές τιμές:

- **T (Temptation to Defect)** η ανταμοιβή που παίρνει κάποιος όταν προδίδει ενώ ο άλλος συνεργάζεται.
- **R (Reward for Mutual Cooperation)**: η ανταμοιβή που παίρνουν και οι δύο όταν συνεργάζονται.
- **P (Punishment for Mutual Defection)**: η τιμωρία που λαμβάνουν όταν και οι δύο προδίδουν.
- **S (Sucker's Payoff)**: η "ζημιά" που υφίσταται κάποιος όταν συνεργάζεται ενώ ο άλλος τον προδίδει

Ο πίνακας αποδόσεων (*payoff matrix*) έχει ως εξής:

		P2	
		C (Συνεργασία)	D (Προδοσία)
P1	C	R, R	S, T
	D	T, S	P, P

Table 1: Payoff Matrix

Οι τυπικές τιμές που χρησιμοποιούνται στο IPD είναι: $T = 5$, $R = 3$, $P = 1$, $S = 0$

Για να υπάρχει πραγματικό «δίλημμα», πρέπει να ισχύουν δύο βασικές **λογικές συνθήκες:**

1. $\boldsymbol{S < P < R < T}$
   Δηλαδή: είναι χειρότερο να σε προδώσουν ενώ συνεργάζεσαι (S), από το να προδώσετε και οι δύο (P), και προτιμότερο να συνεργαστείτε και οι δύο (R), αλλά ο μεγαλύτερος πειρασμός είναι να προδώσεις όταν ο άλλος συνεργάζεται (T).

2. $\boldsymbol{S + T < 2R}$
   Αυτή η επιπλέον συνθήκη εξασφαλίζει ότι η **συνεργασία είναι συλλογικά προτιμότερη** από την εναλλαγή μεταξύ προδοσίας και συνεργασίας. Δηλαδή, αν οι παίκτες εναλλάσσονται μεταξύ C και D, οι αποδόσεις είναι χειρότερες από τη σταθερή συνεργασία.

   Το δίλημμα έγκειται στο γεγονός ότι, ενώ η **αμοιβαία συνεργασία** οδηγεί σε καλύτερη συνολική απόδοση και για τους δύο (R), η **προδοσία είναι η ατομικά καλύτερη επιλογή**, ανεξάρτητα από την κίνηση του άλλου. Αυτό συμβαίνει επειδή:

   - Αν ο άλλος συνεργάσει, το να προδώσεις σου δίνει $\boldsymbol{T > R}$
   - Αν ο άλλος προδώσει, το να προδώσεις σου δίνει $\boldsymbol{P > S}$

   Έτσι, η προδοσία είναι **το μοναδικό σημείο ισορροπίας Nash** στο παίγνιο μίας φοράς. Ωστόσο, όπως θα δούμε στη συνέχεια, όταν το παιχνίδι επαναλαμβάνεται, μπορούν να προκύψουν πολύ διαφορετικές δυναμικές.

Επαναλαμβανόμενο Δίλημμα του Φυλακισμένου (IPD)

Επαναλαμβανόμενο Δίλημμα του Φυλακισμένου (Iterated Prisoner's Dilemma)

Σε αντίθεση με το απλό παίγνιο που παίζεται μόνο μία φορά, **στο επαναλαμβανόμενο δίλημμα του φυλακισμένου**, οι δύο παίκτες παίζουν πολλές φορές μεταξύ τους, και μπορούν να βασίσουν την απόφασή τους για την επόμενη κίνηση στις προηγούμενες κινήσεις του αντιπάλου. Αυτό επιτρέπει την εμφάνιση πιο πολύπλοκων στρατηγικών, όπως η **Tit for Tat** (ξεκινά με συνεργασία και έπειτα αντιγράφει την προηγούμενη κίνηση του αντιπάλου), και δημιουργεί δυναμική αλληλεπίδρασης και «μνήμης».

Το Πρωτάθλημα Axelrod

Το **Πρωτάθλημα Axelrod** ήταν ένα διάσημο υπολογιστικό πείραμα που διεξήγαγε ο πολιτικός επιστήμονας **Robert Axelrod** τη δεκαετία του 1980. Ο Axelrod προσκάλεσε ερευνητές να υποβάλουν στρατηγικές για το επαναλαμβανόμενο δίλημμα του φυλακισμένου. Οι στρατηγικές αυτές διαγωνίστηκαν μεταξύ τους σε ένα τουρνουά όπου η κάθε στρατηγική έπαιζε επαναλαμβανόμενα παιχνίδια με όλες τις υπόλοιπες, συγκεντρώνοντας βαθμούς με βάση τα αποτελέσματα κάθε αναμέτρησης. Το τουρνουά του Axelrod ανέδειξε τη σημασία της συνεργασίας και της «τιμωρίας με μέτρο» ως αποτελεσματική στρατηγική. Χαρακτηριστικά, η απλή στρατηγική **Tit for Tat** αναδείχθηκε ως μία από τις πιο επιτυχημένες, αναδεικνύοντας τη δύναμη της αμοιβαιότητας στη σταθερή συνεργασία.

Σκοπός και Δομή της Παρούσας Εργασίας

Η εργασία επικεντρώνεται στην **εξελικτική εκδοχή** του τουρνουά, όπου οι στρατηγικές «επιβιώνουν» ή «αντικαθίστανται» ανάλογα με την απόδοσή τους. Αναλυτικότερα, η εργασία χωρίζεται σε δύο βασικές προσεγγίσεις

1. **Imitation Dynamics**
   Σε αυτό το μοντέλο, σε κάθε γενιά ένας αριθμός παικτών αντικαθιστά τη στρατηγική του με εκείνη κάποιου άλλου που έχει καλύτερη απόδοση (Score). Ο πληθυσμός εξελίσσεται μιμούμενος τις καλύτερες στρατηγικές.

2. **Fitness Dynamics**: Σε αυτή την προσέγγιση, η πιθανότητα να επιβιώσει ή να αναπαραχθεί μια στρατηγική εξαρτάται από το μέσο Score που συγκεντρώνει. Πρόκειται για πιο αναλυτική, συνεχής προσέγγιση εμπνευσμένη από τη βιολογική εξέλιξη.

Η βασική διαφορά μεταξύ των δύο μοντέλων έγκειται στον τρόπο που οι στρατηγικές διαδίδονται στον πληθυσμό:

- Στο **Fitness Dynamics**, η αλλαγή εξαρτάται από το **συνολικό αποτέλεσμα** κάθε στρατηγικής σε κάθε γενιά.

- Στο **Imitation Dynamics**, επικεντρωνόμαστε στους **καλύτερους παίκτες**, και οι αλλαγές είναι διακριτές και συμβαίνουν σε **συγκεκριμένο αριθμό** παικτών ανά γενιά.

Στη θεωρητική ανάλυση του Fitness Dynamics, επιβεβαιώνονται τα αποτελέσματα των [2] Mathieu et al., ενώ στο Imitation Dynamics γίνεται υλοποίηση της **αλυσίδας Markov** για όλες τις πιθανές κατανομές πληθυσμού, ανάλογα με το μέγεθος πληθυσμού και τη δυναμική μετάβασης.

Κεφάλαιο 1

Quickstart

Τα αρχεία μας βρίσκονται στο παρακάτω github repo : για την κλήση οποιουδήποτε πειράματος χρησιμοποιούμε το script `base.m` με την οποία ο χρήστης μπορεί να αναπαραγάγει τα πειράματα που έχουμε κάνει και εμείς. Σε γενικότερο πλαίσιο, ο κώδικάς μας αναπαράγει τα αποτελέσματα από το paper[2] αλλα και να κάνει πειράματα προσομοίωσης πάνω σε αυτό. Επιπλέον μπορεί να αναπαράξει τόσο αποτελέσματα θεωρητικής ανάλυσης, τόσο και αποτελέσματα Imitation, όπως αναλύσαμε στο μάθημα. Οι βασικές συναρτήσεις που υλοποιούν τους αλγορίθμους με τους οποίους γίνονται αυτές οι διαδικασίες είναι οι: `TourTheFit`, `TourTheFit2`, `TourSimFit`, `TourTheImi`, `TourTheImi2`, `TourSimImi`.

Για το repository δεν απαιτείται κάποια ειδική εγκατάσταση πέραν του απλού `git clone` καθώς όλα τα paths των scripts είναι relative. Σε περίπτωση απροοπτου προβλήματος μπορείτε να επικοινωνήσετε μαζί μας στα παραπάνω email.

Κεφάλαιο 2

Fitness Dynamics

2.1 Θεωρητική Ανάλυση του Fitness Dynamics

Σε αυτό το σημείο θα αναλύσουμε την `TourTheFit` .

Τα ορίσματα όπως ζητάει η εκφώνηση είναι το B ο πίνακας του παίγνιου, το Strategies που είναι διάνυσμα `strings` με τις στρατηγικές και συμπληρώνεται από το `Pop0` το οποίο είναι ο αρχικός πλυθησμός που παρευρίσκεται στο παιχνίδι. Το T ο αριθμός των γύρων και το J ο αριθμός των γενεών που θα τρέχει το πρωτάθλημα. Αρχικά λαμβάνουμε τον πίνακα `Re_matrix` (από τη συνάρτηση `Reward_str`) ο οποίος είναι ο πίνακας των Scores μετά από έναν συγκεκριμένο αριθμό γύρων.

Η συνάρτηση αυτή υλοποιεί ένα παιχνίδι Axel T γύρων και επιστρέφει τον πίνακα με τα scores μεταξύ των στρατηγικών. Στη συνέχεια, λαμβάνουμε στην μεταβλητή P το άθροισμα των πληθυσμών στις στρατηγικές που παίζουμε δηλαδή τον αρχικό συνολικό πληθυσμό και αρχικοποιούμε τις μεταβλητές επιστροφής.

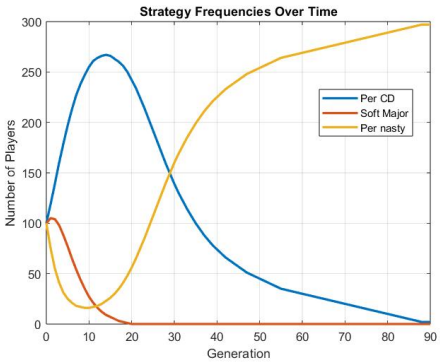
Το `POP` επιστρέφει τις τιμές του πληθυσμού ανά γενιά, το `BST` τις βέλτιστες στρατηγικές ανα γενιά ενώ το `FIT` το fitness των στρατηγικών. Στη συνέχεια αναθέτουμε τον αρχικό πληθυσμό στην πρώτη θέση του array του `POP` και υπολογίζουμε το fitness της κάθε στρατηγικής δηλαδή το άθροισμα των score επί τον πλυθησμό με βάση τη συγκεκριμένη στρατηγική αφαιρώντας τις περιπτώσεις που οι στρατηγικές παίζουν με τον εαυτό τους όπως αναλύεται και στο paper[2].

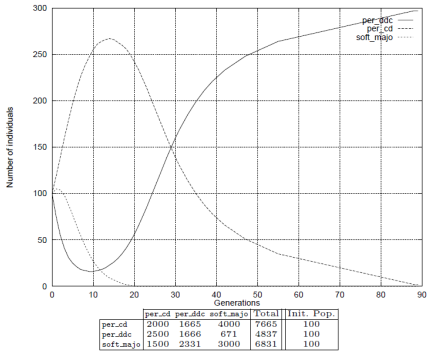
Στη συνέχεια υπολογίζονται οι συνολικοί πόντοι t που έχουν δοθεί σε όλες τις στρατηγικές και ψάχνουμε να βρούμε αυτές με το βέλτιστο fitness για την έξοδο του `BST`.
Ο πληθυσμός ανανεώνεται με βάση τους τύπους του paper[2]:

$$W_{n+1}(A) = \frac{\Pi W_n(A) g_n(A)}{t(n)}$$
$$W_{n+1}(B) = \frac{\Pi W_n(B) g_n(B)}{t(n)}$$
$$W_{n+1}(C) = \frac{\Pi W_n(C) g_n(C)}{t(n)}$$

και το ερώτημα που γεννάται είναι κατά πόσο η ανάλυση αυτή είναι σωστή με την παραδοχή ότι δεν γίνεται στρογγυλοποίηση και το `POP` διάνυσμα περιέχει πραγματικές τιμές. Για τον σκοπό αυτό προσθέτουμε την προτροπή του paper σύμφωνα με το οποίο οι πληθυσμοί της επόμενης γενιάς υπλογίζονται από το floor της κάθε παράστασης. Εδώ πέρα γεννάται το ερώτημα εάν το συγκεκριμένο «επιπόλαιο» rounding κρατήσει ίδιο τον συνολικό πληθυσμό και με την πρώτη δοκιμή παρατηρούμε ότι δεν τον κρατάει. Συνεπώς πρέπει να αποφασίσουμε αν θα χρησιμοποιήσουμε τον ίδιο πληθυσμό με τον αρχικό ή κάθε φορά θα τον ξαναμετράμε για να συμπεριλάβουμε το σφάλμα του rounding. Ωστόσο στη δεύτερη περίπτωση παρατηρούμε το σφάλμα ανά γενιά να συσσωρεύεται οδηγώντας στην κατάρρευση του συστήματος. Συνεπώς η υλοποίηση η οποία ακολουθεί πιστά και το paper είναι να μετρήσουμε μια φορά τον συνολικό πληθυσμό και σε κάθε γενιά να

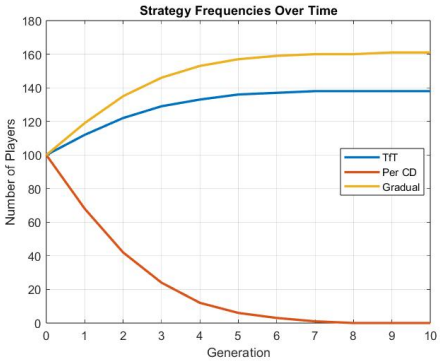
χρησιμοποιούμε αυτή τη σταθερή τιμή P κάνοντας floor rounding ακόμα και αν αυτό οδηγήσει στο τέλος σε μια μικρή απώλεια. Τα αποτελέσματα που παρήχθησαν μπορούν να φανούν παρακάτω:

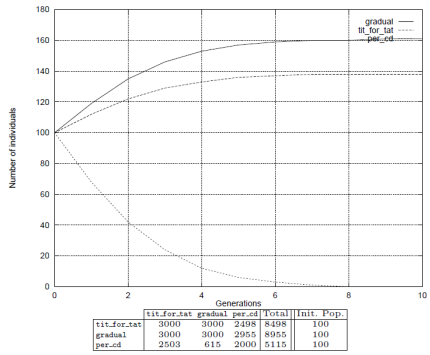




	per_cd	per_ddc	soft_majo	Total	Init. Pop.
per_cd	2000	1665	4000	7665	100
per_ddc	2500	1666	671	4837	100
soft_majo	1500	2331	3000	6831	100

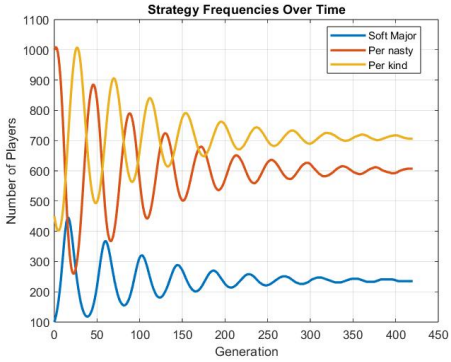
Figure 2.1: Defectors may be Strong (`defectors_may_be_strong.m`)

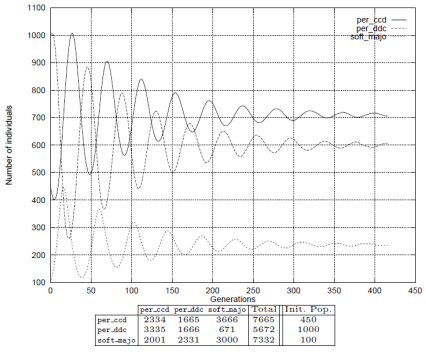




	tit_for_tat	gradual	per_cd	Total	Init. Pop.
tit_for_tat	3000	3000	2498	8498	100
gradual	3000	3000	2955	8955	100
per_cd	2503	615	2000	5115	100

Figure 2.2: Monotonous Convergence (`monotonous_convergence.m`)





	per_ccd	per_ddc	soft_majo	Total	Init. Pop.
per_ccd	2334	1665	3666	7665	450
per_ddc	3335	1666	671	5672	1000
soft_majo	2001	2331	3000	7332	100

Figure 2.3: Attenuated_oscillatory_movements (`attenuated_oscillatory_movements.m`)

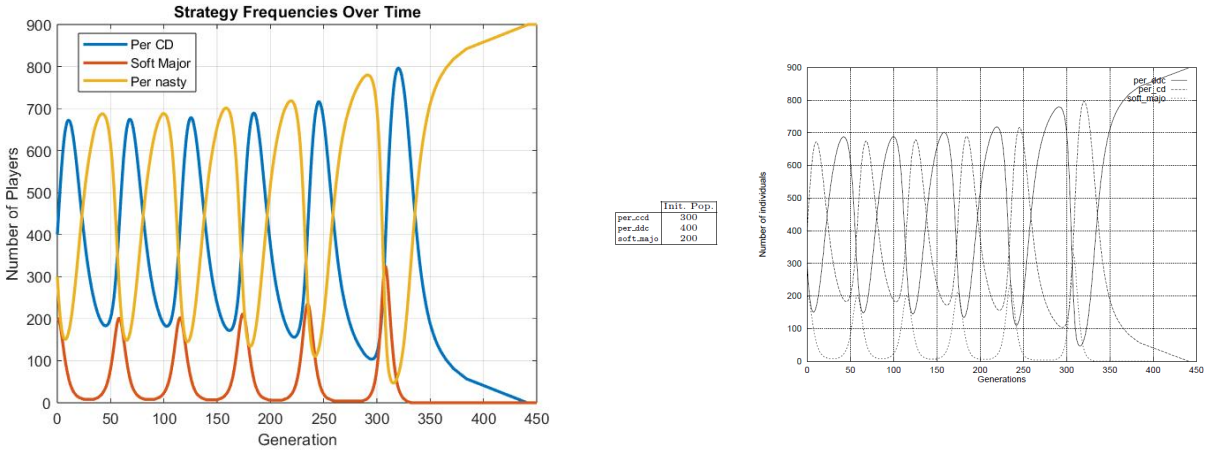


Figure 2.4: Increasing Oscillations (`increasing_oscillations.m`)

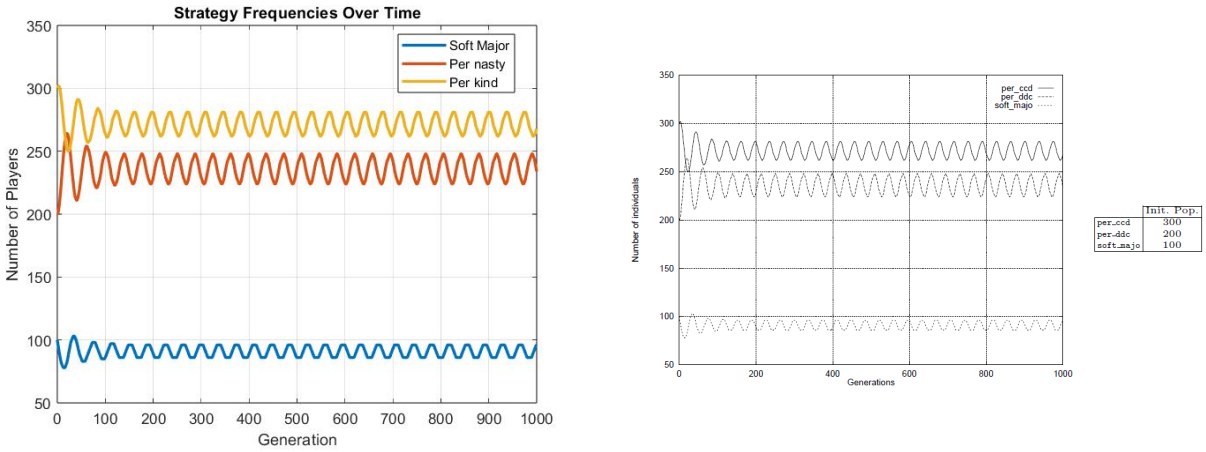


Figure 2.5: Periodic Movements (`periodic_movements.m`)

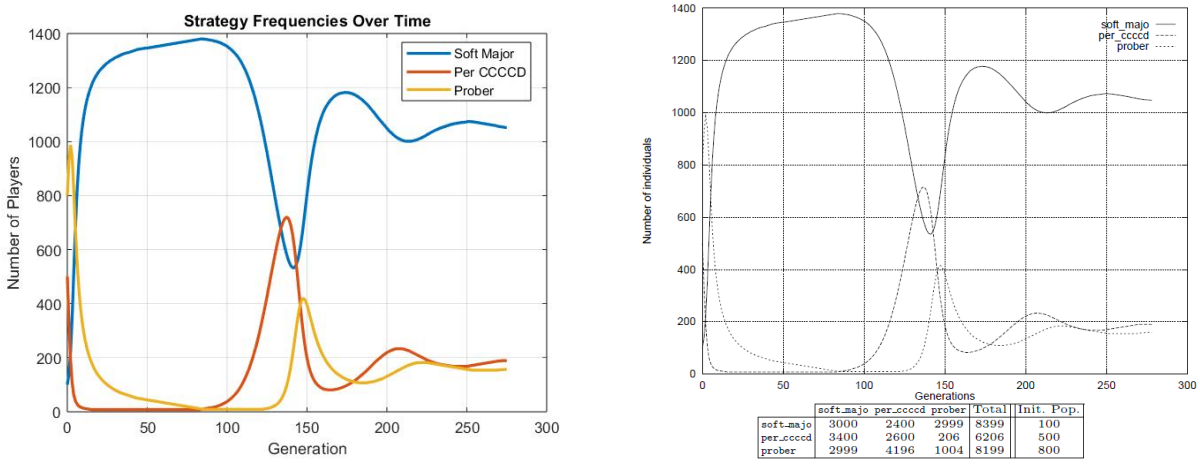


Figure 2.6: Disordered Oscillations (`Disordered Oscillations.m`)

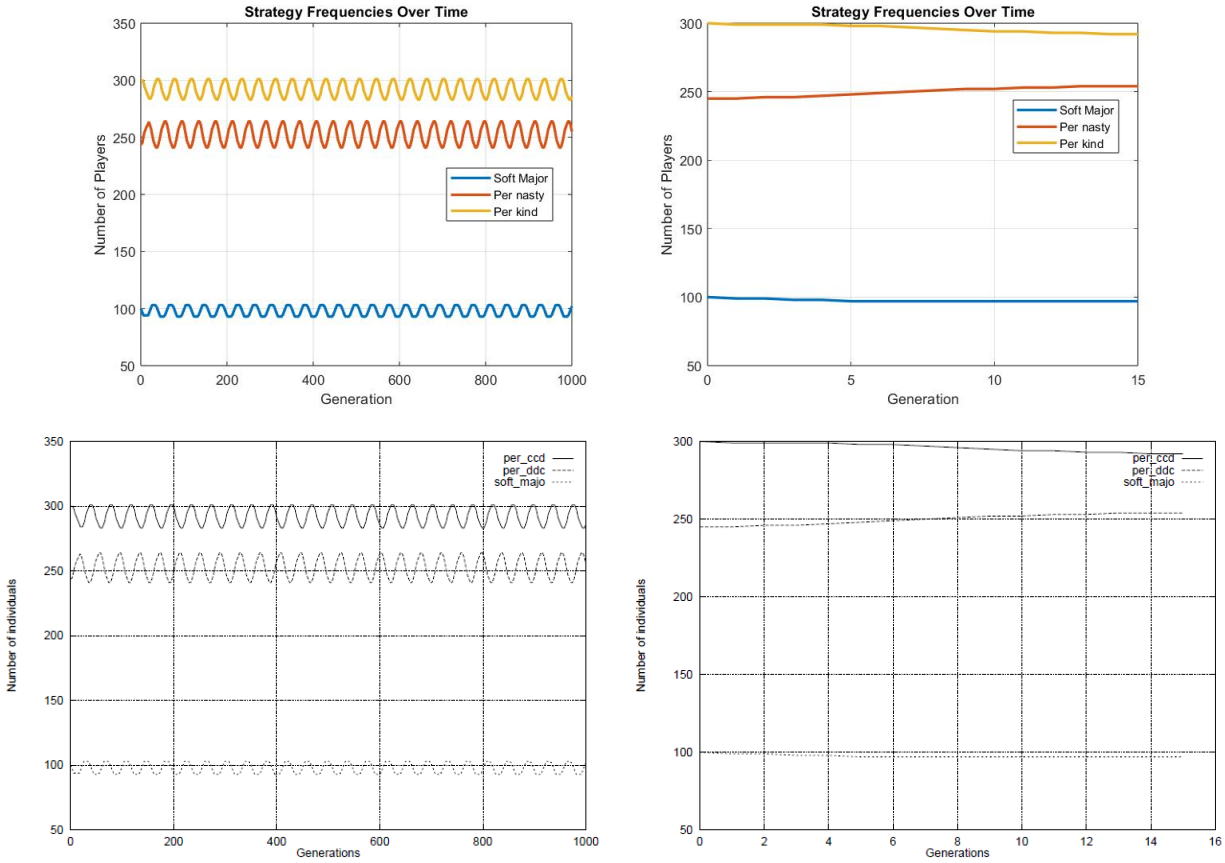


Figure 2.7: Sensitivity of dynamics to population's size. All parameters are identical except for the initial size of per ddc which is 244 on the left and 245 on the right (`sens_dyn_pop_size.m`)

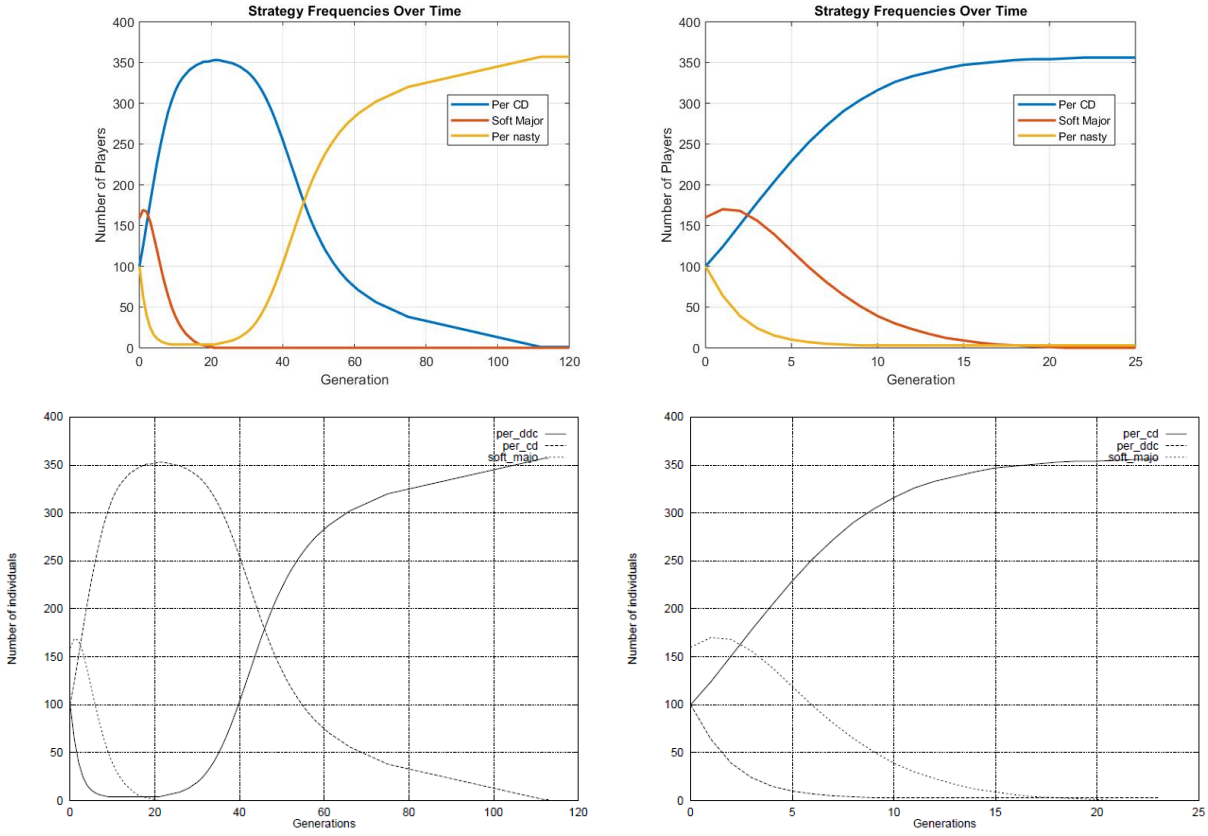


Figure 2.8: Sensitivity of winner to population's size. All parameters are identical except for the initial size of soft majo which is 159 on the left and 160 on the right.(`sens_win_pop_size.m`)

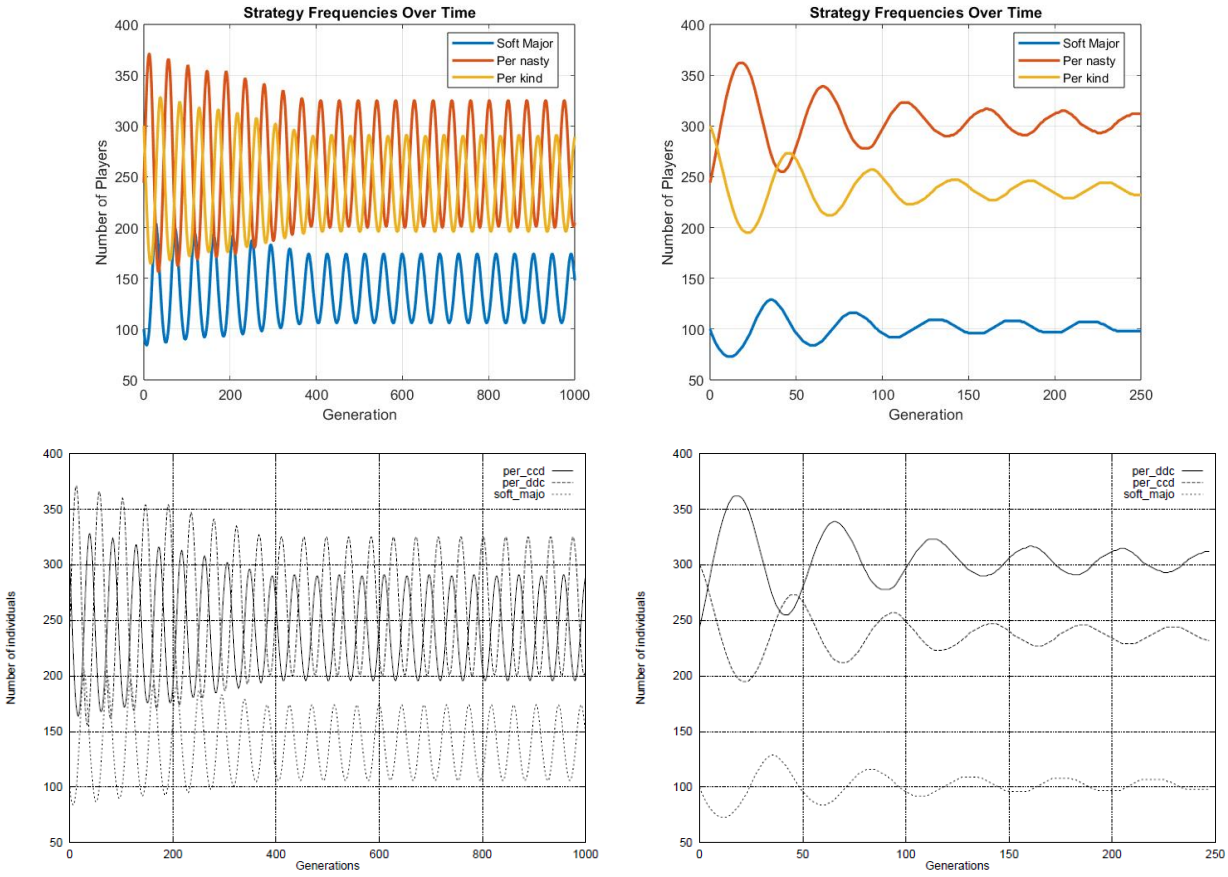


Figure 2.9: Sensitivity to game length. All parameters are identical except for the game length which is 7 moves on the left and 6 on the right `(sens_game_length.m)`

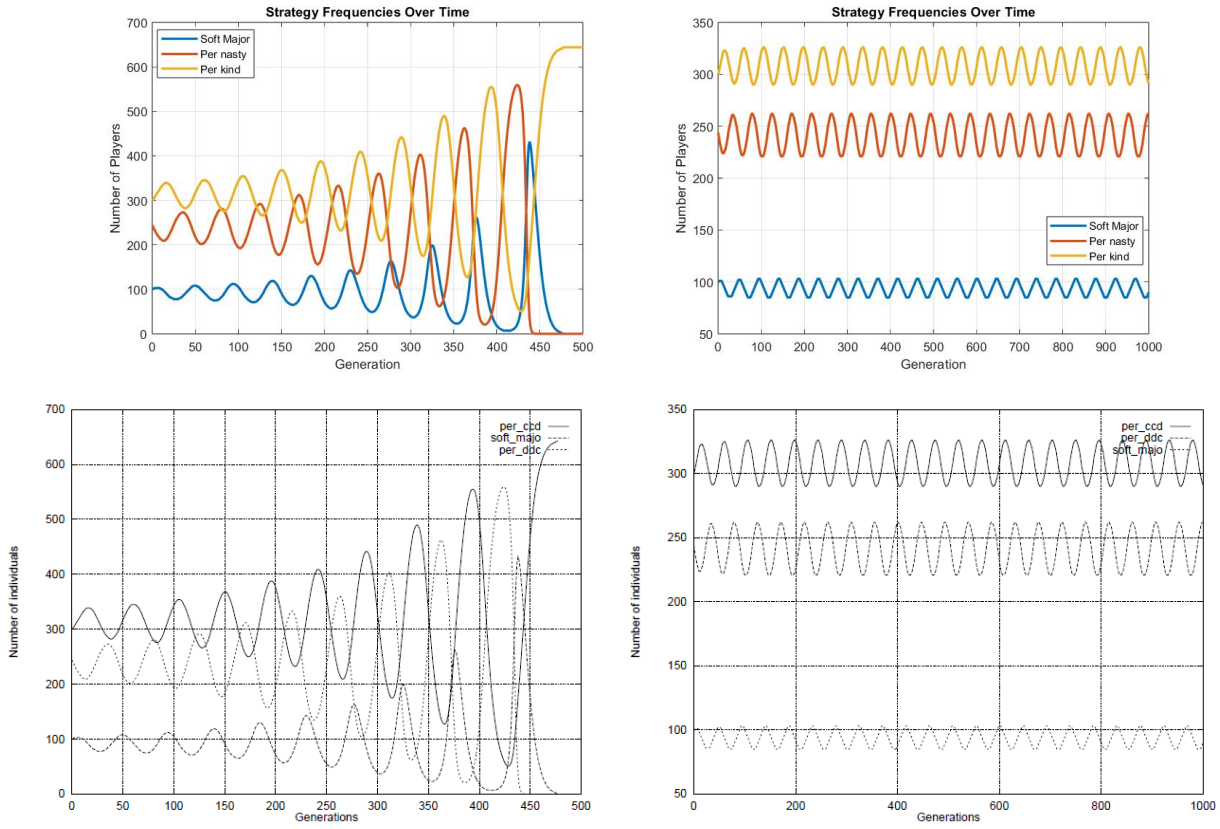


Figure 2.10: Sensitivity to CIPD payoff. All parameters are identical except that $T = 4.6$ on the left and $T = 4.7$ on the right. (`sens_cipd_payoff.m`)

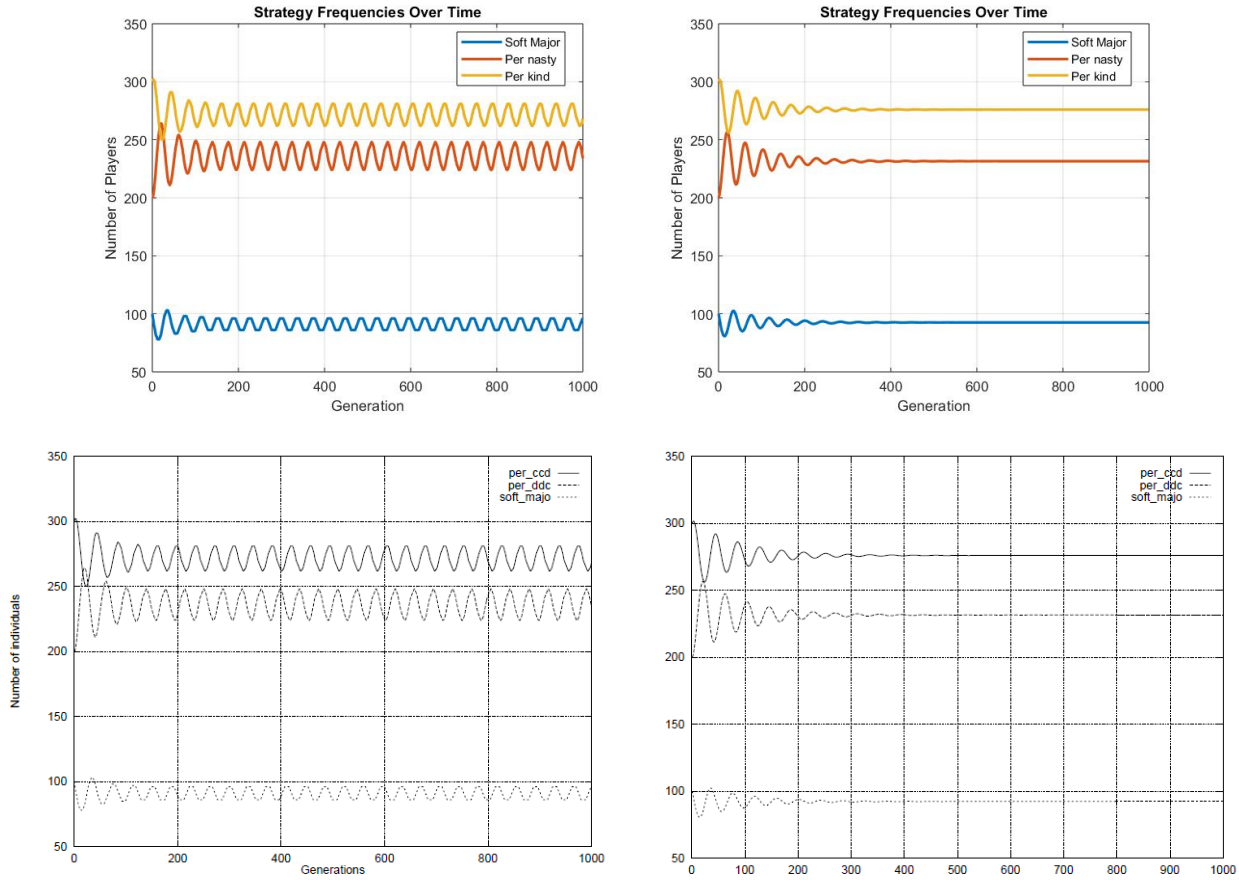


Figure 2.11: Sensitivity to repartition computation method. All parameters are identical except that repartition on the left is done by rounding and uses real value on the right. (`sens_repartition_real.m`)

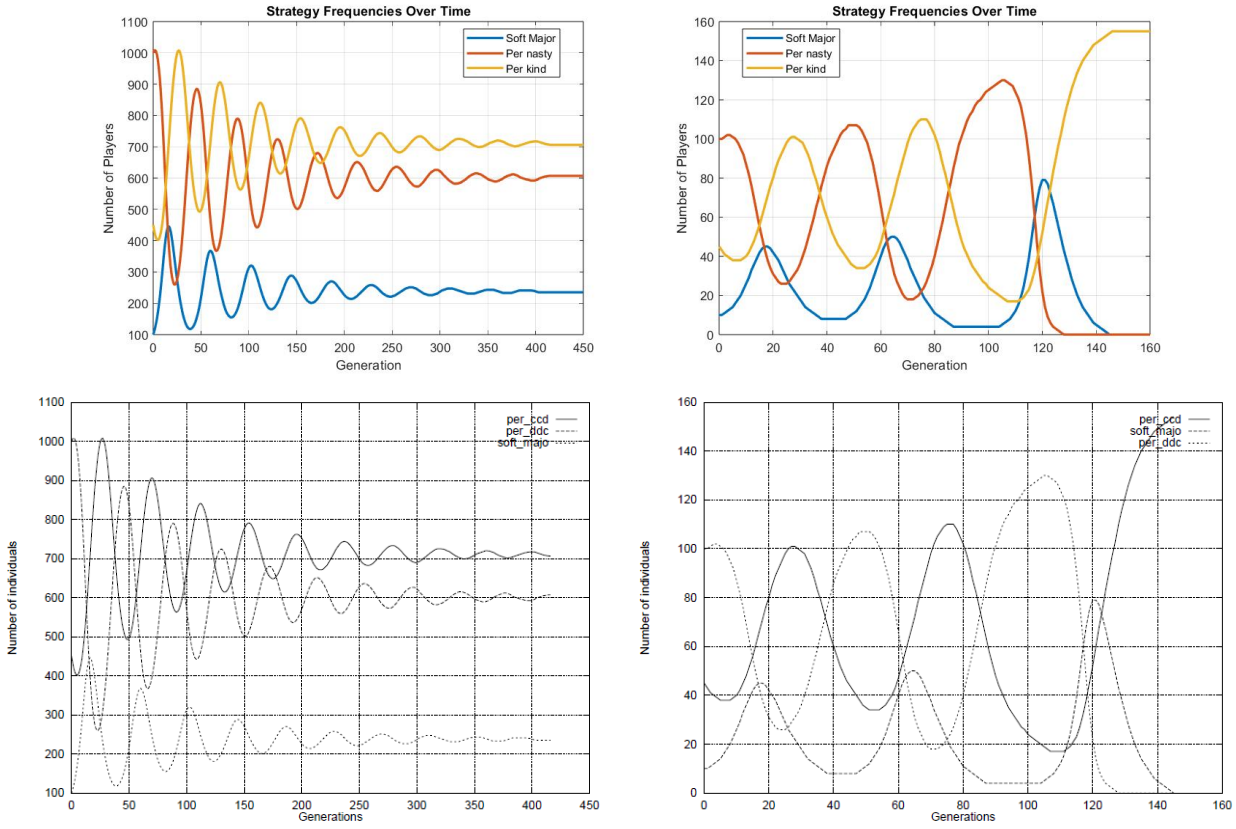


Figure 2.12: Sensitivity to repartition computation method. All parameters are identical except that populations on the right are divided by 10. (`sens_repartition_div10.m`)

Παρατηρούμε την επ' ακριβώς αντιστοίχιση με τα αποτελέσματα του paper ακόμα και σε επίπεδο debugging βλέποντας τους πίνακες των κερδών. Στη συνέχεια επιχειρήσαμε να δημιουργήσουμε αυτές τις καμπύλες τρέχοντας προσομοιώσεις όπου είχαμε να αποφασίσουμε μεταξύ της πλήρους ή της μερικής προσομοίωσης.

2.2 Προσομοίωση του Fitness Dynamics

Στη συνέχεια επιχειρήσαμε να δημιουργήσουμε αυτές τις καμπύλες τρέχοντας προσομοιώσεις όπου είχαμε να αποφασίσουμε μεταξύ της πλήρους ή της μερικής προσομοίωσης. Για τους σκοπούς αυτούς χρησιμοποιούμε τη συνάρτηση `TourSimFit` η οποία αναλύεται παρακάτω:

2.2.1 Θεωρητική Αναλυση του TourSimFit:

Η συνάρτηση αυτή είναι παρόμοια με την `TourTheFit` . Η βασική διαφορά είναι ότι για να διατηρηθεί ο πληθυσμός με ακέραιες τιμές χρησιμοποιούμε την συνάρτηση `Close_int_v`

Η συνάρτηση αυτή ουσιαστικά μεταφέρει το περισσευούμενο fractional part στα στοιχεία με το χαμηλότερο fractional part.

Σημαντική διαφορά εδώ είναι ότι , για να έχουμε μεγαλύτερη ταχύτητα στο Simulation, εκτελούμε τον κώδικα χρησιμοποιώντας την συνάρτηση `Axel2_improved`.

Έτσι εκτελούμε μια μορφή half simulation (όχι καθαρή προσομοίωση).

Τρέχοντας ξανά τις παρακάτω προσομοιώσεις παρατηρούμε τα εξής για το κάθε ένα πείραμα.

Defectors may be strong

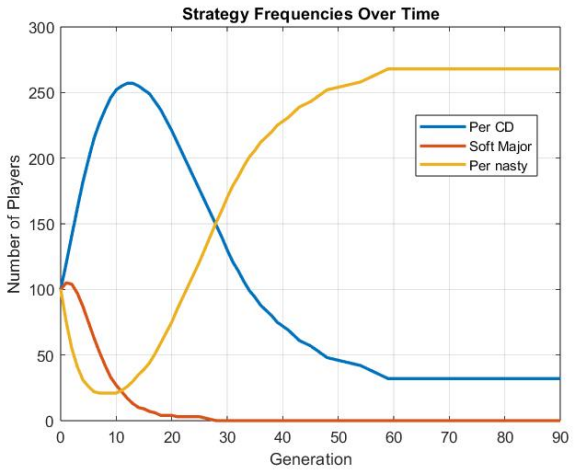


Figure 2.13: `defectors_may_be_strong_sim.m`

Παρατηρούμε παρόμοια συμπεριφορά με ίδιες γραφικές παραστάσεις με τη μόνη διαφορά η προσομοίωση να οδηγείται σε steady state μερικές γενιές νωρίτερα

Monotonous Convergence

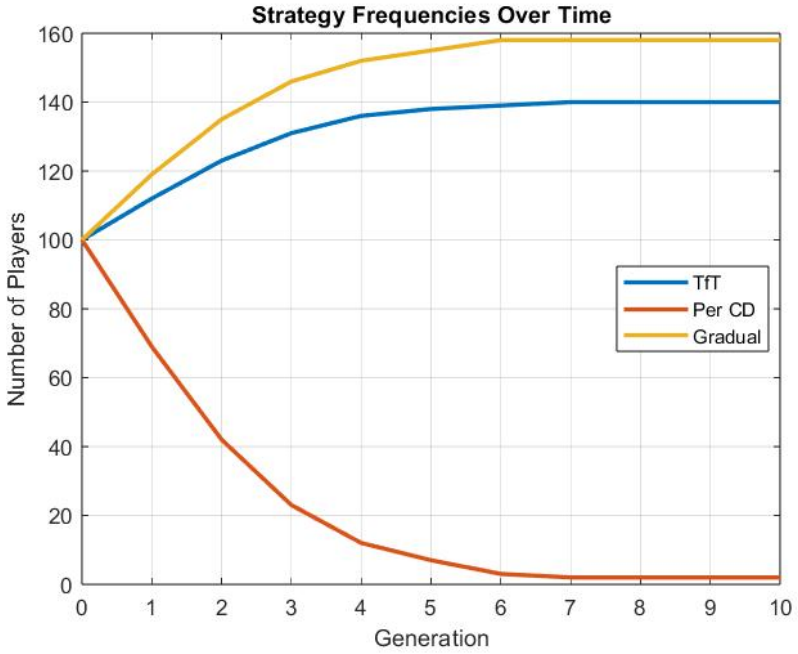


Figure 2.14: `monotonous_convergence_sim.m`

Παρατηρούμε και εδώ παρόμοια συμπεριφορά με ίδιες γραφικές παραστάσεις και ελαφρά αλλαγμένους τους πληθυσμούς του steady state.

Attenuated Oscillations

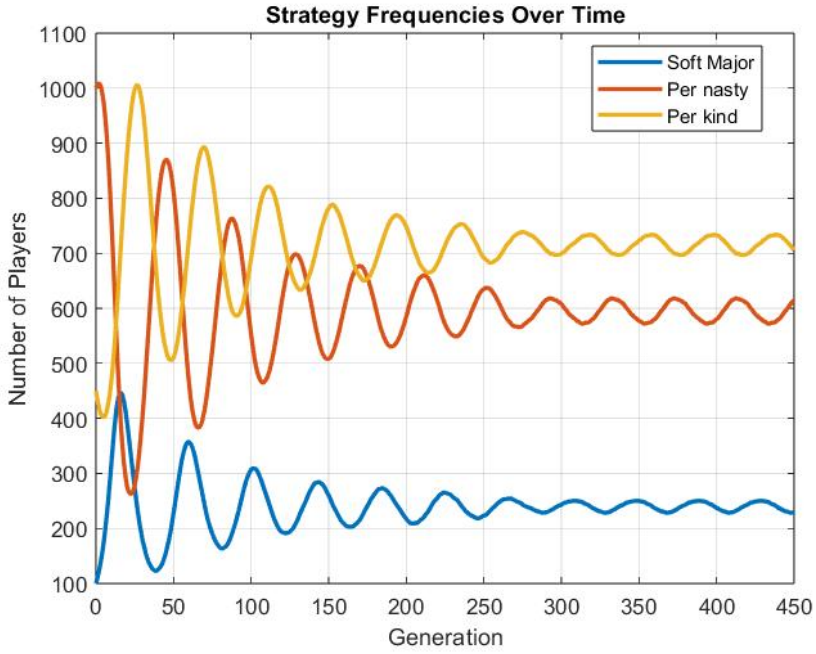


Figure 2.15: `attenuated_oscillatory_movements_sim.m`

Παρατηρούμε παρόμοια συμπεριφορά με ίδιες γραφικές παραστάσεις με το simulation να έχει μικρότερο βαθμό απόσβεσης.

Periodic Movements

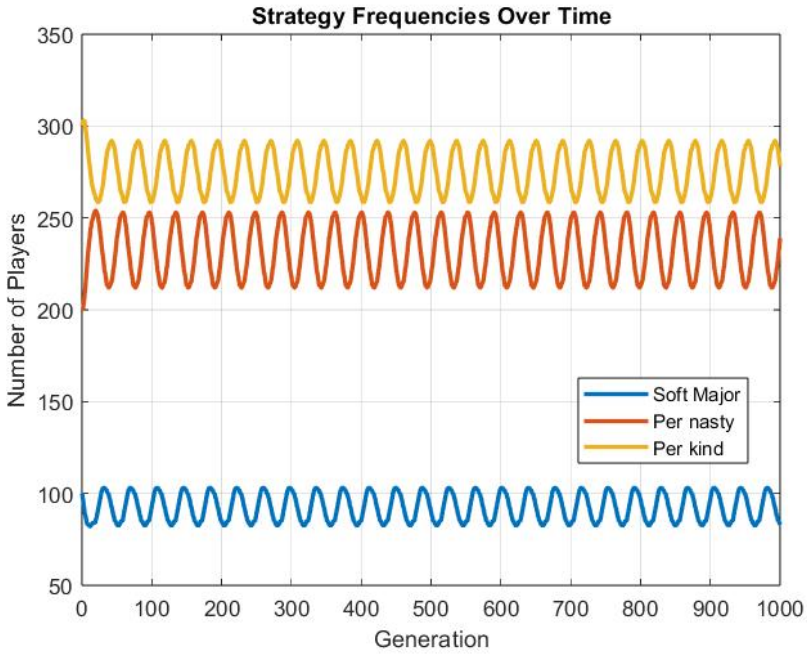


Figure 2.16: `periodic_movements_sim.m`

Παρατηρούμε παρόμοια συμπεριφορά με τη θεωρητική ανάλυση

Increasing Oscillations

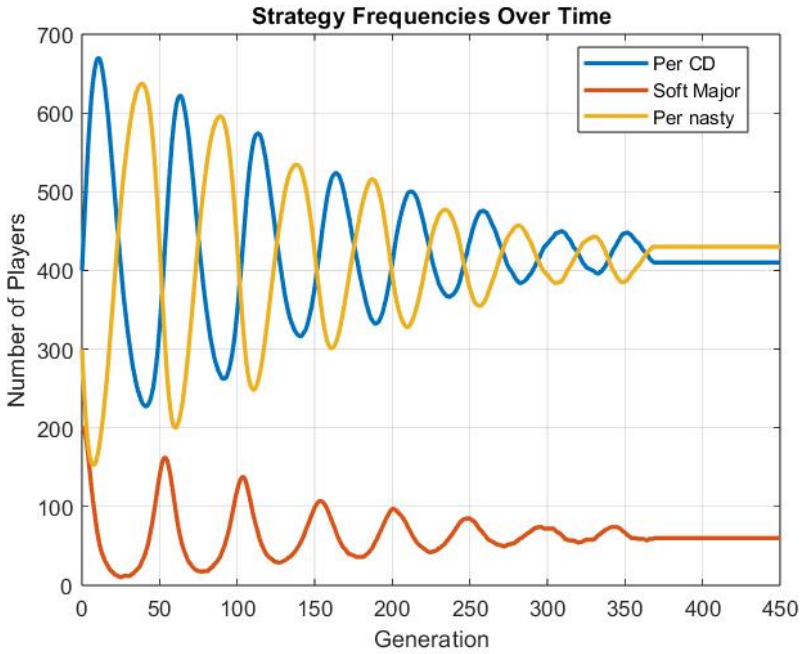


Figure 2.17: `increasing_oscillations_sim.m`

Παρατηρούμε σημαντική διαφορά καθώς στην περίπτωση της προσομοίωσης δε συμβαίνουν αυξανόμενες ταλαντώσεις. Αυτό οφείλεται στον τρόπο που γίνεται το rounding στην προσομοίωση και εμφανίζεται και σε επόμενα πειράματα πολλές φορές χαλώντας και την κατάταξη των στρατηγικών. Στο συγκεκριμένο αν και οι ταλαντώσεις είναι μειούμενες η κατάταξη διατηρείται.

Disordered Oscillations

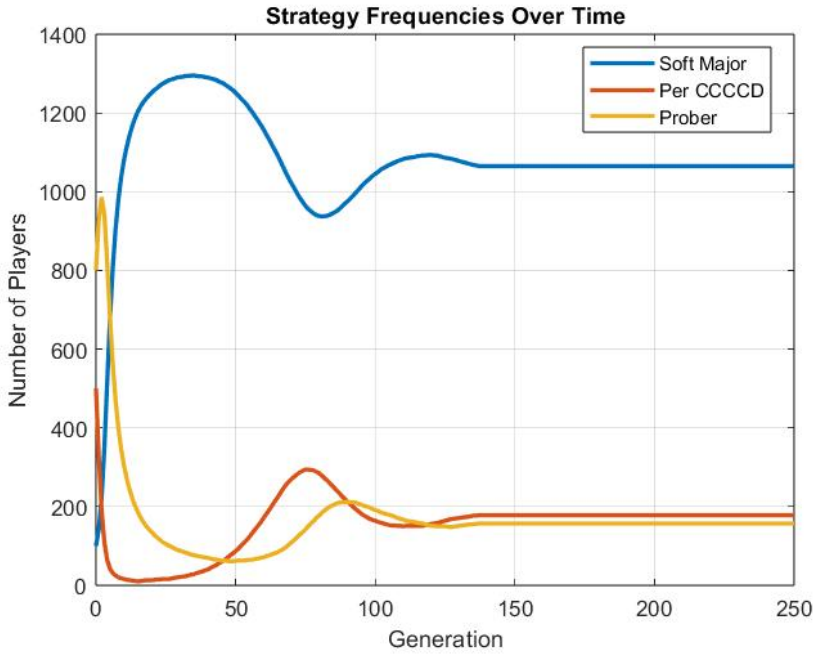


Figure 2.18: `disordered_oscillations_sim.m`

Παρατηρούμε παρόμοια συμπεριφορά με τη θεωρητική ανάλυση, ωστόσο οι μεταβολές είναι μικρότερες και καταλήγει σε λιγότερες γενιές σε steady state.

Sensitivity of dynamics to population size

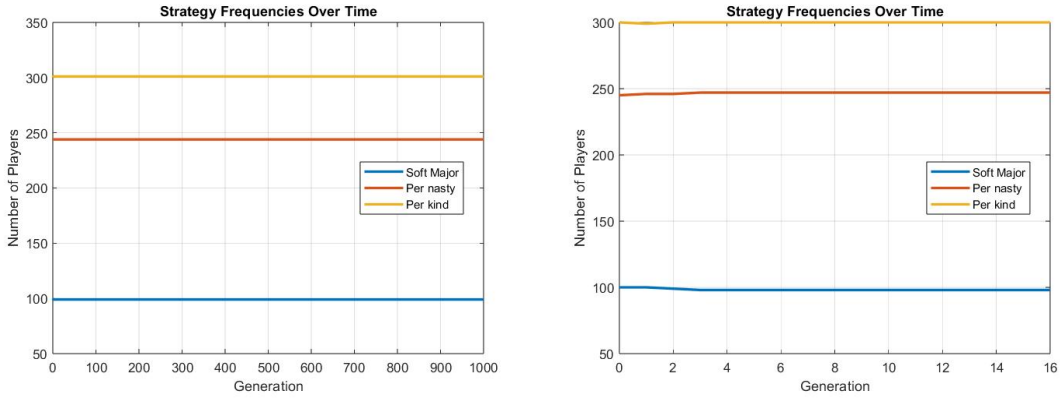


Figure 2.19: `sens_dyn_pop_size_sim.m`

Παρατηρούμε ότι οι αρχικές συνθήκες που στην θεωρητική ανάλυση προκαλούν περιοδικές ταλαντώσεις στην προσομοίωση αποτελούν ήδη steady state.

Sensitivity of dynamics to winner

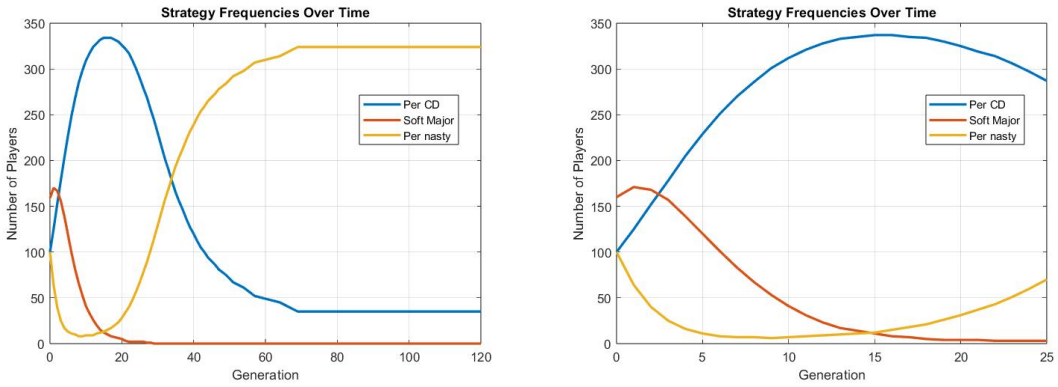


Figure 2.20: `sens_win_pop_size_sim.m`

Παρατηρούμε παρόμοια συμπεριφορά με τη θεωρητική ανάλυση

Sensitivity to CIPD Payoff

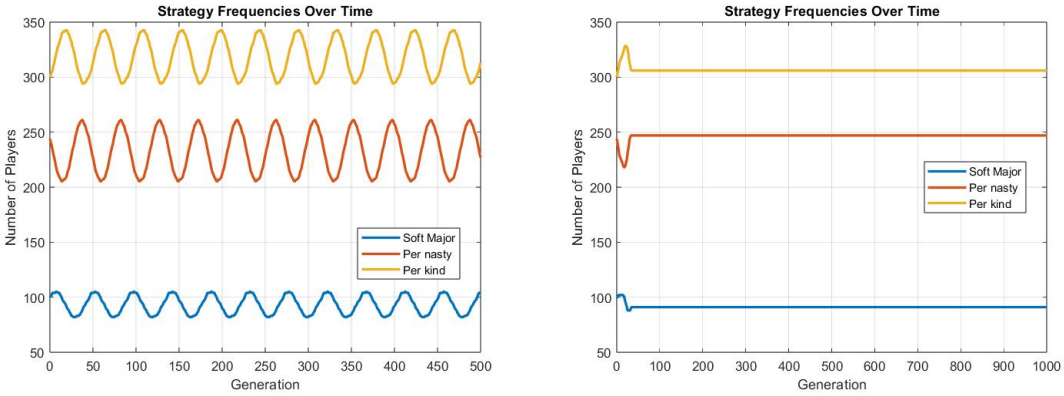


Figure 2.21: `sens_cipd_payoff_sim.m`

Παρατηρούμε στη θεωρητική ανάλυση ότι με τις τιμές αυτές του πίνακα των σκορ έχουμε αυξανόμενες ταλαντώσεις και περιοδικές ταλαντώσεις ενώ στην προσομοίωση αντίστοιχα περιοδικές ταλαντώσεις και ένα σύντομο μεταβατικό φαινόμενο που οδηγεί σε steady state.

Sensitivity to Game Length

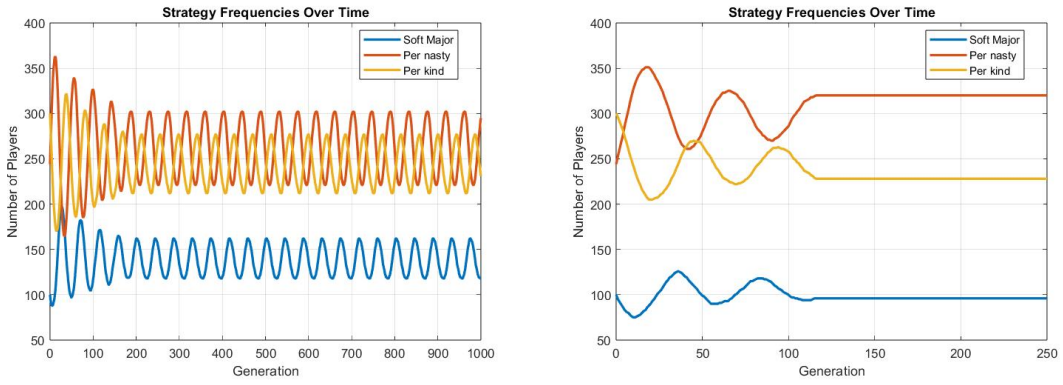


Figure 2.22: `sens_game_length_sim.m`

Παρατηρούμε παρόμοια συμπεριφορά με τη θεωρητική ανάλυση

Repartition divided by 10

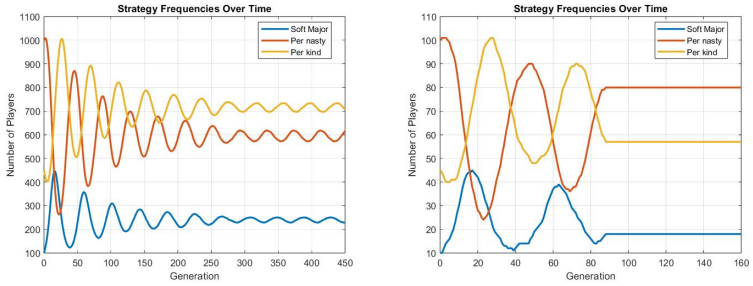


Figure 2.23: `sens_repartition_div10_sim.m`

Παρατηρούμε πως ενώ στη θεωρητική ανάλυση είχαμε μειούμενες και αυξανόμενες ταλαντώσεις στην προσομοίωση έχουμε και στις δύο περιπτώσεις μειούμενες ταλαντώσεις οι οποίες όμως στην περίπτωση της διαίρεσης με το 10 των πληθυσμών ανατρέπουν και την κατάταξη των στρατηγικών.

Κεφάλαιο 3

Imitation Dynamics

3.1 Θεωρητική ανάλυση του Imitation Dynamics:

Σε αυτό το σημείο θα αναλύσουμε τη θεωρητική ανάλυση του Imitation Dynamics. Αυτό περιλαμβάνει την δημιουργία ενός πίνακα μεταβάσεων μιας αλυσίδας Markov. Παρακάτω θα αναλύσουμε πως λειτουργεί αυτή η μέθοδος παραγωγής του πίνακα μεταβάσεων.

Παρακάτω ο κώδικας της συνάρτησης `TourTheImi`:

Παμε να δουμε αναλυτικά γραμμή προς γραμμή:
Αρχικα καλούμε την συνάρτηση `all_combinations_sum_m`, η οποία δίνει όλες τις πιθανές καταστάσεις της διαδικασίας Markov, της οποίας τον πίνακα μεταβάσεων αναζητούμε .

Η συνάρτηση ακολουθεί την λογική του *stars and bars problem* για να υπολογίσει ολους τους πιθανούς συνδυασμούς θετικών αριθμων που αθροίζουν στον πληθυσμο που έχουμε βάλει σαν όρισμα.

Προχωρώντας στις επόμενες γραμμές: Αρχικοποιούμε τον πίνακα μεταβάσεων και ξεκινάμε μια `for` loop για κάθε γραμμή του πίνακα, (δηλαδή για κάθε συνδυασμό των combos).

Εντός της λούπας: Υπολογίζουμε τα scores για κάθε στρατηγική με την συνάρτηση `Axel2_improved`.

Η βελτιωμένη αυτή εκδοση αξιοποιεί την ανάλυση από το `TourTheFit` για να υπολογίσει με μεγαλύτερη ταχύτητα τα Scores για κάθε στρατηγική.
Επειτα , με την συνάρτηση `find_best_inf_str` βρίσκω τις στρατηγικές (μια η περισσότερες) που είναι βέλτιστες. Επιπλέον υπολογίζει και τις υποβελτιστες στρατηγικές(θα μας χρησιμεύσουν μετα).

Προχωρώντας, χρησιμοποιώντας την `Assign_str` , την οποία έχουμε υλοποιήσει για το πρωτάθλημα Axelrod της 1ης εργασίας, μετατρέπουμε την μορφή:

$$X(t_i) = \Big(x_1(t_i), x_2(t_i), \ldots, x_m(t_i)\Big) \rightarrow Y(t_i) = \underbrace{\texttt{str}_1, \ldots \texttt{str}_1}_{x_1(t_i) \text{ times}}, \underbrace{\texttt{str}_2, \ldots \texttt{str}_2}_{x_2(t_i) \text{ times}}, \ldots, \underbrace{\texttt{str}_m, \ldots \texttt{str}_m}_{x_m(t_i) \text{ times}}$$

Η μορφή αυτή εξυπηρετεί τους εξής σκοπούς:

- Είναι ευκολότερη η εύρεση πιθανών μεταβολών παιχτών

- Δεν μας ενδιαφέρει η διάταξη των στοιχείων. Όπως είχαμε αναφέρει στο μάθημα, η διαδικασία markov για τις καταστάσεις Y (ας τις ονομάσουμε `r_states` έναντι των `s_states`) είναι lumpable

σε σχέση με τα `s_states` [1] Αυτό σημαίνει ότι το άθροισμα των πιθανοτήτων μεταβολών ενος συγκεκριμένου `group r_states` που αναπαριστούν ένα συγκεκριμένο `s_state` είναι σταθερό από οποιαδήποτε άλλη κατάσταση ενος συγκεκριμένου `s_state` και να προερχόμαστε.

Παρακάτω, εφόσον έχουμε τώρα το `r_state`, υπολογίζουμε τα *indexes* που αντιστοιχούν σε βέλτιστες και υποβέλτιστες στρατηγικές. Εδώ κάνουμε και μια παραδοχή για να μπορέσει ο μηχανισμός μας να αντιμετωπίσει ειδικες περιπτώσεις: Εφόσον ο αριθμός υποβέλτιστων παιχτών είναι μικρότερος από K, τότε σαν υποψήφιοι παίκτες προς αλλαγή στρατηγικής συμπεριλαμβάνονται και οι βελτιστοι παίκτες(δηλαδή όλοι οι παίκτες είναι υποψήφιοι για αλλαγή στρατηγικής, ανεξαρτήτως αν έχουν ήδη βέλτιστη στρατηγική).

Εφόσον έχουμε βρει τα indexes , αναπαράγουμε ολους τους πιθανούς συνδυασμούς indexes από το `idx_r` μήκους K και όλους τους πιθανούς συνδυασμούς βέλτιστων στρατηγικών που μπορούν να μεταβληθούν οι K παίκτες. Για το 2ο έχουμε υλοποιήσει μια συνάρτηση για να βρεί όλους τους πιθανούς συνδυασμούς με επανατοποθέτηση.
Η συνάρτηση αυτή μάλλον χρήζει βελτίωσης, αφού στην συγκεκριμένη μορφή της είναι αναδρομική.

Σε αυτό το σημείο ξεκινάνε οι υπολογισμοί στοιχείων του πίνακα μεταβάσεων. Ορίζουμε με την βοήθεια ενός cell array όλες τις πιθανές μεταβάσεις (σε μορφή `r_state`) από την παρούσα κατάσταση. Επειτα βρίσκω σε ποιό `s_state` αντιστοιχεί το κάθε `r_state` και στο αντίστοιχο στοιχείο $s \to s'$ προσθέτω $\frac{1}{p}$, όπου p ο πιθανός αριθμός των μεταβάσεων `r_states`.

Θα δοκιμάσουμε μερικά παραδείγματα με την συνάρτηση αυτή. (`TourTheImi`)

Η συνάρτηση έχει αρκετά γρήγορη ταχύτητα ακόμα και για μεγάλο πληθυσμό ή μήκος στρατηγικών. Θα δοκιμάσουμε ενα παράδειγμα με 5 παίκτες `Per_CD` , 5 παίκτες `Soft_Major` και 5 παίκτες `Per_Nasty`.

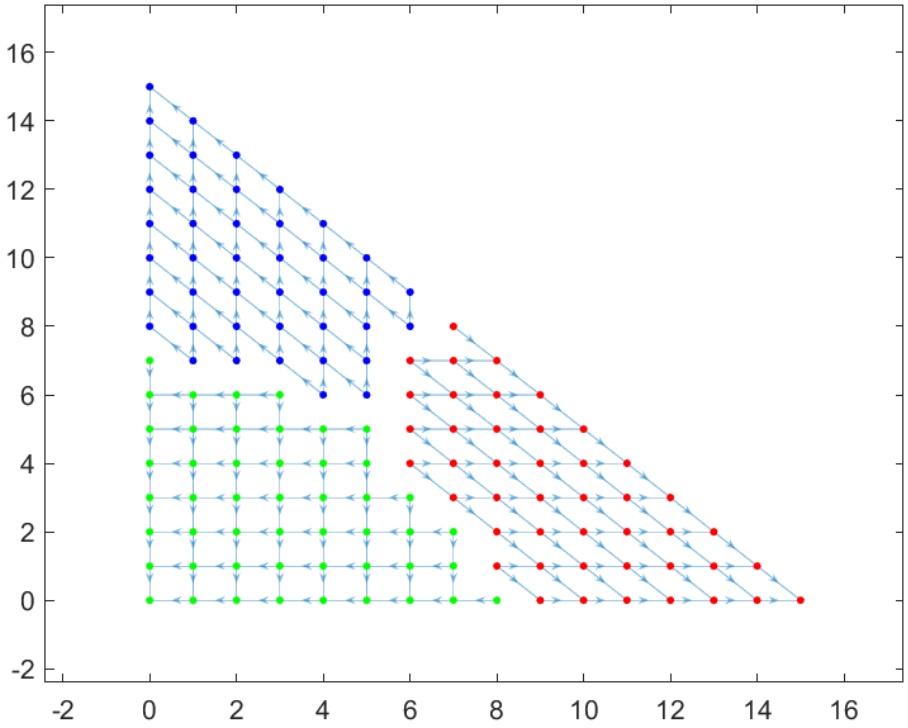


Figure 3.1: `TourTheImi` Ex. 1 `TourTheImi_Ex1.m`

Βλέπουμε ότι ο γράφος χωρίζεται σε ξεχωριστά groups ανάλογα με την σύγκλιση τους . Φαίνεται πως υπάρχει μια 1 προς 1 σχέση μεταξύ της αρχικής και τελικής κατάστασης(μπορούμε να γνωρίζουμε

[1] βλέπε στο Appendix για λεπτομέρειες

σίγουρα που θα καταλήξει μια κατάσταση από την αρχική κατάσταση και μόνο)
Έπειτα κάνουμε ένα πείραμα με 5 `TfT`, 5 `Per_CD`, 5 `Gradual`

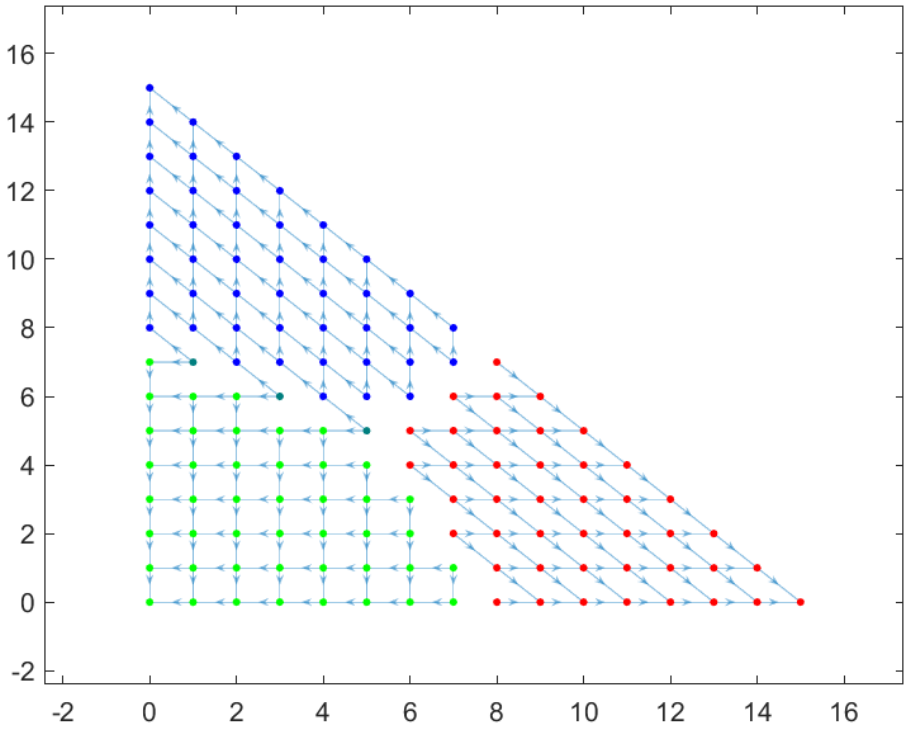


Figure 3.2: `TourTheImi` Ex. 2 `TourTheImi_Ex2.m`

Εδώ βλέπουμε ότι, παρόλο που ο γράφος πάλι χωρίζεται σε groups, υπάρχει μια επικοινωνία μεταξύ των 2 από τα 3 groups . Συγκεκριμένα , οι καταστάσεις [7, 1, 7] ,[6, 3, 6] και [5, 5, 5] έχουν πιθανότητα να οδηγηθεί τόσο στην κατάσταση [15, 0, 0] όσο και στην κατάσταση [0, 0, 15]. Αυτό λογικά συμβαίνει καθώς οι στρατηγικές `Tit_for_Tat` και `Gradual` έχουν ίδιο συνολικό score και στις 3 αυτές καταστάσεις.

Το επόμενο πείραμα αποτελείται από 3 `Soft_major`, 2 `Per_Nasty`, 1 `Per_kind`.

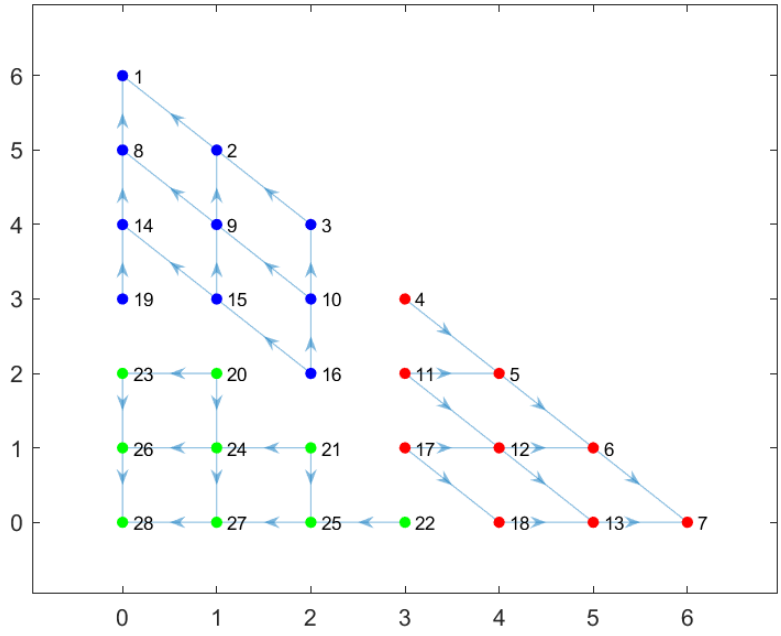


Figure 3.3: `TourTheImi` Ex. 3 `TourTheImi_Ex3.m`

Για την συνάρτηση `TourTheimi` δημιουργήσαμε μια δεύτερη παραλλαγή `TourTheImi2` με τον εξής

συλλογισμό: Το Score ενος πληθυσμού δεν θα έπρεπε να κρίνεται από τον αριθμό που τον εκπροσωπεί αλλά από την απόδοση ενός παίκτη.

Η μεγάλη διαφορά της `TourTheimi2` είναι ότι χρησιμοποιεί μια νέα συνάρτηση (`Axel2_Improved_2`) για τον υπολογισμό του Score ανα στρατηγική , η οποία λαμβάνει υπόψη το Score μόνο ενός παίκτη.

Παμε να εξετάσουμε τα ίδια παραδείγματα με πάνω με την νέα συνάρτηση.

Δοκιμάζουμε με 5 `per_CD`,5 `soft_major`, 5 `per_nasty`.

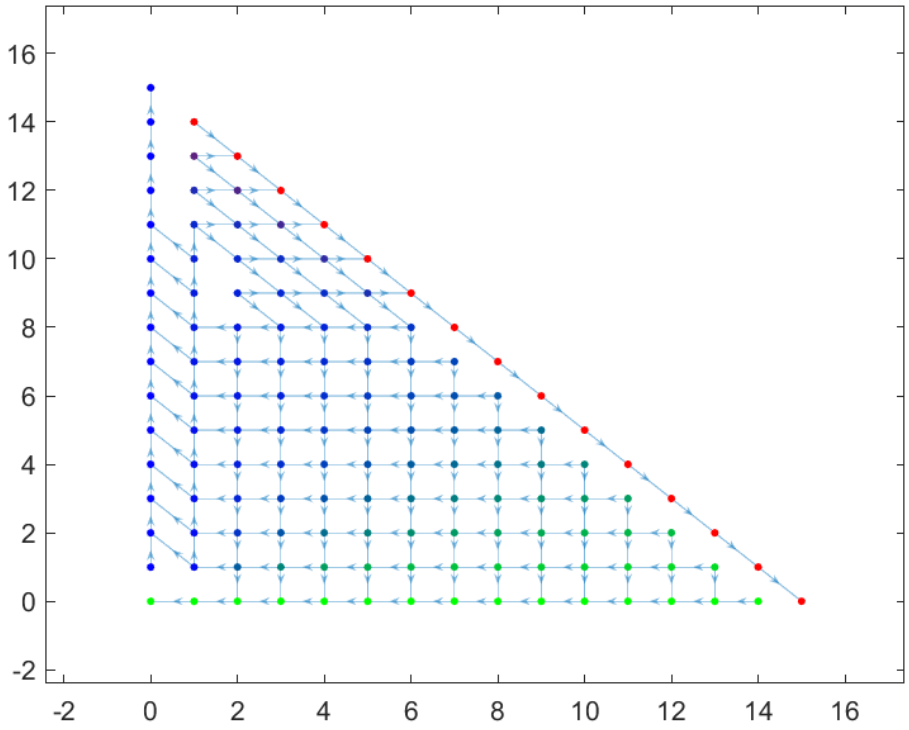


Figure 3.4: `TourTheImi` Ex1_2 `TourTheImi_Ex1_2.m`

Παρατηρούμε οτι το `TourTheImi2` έχει μεγάλη διαφοροποίηση. Εδω βλέπουμε ότι ο γράφος δεν χωρίζεται τόσο αισθητά σε Groups. Αυτό μας δείχνει πως υπάρχουν ισοδύναμες στρατηγικές στο συγκεκριμένο παράδειγμα.

Έπειτα έχουμε 5 `TfT`, 5 `Per_CD`, 5 `Gradual`.

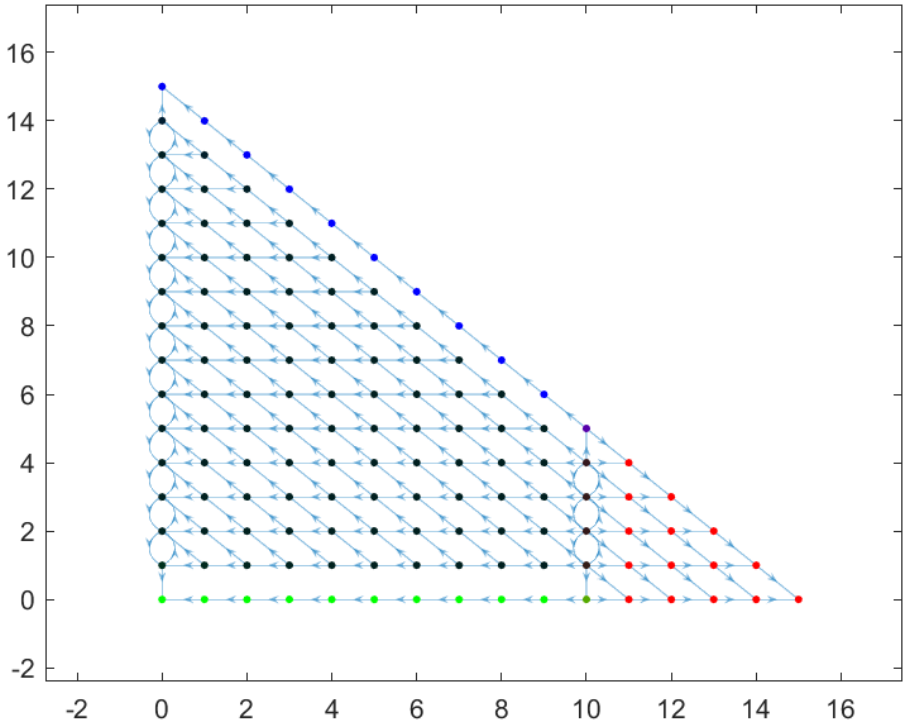


Figure 3.5: `TourTheImi` Ex2_2 `TourTheImi_Ex2_2.m`

Στο συγκεκριμένο παράδειγμα παρατηρούμε ότι στην περίπτωση που η στρατηγική `Per_CD` εκλείψει, οι στρατηγικές `Tit_for_Tat` και `Gradual` ταλαντεύονται στο ποιά θα υπερισχύσει. Επιπλέον στις καταστάσεις [1, 10, 4],[2, 10, 3],[3, 10, 2] και [4, 10, 1] βλέπουμε ότι από αυτές τις καταστάσεις μπορούμε να μετακινηθούμε προς πολλές κατευθύνσεις.

Επόμενο πείραμα: 3 `Soft_Major`, 2 `Per_Nasty`, 1 `Per_kind`

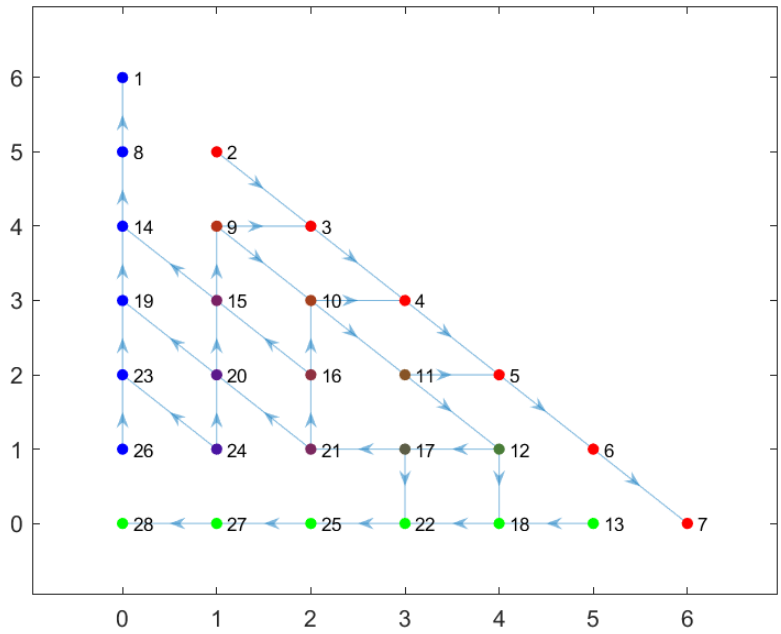


Figure 3.6: `TourTheImi` Ex3_2 `TourTheImi_Ex3_2.m`

Οπως και πριν , δεν παρατηρούμε κάτι ιδιαίτερο στο συγκεκριμένο πείραμα , ακόμα και με το TourTheImi2.

3.2 Προσομοίωση του Imitation Dynamics

Ο κώδικας για το Simulation του Imitation Dynamics βρίσκεται εδω:`TourSimImi`

Η λογική είναι παρόμοια με την `TourTheImi`, με την διαφορά ότι αυτή τη φορά δεν χρειάζεται να υπολογίσουμε τον πίνακα μεταβάσεων: Επιλέγουμε τυχαία K παίκτες και επιλέγουμε τυχαία μια από τις βέλτιστες στρατηγικές για να αλλάξουμε τους K παίκτες.

Εκτελώντας τα αντίστοιχα πειράματα με τη θεωρητική υλοποίηση παρατηρούμε ότι στην περίπτωση του Imitation όπου η βέλτιστη στρατηγική βρίσκεται από το συνολικό σκορ όλών των παικτών η συμπεριφορά του συστήματος εμφανίζει τις παρακάτω ιδιομορφίες:

1. Στην περίπτωση που ένας πληθυσμός είναι αρχικά μεγαλύτερος από έναν άλλο, ο πληθυσμος τείνει να υπερισχύει έναντι των άλλων, όπως βλέπουμε στην 3η εικόνα που εκτελείται το χαοτικό πείραμα.

2. Στην περίπτωση που οι πληθυσμοί είναι ίδιοι, τείνει να υπερισχύει αυτός που σε σχέση με το Theoretical Fitness εμφανίζει την πιο πρώιμη αύξηση. Αυτό μπορεί να οδηγήσει τόσο σε ορθή κατάταξη , όπως στην 2η εικονα στο πείραμα της μονότονης σύγκλισης, όσο και σε λανθασμένη , όπως στην πρώτη εικόνα με το πείραμα των ισχυρών αποστατών.

Στην περίπτωση του Imitation όπου η βέλτιστη στρατηγική βρίσκεται από το σκορ του καλύτερου παίκτη, τα αποτελέσματα παρουσιάζονται πιο χαοτικά και δεν υπάρχει απαραίτητα μονοτονία στις μεταβολές όπως παρατηρείται στην πρώτη περίπτωση.

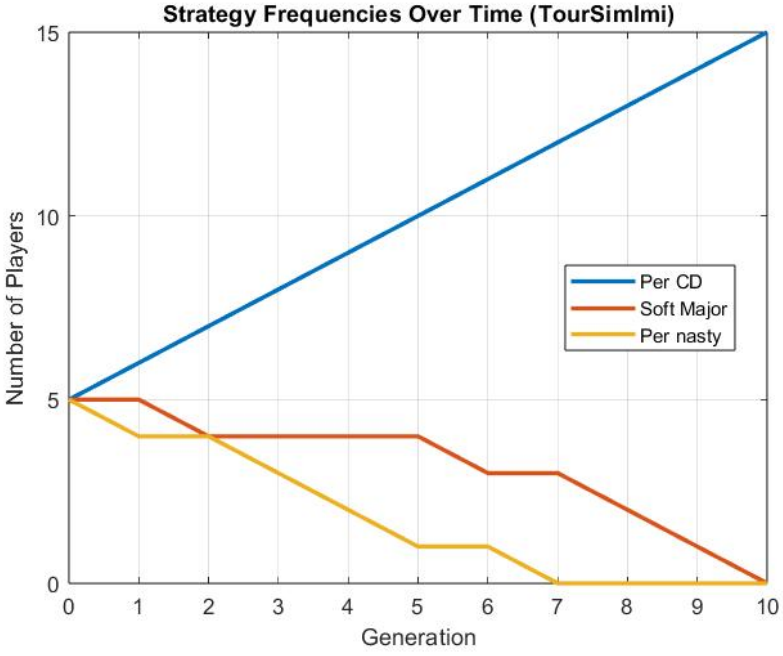


Figure 3.7: Convergence to the Final State `TourSimImi_Ex1.m`

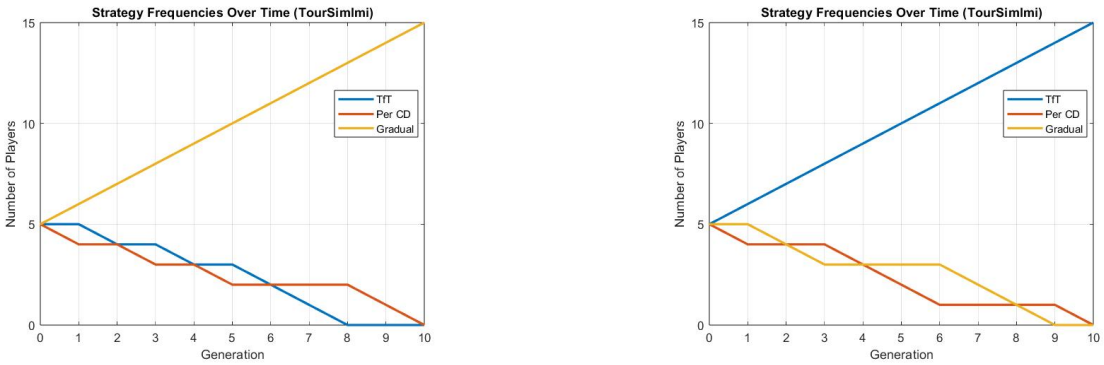


Figure 3.8: Convergence of the states to the two possible final states shown by Markov Chain `TourSimImi_Ex2.m`

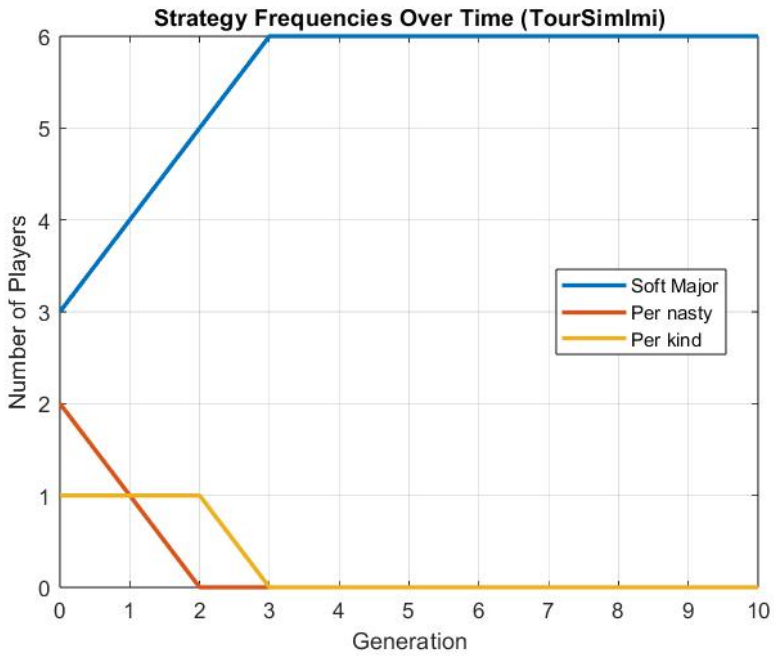


Figure 3.9: Convergence to the Final State (`TourSimImi_Ex3.m`)

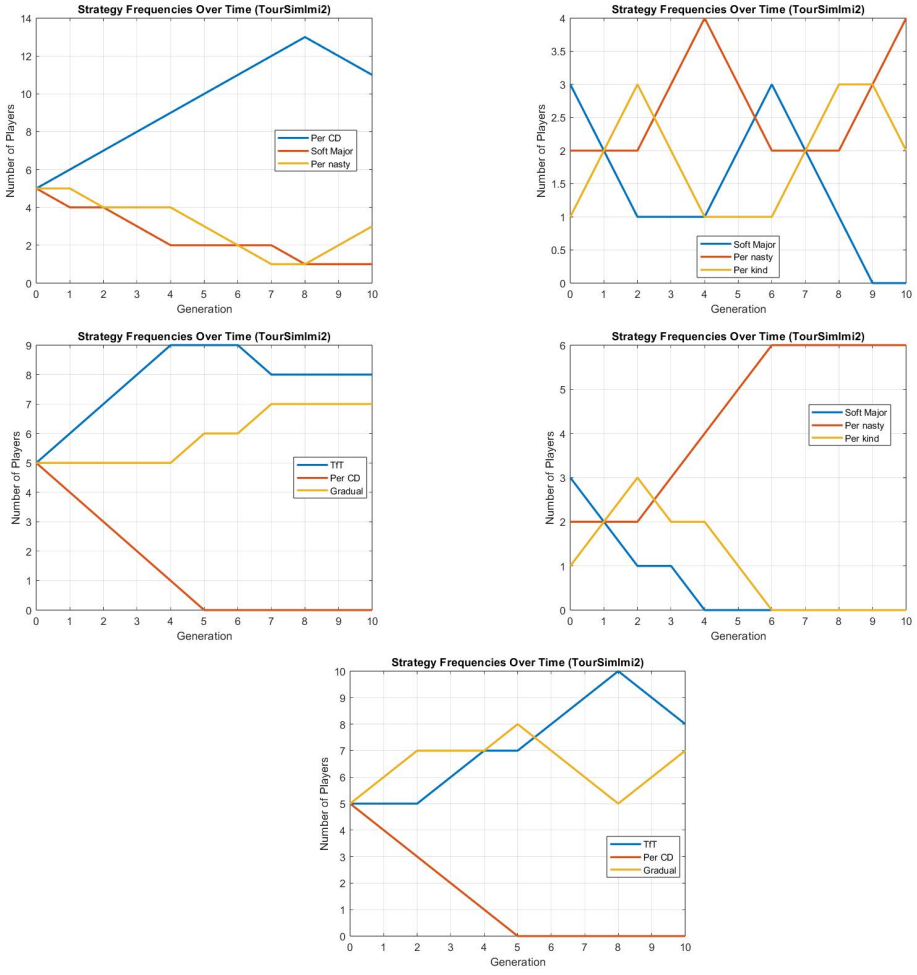


Figure 3.10: Increased Oscilations and random state transition using `TourSimImi2` `TourSimImi_Ex1_2.m`, `TourSimImi_Ex2_2.m`, `TourSimImi_Ex3_2.m`

Κεφάλαιο 4

Discussion

Μέσα από την θεωρητική ανάλυση του Fitness Dynamics παρατηρούμε την ακριβή επαλήθευση των αποτελεσμάτων του paper καθώς και την επαλήθευσή τους σε πραγματικό χρόνο από την προσομοίωση.

Σε κάθε περίπτωση οι αρχικές συνθήκες παίζουν πολύ σημαντικό ρόλο και δόθηκε αρκετός χρόνος στην επίτευξη των ακριβών αποτελεσμάτων. Στη συνέχεια από το Imitation Dynamics παρατηρούμε την σύγκλιση των πληθυσμών προς σταθερές καταστάσεις μέσα από τις αλυσίδες Markov και πώς μερικές φορές από μια αρχική κατάσταση είμαστε βέβαιοι ότι στο τέλος θα οδηγηθούμε σε μια συγκεκριμένη τελική ενώ άλλες όχι. Τα βήματα προς την τελική κατάσταση από το Simulation των Imitation Dynamics εξαρτώνται και αυτά από διάφορους μηχανισμούς όπως την επιλογή βέλτιστης στρατηγικής και τον μηχανισμό επιλογής των παικτών προς αλλαγή. Σε αυτό το σημείο και καθώς δόθηκε αρκετή ελευθερία στην μοντελοποίηση του συστήματος τα αποτελέσματα μας ενδέχεται να διαφέρουν σε σχέση με τις υπόλοιπες ομάδες, ενώ καθώς ο τρόπος επιλογής παικτών προς αλλαγή περιλαμβάνει στοχαστικότητα τα figures που συμπεριλάβαμε ενδέχεται να μη παραχθούν ολόιδια με τα scripts ασχέτως αν ο μηχανισμός παραμένει κοινός.
Τέλος, η ανάλυση αυτή μπορεί να δώσει χρήσιμα συμπεράσματα τόσο για τα δυναμικά φαινόμενα στα εξελικτικά πρωταθλήματα ενώ σημαντικό ερευνητικό ενδιαφέρον θα παρουσίαζε η εφαρμογή της παραπάνω ανάλυσης και στα άλλα 2 διαφορετικά κοινωνικά διλλήματα τα οποία είναι οι Ιέρακες και Περιστερές και το Κυνήγι του Ελαφιού με αντίστοιχες προσαρμογές.

Appendices

Appx. A:

Απόδειξη μεταξύ s_states , r_states από κ. Κεχαγιά

The General Case $N \in \mathbb{N}$

As already mentioned, the process $(\boldsymbol{r}(t))_{t=0}^{\infty}$ is a MC. Let us now look at the $(s(t))_{t=0}^{\infty}$ process. We have

$$\forall t : \boldsymbol{s}(t) = F(\boldsymbol{r}(t)).$$

We note that the inverse F^{-1} is a set-valued function:

$$\forall \boldsymbol{s} : F^{-1}(\boldsymbol{s}) = \{\boldsymbol{r} : F(\boldsymbol{r}) = \boldsymbol{s}\}.$$

The following Definition 1 and Proposition 1 are from [1].

Definition 1

Suppose we are given: (a) a Markov chain $(\boldsymbol{x}(t))_{t=0}^{\infty}$ with state space $R = \{1, \ldots, N\}$ and (b) a partition $\mathcal{S} = \{R_1, \ldots, R_M\}$ of R. We construct a new stochastic process $(\boldsymbol{y}(t))_{x=0}^{\infty}$ with state space $\mathcal{S}$ by setting

$$\forall t : \boldsymbol{y}(t) = R_m \text{ iff } \boldsymbol{x}(t) \in R_m.$$

We say that $(\boldsymbol{x}(t))_{t=0}^{\infty}$ is *lumpable with respect to* $\mathcal{S}$ iff $(\boldsymbol{y}(t))_{t=0}^{\infty}$ is a MC with transition probabilities which do not depend on the initial probability $(\pi_{\boldsymbol{x}})_{\boldsymbol{x}\in R}$ (where $\pi_{\boldsymbol{x}} = \Pr(x(0) = x)$).

Proposition 1

Given $(\boldsymbol{x}(t))_{t=0}^{\infty}$ and $(\boldsymbol{y}(t))_{t=0}^{\infty}$ as in Definition 3.2.1, $(x(t))_{t=0}^{\infty}$ is lumpable with respect to $\{R_1, \ldots, R_M\}$ iff

$$\forall \boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathcal{S}, \forall \boldsymbol{x}, \hat{\boldsymbol{x}} \in F^{-1}(\boldsymbol{y}) : \Pr(\boldsymbol{x} \to \hat{\boldsymbol{y}}) = \Pr(\hat{\boldsymbol{x}} \to \hat{\boldsymbol{y}}).$$

Then it is also true that

$$\forall \boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathcal{S}, \forall \boldsymbol{x} \in F^{-1}(\boldsymbol{y}) : \Pr(\boldsymbol{y} \to \hat{\boldsymbol{y}}) = \Pr(\boldsymbol{x} \to \hat{\boldsymbol{y}}).$$

Now, regarding our $(\boldsymbol{r}(t))_{t=0}^{\infty}$ and $(\boldsymbol{s}(t))_{t=0}^{\infty}$ processes, we have the following.
$(\boldsymbol{r}(t))_{t=0}^{\infty}$ is lumpable with respect to $\mathcal{S}$.

Proof. Take any $\boldsymbol{s}, \hat{\boldsymbol{s}} \in \mathcal{S}$ and any $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$, $\hat{\boldsymbol{r}} = (\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_N) \in F^{-1}(\boldsymbol{s})$. So we have $F(\boldsymbol{r}) = F(\hat{\boldsymbol{r}}) = \boldsymbol{s}$. This means that $\boldsymbol{r}$ and $\hat{\boldsymbol{r}}$ contain the same number of 1's, 2's and 3's; in other words, $\hat{\boldsymbol{r}}$ is obtained by a permutation of the elements of $\boldsymbol{r}$:

$$(\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_N) = (r_{i_1}, r_{i_2}, \ldots, r_{i_N})$$

where $(i_1, \ldots, i_N)$ is a permutation of $(1, 2, \ldots, N)$. Now

$$\begin{aligned}
\Pr(\boldsymbol{r} \to \boldsymbol{s}) &= \Pr((r_1, r_2, \ldots, r_N) \to \boldsymbol{s}) \\
&= \sum_{(r'_1, r'_2, \ldots, r'_N) \in F(\boldsymbol{s})} \Pr((r_1, r_2, \ldots, r_N) \to (r'_1, r'_2, \ldots, r'_N)) \\
&= \sum_{(r'_1, r'_2, \ldots, r'_N) \in F(\boldsymbol{s})} \Pr\left((r_{i_1}, r_{i_2}, \ldots, r_{i_N}) \to (r'_{i_1}, r'_{i_2}, \ldots, r'_{i_N})\right) \\
&= \sum_{(r'_{i_1}, r'_{i_2}, \ldots, r'_{i_N}) \in F(\boldsymbol{s})} \Pr\left((\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_N) \to (r'_{i_1}, r'_{i_2}, \ldots, r'_{i_N})\right) \\
&= \Pr(\hat{\boldsymbol{r}} \to \hat{\boldsymbol{s}}).
\end{aligned}$$

So the condition of Proposition 3.2.2 is satisfied and $(\boldsymbol{r}(t))_{t=0}^{\infty}$ is lumpable. ■

Appx. B: Code

TourTheFit

```
function [POP,BST,FIT] = TourTheFit(B,Strategies,Pop0,T,J)
Re_matrix=Reward_str(B,Strategies,T);

l_str=length(Strategies);

P=sum(Pop0);

POP=zeros(J,l_str);
FIT=zeros(J-1,l_str);
BST=cell(J-1,1);

POP(1,:)=Pop0;


for i=1:J-1
FIT(i,:)=(Re_matrix*POP(i,:)'-diag(Re_matrix))';

t=FIT(i,:)*POP(i,:)';

m=max(FIT(i,:));

temp= FIT(i,:)==m;

BST{i}=Strategies(temp);

POP(i+1,:)=POP(i,:).*FIT(i,:)*(P/t);



end
end

```

Reward_str

```
function Matrix = Reward_str(B,Strategies,T)
l=length(Strategies);

Matrix=zeros(l);

for i=1:l
for j=i:l
memory_game=[]; %The memory for each game
for k=1:T
p1_move=move(memory_game,Strategies(i));
if isempty(memory_game)
p2_move=move(memory_game,Strategies(j));
else
memory_game2=memory_game(:,[2 1]); % change the perspective of the memory so the 2
  nd column is the opponent
p2_move=move(memory_game2,Strategies(j));
end

Matrix(i,j)=Matrix(i,j)+B(p1_move,p2_move);
if i ~= j
```

```
Matrix(j,i)=Matrix(j,i)+B(p2_move,p1_move);
end

memory_game=[memory_game;p1_move p2_move]; %#ok<AGROW> % put the new moves into
  memory


end

end

end
end

```

TourSimFit

```
function [POP,BST,FIT]=TourSimFit(B,Strategies,Pop0,T,J)
l_str=length(Strategies);

P=sum(Pop0);

POP=zeros(J,l_str);
FIT=zeros(J-1,l_str);
BST=cell(J-1,1);

POP(1,:)=Pop0;

for i=1:J
disp(i);
Scores=Axel2_improved(B,Strategies,POP(i,:),T);

%FIT(i,:)=Sum_scores(Scores,Strategies,POP(i,:));

FIT(i,:) = Axel2_improved(B,Strategies,POP(i,:),T);

m=max(FIT(i,:));

temp= FIT(i,:)==m;

BST{i}=Strategies(temp);

prc=FIT(i,:)/sum(FIT(i,:));

POP(i+1,:)=P*prc;

alive_str=find(POP(i+1,:)>0);



POP(i+1,alive_str(1:end))=Close_int_v(POP(i+1,alive_str(1:end)));



end
end

```

Close_int_v

```
function y= Close_int_v(x)
fl_x=floor(x); % floor function of all the elements

fr_x=x-fl_x; % fractional part of all the elements

r=int64(sum(fr_x)); % the integernumber we need to put back into fl_x

[~,I]=sort(fr_x); %sorting the numbers based on their fractional part

fl_x(I(1:r))=fl_x(I(1:r))+1;

y=fl_x;


end

```

Axel2_improved

```
function Scores_per_Group= Axel2_improved(B,Strategies,Pop,T)
Re_matrix=Reward_str(B,Strategies,T);

Score_per_category=(Re_matrix*Pop'-diag(Re_matrix))';

Scores_per_Group=Score_per_category.*Pop;
End



```

TourTheImi

```
function P_table=TourTheImi(B,Strategies,PoP0,K,T)
combos=all_combinations_sum_m(length(Strategies),sum(PoP0)); %all combinations of
  the population with the same Strategies
P_table=zeros(size(combos,1));
for i=1:size(combos,1)
Score_str=Axel2_improved(B,Strategies,combos(i,:),T);
[bstr, rest]= find_best_inf_str(Score_str,Strategies);
Popr=Assign_str(Strategies,combos(i,:));
idx_r= find(ismember(Popr,rest));

idx_bst=find(ismember(Popr,bstr));
if length(idx_r)<K
idx_r=[idx_r idx_bst];
end
comb_pick=nchoosek(idx_r,K);

poss_ch=combinations_with_replacement(length(bstr),K);
New_st=cell(size(comb_pick,1),size(poss_ch,1));

p=numel(New_st);

%
for j=1:size(comb_pick)
for k=1:size(poss_ch,1)
temp=Popr;
```

```
temp(comb_pick(j,:))=bstr(poss_ch(k,:));


New_st{j,k}=r2s_state(temp,Strategies);
check= sum(combos == New_st{j,k},2) == size(combos,2);
check=find(check);
P_table(i,check)=P_table(i,check)+1/p;
end

end

end

end
```

all_combinations_sum_m

```
    function combos = all_combinations_sum_m(n, m)
    % This function returns all combinations of non-negative integers
    % of length n that sum to m
    % Total number of places (m stars + n - 1 bars)
    total = m + n - 1;
    % Generate all combinations of n - 1 bar positions
    bar_positions = nchoosek(1:total, n - 1);
    num_combos = size(bar_positions, 1);
    combos = zeros(num_combos, n);
    for i = 1:num_combos
    % The positions of bars split m stars into n parts
    % Add 0 and total+1 to make it easier to compute differences
    positions = [0, bar_positions(i,:), total + 1];
    combos(i, :) = diff(positions) - 1;
    end
    end


```

find_best_inf_str

```

    function [best_str,rest_str]=find_best_inf_str(Score_per_str,Strategies)
    m=max(Score_per_str);
    temp= Score_per_str==m;
    temp2= ~temp;
    best_str=Strategies(temp);
    rest_str=Strategies(temp2);
    end


```

Assign_str

```
    function str_assign= Assign_str(Strategies,Pop)
    sum_Pop=sum(Pop); %Find the size of the population
    Pop2=[0 Pop];
    Pop2=cumsum(Pop2); % The Pop2 helps by defining the limits between the population
     of one Strategy and another
```

```
    str_assign=strings(1,sum_Pop); %Initialize the string array
    for i=1:length(Strategies)
    str_assign(Pop2(i)+1:Pop2(i+1))=Strategies(i); % since the populations and
    strategies are well organized we use Pop2 to assign the Strategies
    end

    end


```

TourTheImi2

```
    function P_table=TourTheImi2(B,Strategies,PoP0,K,T)
    combos=all_combinations_sum_m(length(Strategies),sum(PoP0));
    P_table=zeros(size(combos,1));
    for i=1:size(combos,1)
    Score_str=Axel2_improved_2(B,Strategies,combos(i,:),T);
    [bstr, rest]= find_best_inf_str(Score_str,Strategies);
    Popr=Assign_str(Strategies,combos(i,:));
    idx_r= find(ismember(Popr,rest));

    idx_bst=find(ismember(Popr,bstr));
    if length(idx_r)<K
    idx_r=[idx_r idx_bst];
    end
    comb_pick=nchoosek(idx_r,K);

    poss_ch=combinations_with_replacement(length(bstr),K);
    New_st=cell(size(comb_pick,1),size(poss_ch,1));

    p=numel(New_st);

    %
    for j=1:size(comb_pick)
    for k=1:size(poss_ch,1)
    temp=Popr;

    temp(comb_pick(j,:))=bstr(poss_ch(k,:));


    New_st{j,k}=r2s_state(temp,Strategies);
    check= sum(combos == New_st{j,k},2) == size(combos,2);
    check=find(check);
    P_table(i,check)=P_table(i,check)+1/p;
    end
    end
    end
    end


```

Axel2_Improved_2

```
    function Scores_per_Group= Axel2_improved_2(B,Strategies,Pop,T)
    Re_matrix=Reward_str(B,Strategies,T);
    Score_per_category=(Re_matrix*Pop'-diag(Re_matrix))';
    Pop_check= Pop>0;
    Scores_per_Group=Score_per_category.*Pop_check;
    end
```

TourSimImi

```
function [POP, BST] = TourSimImi(B, Strategies, POP0, K, T, J)
% TourSimImi - K-player imitation simulation over J generations
% Inputs and outputs same as before
N = length(Strategies);
total_pop = sum(POP0);
POP = zeros(J+1, N);
POP(1, :) = POP0;
BST = zeros(J, 1);  % Track best strategy index per generation
for gen = 1:J
current_counts = POP(gen, :);
current_pop = Assign_str(Strategies, current_counts);  % Get full strategy list
% Compute scores for each strategy group
score_str = Axel2_improved(B, Strategies, current_counts, T);
% Identify best-performing strategies
[best_strats, rest_strats] = find_best_inf_str(score_str, Strategies);
best_idx = find(strcmp(Strategies, best_strats{1}));
a=randi(length(best_strats));
if (length(best_strats)) > 1
a=randi(length(best_strats));
BST(gen) = a;
else
BST(gen) = best_idx;
end
% Identify which individuals are using non-best strategies
rest_idx = find(ismember(current_pop, rest_strats));
% If not enough, allow some best to "reconsider"
if length(rest_idx) < K
bst_idx = find(ismember(current_pop, best_strats));
needed = K - length(rest_idx);
rest_idx = [rest_idx; bst_idx(randperm(length(bst_idx), needed))];
else
rest_idx = rest_idx(randperm(length(rest_idx), K));
end
% Choose new strategies for these K individuals
new_pop = current_pop;
for i = 1:K
% Randomly choose one of the best strategies to imitate

new_pop(rest_idx(i)) = best_strats(a);
end

POP(gen+1,:) = r2s_state(new_pop,Strategies);
end
end


```

References

1. J.G. Kemeny, J.L. Snell, Finite Markov Chains. Van Nostrand, 1976.

2. Mathieu, Philippe & Beaufils, Bruno & Jean-Paul, Delahaye. (1999). Studies on Dynamics in the Classical Iterated Prisoner's Dilemma with Few Strategies. 177-190. 10.1007/10721187_13.

3. Report κ. Κεχαγιά

4. Alexander, J. McKenzie. Evolutionary game theory. Cambridge University Press, 2023.

5. Axelrod, Robert, and William D. Hamilton. "The evolution of cooperation." Science, Vol. 211, No. 4489, pp. 1390-1396, 1981.

6. Axelrod, Robert, and Douglas Dion. "The further evolution of cooperation." Science, Vol. 242, No. 4884, pp. 1385-1390, 1988.

7. Mathieu, Philippe, Bruno Beaufils, and Jean-Paul Delahaye. "Studies on Dynamics in the Classical Iterated Prisoner's Dilemma with Few Strategies" European conference on artificial evolution. Springer: Berlin Heidelberg, 1999.