Game Theory: Kinetic Duels and Learning

George Anastasiadis

October 3, 2024

Contents

1 Εισαγωγή

Στο πλαίσιο της παρούσας εργασίας υλοποιήσαμε μια προγραμματιστική προσομοίωση ενός παίγνιου κινητικής διμαχίας. Το παίγνιο είναι μια χαρακτηριτική περίπτωση παιγνίου μηδενικού αθροίσματος(zero-sum game). Επίσης εξετάσαμε δυνατότητες εφαρμογής μεθόδων μάθησης στο παίγνιο αυτό. Συγκεκριμένα, εξετάσαμε διαφορετικές περιπτώσεις στις οποίες έχουμε διαφορετικά επίπεδα γνώσεις των παραμέτρων του παιγνίου, άρα διαφορετικές μορφές του παιγνίου της κινητικής διμαχίας μερικής πληροφόρησης. Στις μεθόδους μάθησης που δοκιμάσαμε οι παίκτες παίζοντας επαναλαμβανόμενα το παίγνιο προσπαθούν να αποκτήσουν πληρέστερη γνώση των δεδομένων που αρχικά τους είναι άγνωστα ή έχουν μια γενική a priori γνώση για αυτά τα δεδομένα. Σκοπός είναι με τις πάροδο των επαναλήψεων να βελτιστοποιούν τις στρατηγικές τους αξιοποιώντας τη γνώση που έχουν λάβει μέσω εμπειρίας τους από τα πρηγούμενα παίγνια.

1.1 Κανόνες του παίγνιου

Η συγκεριμένη περίπτωση παίγνιου κινητικής διμαχίας που εξετάσαμε ακολουθεί τους εξής κανόνες:

1. Το παίγνιο παίζεται από δυο παίκτες, τους P_1 και P_2
2. Το παιχνίδι εξελίσσεται σε διακριτά χρονικά βήματα, σε γύρους $t = \{1, 2, ...\}$
3. Οι παίκτες παίζουν εναλλάξ (ακολουθιακά), δηλαδή σε κάθε γύρο μόνο ένας παίκτης εκτελεί μία δράση. Ο παίκτης P_1 έχει την πρώτη κίνηση. Ο P_1 έχει επομένως σειρά σε περιττούς γύρους και ο P_2 σε άρτιους γύρους
4. Οι αρχικές θέσεις των παικτών είναι $x_1(0) = 0$,$x_2(0) = D = 2N$ $(N \in \mathbb{N})$
5. Σε κάθε γύρο, ο παίκτης που έχει σειρά έχει την επιλογή είτε (α) να μετακινηθεί κατά μία απόσταση $x_m = 1$ προς τον άλλο παίκτη ή (β) να πυροβολήσει.
6. Η πιθανότητα ευστοχίας p_n ενός παίκτη P_n είναι:

$$p_n = \begin{cases} 1 & ,\text{όταν } d = 1 \\ \min\left(\frac{c_n}{d^{k_n}}, 1\right) & ,\text{όταν } d > 1. \end{cases}$$

   όπου $k_n, c_n > 0$ είναι παράμετροι
7. Κάθε παίκτης έχει μια μόνο σφαίρα
8. Αν κάποιος παίκτης πυροβολήσει και αστοχήσει, τότε ο αντίπαλος του πλησιάζει σε απόσταση 1 και τον σκοτώνει
9. Η απολαβή κάθε παίκτη αν σκοτώσει τον αντίπαλο του είναι 1, ενώ είναι -1 αν ο ίδιος πεθάνει

1.2 Στρατηγικές

Οι πιθανές στρατηγικές των παικτών είναι ένα κατώφλι απόστασης το οποίο επιλέγει ο κάθε παίκτης πριν την έναρξη της διμαχίας. Κάθε παίκτης αν βρίσκεται σε απόσταση μικρότερη από το κατώφλι που επέλεξε και είναι η σειρά του να δράσει πυροβολεί τον αντίπαλο του, διαφορετικά συνεχίζει να προχωράει προς αυτόν. Για παράδειγμα αν ένας παίκτης επιλέξει κατώφλι ίσο με 5 τότε αν έρθει η σειρά του και η απόσταση μεταξύ των 2 παικτών είναι μικρότερη από 5, τότε επιλέγει να πυροβολήσει, διαφορετικά συνεχίζει και προχωράει προς τον αντίπαλο του

1.3 Απολαβές

Έστω πως βρισκόμαστε σε έναν οποιοδήποτε γύρο του παιχνιδιού και κανένας από τους δυο παίκτες δεν έχει πυροβολήσει ακόμα. Για να κερδίσει το παιχνίδι ο P_1(σε περριτό γύρο) πρέπει να πετύχει τον αντίπαλο ή να αστοχήσει ο αντίπαλος του(δηλαδή ο P_2) σε κάποιο άρτιο γύρο όπου θα έχει σειρά.

Ας εξετάσουμε λοιπόν την περίπτωση που οι παίκτες βρίσκονται στις θέσεις $x1 = m$ και $x2 = n$(με $m < n$). Έστω πως ο γύρος είναι περιττός, άρα είναι η σειρά του P_1, η πιθανότητά του να πετύχει τον αντίπαλο του σε περίπτωση που πυροβολήσει είναι $p_1(n-m)$. Αντίστοιχα αν ήμασταν σε άρτιο γύρο(άρα είχε σειρά ο P_2) τότε αν ο P_2 πυροβολούσε η πιθανότητα αστοχίας του είναι $1-p_2(n-m)$. Σε περιττούς γύρους η απόσταση είναι άρτιος αριθμός, καθώς η αρχική απόσταση τον πρώτο γύρο είναι D = 2N . Πιο αναλυτικά:

$$P_{mn} = Pr(\text{νίκη } P_1) = \begin{cases} p_1(n-m) & ,\text{όταν } m-n \text{ είναι άρτιος αριθμός} \\ 1-p_2(n-m) & ,\text{όταν } m-n \text{ είναι περιττός αριθμός} \end{cases}$$

Επομένως το κέρδος του P_1 είναι:

$$\begin{matrix} 1 \cdot p_1(n-m) + (-1) \cdot (1-p_2(n-m)), \text{ όταν } n-m \text{ είναι άρτιος αριθμός} \\ (-1) \cdot p_2(n-m) + 1 \cdot (1-p_1(n-m)), \text{ όταν } n-m \text{ είναι περιττός αριθμός} \end{matrix} \quad (1)$$

Οπότε τελικά έχουμε:

$$Q_{mn} = \begin{cases} 2p_1(n-m)-1 & ,\text{όταν } m-n = 2k \\ 1-2p_2(n-m) & ,\text{όταν } m-n = 2k+1 \end{cases}$$

Το κέρδος Q_{mn} ισούται με το αναμενόμενο κέρδος του παίκτη P_1 αν κάποιος από τους δυο παίκτες πυροβολήσεις από απόσταση $d = m-n$. Δεδομένου πως το παιχνίδι είναι μηδενικού αθροίσματος το αντίστοιχο κέρδος του παίκτη P_2 είναι το ιδίο με αυτό του P_1 πολλαπλασιασμένο με το -1.

1.4 Πίνακας απολαβών

Όπως αναφέραμε και προηγουμένως οι πιθανές στρατηγικές κάθε παίκτη είναι η επιλογή ενός κατωφλιού απόστασης κάτω από το οποίο εκείνος πυροβολεί τον αντίπαλο του. Γνωρίζοντας το πως προκύπτουν οι απολαβές κάθε παίκτη, μπορούμε να κατασκευάσουμε έναν πίνακα απολαβών. Ο πίνακας απολαβών θα είναι ένας $D \times D$ πίνακας, όπου κάθε γραμμή του θα αντιπροσωπεύει την επιλογή στρατηγικής του παίκτη P_1 ενώ κάθε στήλη την επιλογή στρατηγικής του P_2. Σε κάθε κελί θα συμπληρώνουμε την αναμενόμενη απολαβή για αυτό το σετ στρατηγικών. Ο κώδικας που υπολογίζει τον πίνακα απολαβών A είναι ο εξής:

```
import numpy as np
def calculate_A(D, c1, k1, c2, k2):

    A = np.zeros((D, D))
    for d1 in range(1, D + 1):
        for d2 in range(1, D + 1):
            for x1 in range(0, D + 1):
                for x2 in range(D, -1, -1):
                    d = abs(x1 - x2)
                    if d != 0:
                        if d % 2 == 0 and d <= d1:
                            A[d1 - 1, d2 - 1] = 2 * c1 / (d ** k1) - 1
                        elif d % 2 == 1 and d <= d2:
                            A[d1 - 1, d2 - 1] = 1 - 2 * c2 / (d ** k2)
    return A
```

Φυσικά για τη λειτουργία της παραπάνω συνάρτησης θα πρέπει να γνωρίζουμε τις παραμέτρους των συναρτήσεων ευστοχίας (c, k) κάθε παίκτη.

1.5 Κυρίαρχες στρατηγικές

Γνωρίζοντας της παραμέτρους ευστοχίας των παικτών(ή έστω έχοντας μια εκτίμηση για αυτές) μπορούμε να υπολογίσουμε τον πίνακα απολαβών. Στη συνέχεια μπορούμε στον πίνακα αυτόν να εφαρμόσουμε την μέθοδο εξάλειψης στρατηγικών minmax, ώστε να βρούμε την κυρίαρχη στρατηγική για κάθε παίκτη. Σύμφωνα με αυτή τη μέθοδο κάθε παίκτης επιλέγει τη δράση που θα φέρει τη μέγιστη απολαβή όταν έρθει η σειρά του και απαλείφει τις υπόλοιπες διαθέσιμες στρατηγικές. Κάθε παίκτης θεωρεί πως ο αντίπαλος του θα πράξει παρόμοια επιλέγοντας την στρατηγική που θα του δώσει τη μέγιστη απολαβή.

Ο παίκτης P_1(γραμμοπαίκτης) προσπαθεί να μεγιστοποιήσει την τελική απολαβή(υπενθυμίζεται πως ο πίνακας απολαβών περιλαμβάνει τις απολαβές του παίκτη P_1), ενώ ο P_2(στηλοπαίκτης) προσπαθεί να την ελαχιστοποιήσει. Έτσι, πρακτικά εξετάζουμε το εξής: Αν όλες οι τιμές μιας γραμμής είναι μικρότερες ή ίσες από τις τιμές των υπολοίπων γραμμών του πίνακα, τότε εκείνη η γραμμή απαλείφεται καθώς δεν θα την επιλέξει σε καμια περίπτωση ο γραμμοπαίκτης P_1. Αντίστοιχα μια στήλη απαλείφεται αν οι τιμές τις είναι μεγαλύτερες ή ίσες από οποιαδήποτε άλλη στήλη.

```
import numpy as np
from scipy.optimize import linprog
def minmax(A):
    r, c = A.shape

    # Solve for Player 1's strategy
    AA1 = np.hstack([-A.T, np.ones((c, 1))])
    Aeq1 = np.append(np.ones(r), 0).reshape(1, -1)
    b1 = np.zeros(c)
    beq1 = 1
    lb1 = [(0, None)] * r + [(-np.inf, None)]
    f1 = np.append(np.zeros(r), -1)

    result1 = linprog(f1, A_ub=AA1, b_ub=b1, A_eq=Aeq1, b_eq=beq1, bounds=lb1, method
    ='highs')

    if not result1.success:
        raise ValueError("Linear programming did not converge for Player 1")

    p1 = result1.x[:r]
    v1 = result1.x[r]

    # Solve for Player 2's strategy
    AA2 = np.hstack([-A, np.ones((r, 1))])
    Aeq2 = np.append(np.ones(c), 0).reshape(1, -1)
    b2 = np.zeros(r)
    beq2 = 1
    lb2 = [(0, None)] * c + [(-np.inf, None)]
    f2 = np.append(np.zeros(c), -1)

    result2 = linprog(f2, A_ub=AA2, b_ub=b2, A_eq=Aeq2, b_eq=beq2, bounds=lb2, method
    ='highs')

    if not result2.success:
        raise ValueError("Linear programming did not converge for Player 2")

    p2 = result2.x[:c]
    v2 = result2.x[c]

return v1, p1, p2
```

2 Περιπτώσεις και Προσεγγίσεις

2.1 Εισαγωγή

Στο συγκεριμένο κεφάλαιο, θα παρουσιάσουμε όλες τις διαφορετικές εκδοχές τους παιγνίου στις οποίες προσπαθήσαμε να εφαρμόσουμε μια μέθοδο μάθησης. Σε κάθε εκδοχή οι παραμέτροι ευστοχίας που θα είναι γνωστοί για κάθε παίκτη διαφοροποιούνται. Σε κάποιες περιπτώσεις για παράδειγμα είναι άγνωστη ακόμα και η μορφή της συνάρτησης ευστοχίας των παικτών ενώ σε άλλες είναι γνωστή η μορφή αυτή αλλά είναι άγνωστες κάποιες ή και όλες οι παράμετροι της συνάρτησης αυτής. Πρακτικά όταν αναφερόμαστε σε μια μέθοδο μάθησης εννοούμε κάποιες μεθόδους εκτίμησης παραμέτρων του παιγνίου τις οποίες ο κάθε παίκτης αγνοεί. Για να γίνει αυτό κάθε παίκτης συλλέγει δεδομένα από τις επαναλήψεις του παιγνίου και τις αξιοποιεί ώστε να κάνει εκτιμήσεις για τις παραμέτρους που τον ενδιαφέρουν και να ανανεώνει τις δράσεις του στις επαναλήψεις του παίγνιου που θα ακολουθήσουν.

2.2 Άγνωστη συνάρτηση ευστοχίας-Χρήση μέσου όρου

2.2.1 Περιγραφή

Αρχικά εξετάζουμε μια πολύ βασική περίπτωση όπου ισχύουν τα εξής:

- Κάθε παίκτης δεν γνωρίζει τίποτα για τη δική του ευστοχία αλλά ούτε και του αντιπάλου του (τόσο για τη μορφή της συνάρτησης ευστοχίας αλλά και για τις τιμές που αυτή λαμβάνει)
- Προσομοιώνουμε πολλαπλά παίγνια της μορφής που αναφέτεραι στο 1.1. Σε κάθε παίγνιο η στρατηγική που ακολουθεί κάθε παίκτης επιλέγεται τυχαία από μια ομοιόμορφη κατανομή πάνω στο διάνυσμα των πιθανών στρατηγικών*
- Κάθε παίκτης προσπαθεί να κάνει εκτίμηση της ευστοχίας του αντιπάλου του. Για να το πετύχει αυτό μετά το τέλος των πολλαπλών παιγνίων με τυχαίες στρατηγικές παρατηρεί το μέσο όρο της ευστοχίας του (αλλά και του αντιπάλου του) από κάθε απόσταση και υπολογίζει το ποσοστό αυτό για κάθε τιμή της απόστασης.
- Στη συνέχεια υπολογίζεται ο πίνακας απολαβών με βάση τις ευστοχίες που εκτίμησαν οι παίκτες από κάθε απόσταση.
- Τέλος, χρησιμοποιώντας την μέθοδο minmax στον πίνακα απολαβών βρίσκουν το εκτιμώμενο σημείο ισορροπίας του παίγνιου και καταλήγουν σε αυτό.

2.2.2 Ο ψευδοκώδικας

Ο κώδικας που βρίσκεται στο αρχείο Mean_Estimate.py υλοποιεί την παραπάνω προσέγγιση που περιγράφεται στο 2.2.1 έχοντας ως δεδομένο πως η αρχική απόσταση μεταξύ των δυο παικτών είναι ίση με 10 και πως και οι δυο παίκτες έχουν την ίδια συνάρτηση ευστοχίας($c = 1$ και $k = 0.5$). Οι παίκτες παίζουν επαναλαμβανόμενα 100.000 παίγνια με τυχαίες στρατηγικές και στο τέλος των επαναλήψεων υπολογίζουν τον πίνακα αναμενόμενων απολαβών(πίνακας A) και με την τεχνική minmax βρίσκουν το σημείο ισορροπίας του παίγνιου. Τέλος, συγκρίνουμε το αποτέλεσμα με αυτό που θα αναμενόταν αν υπολογίζαμε τον πραγματικό πίνακα απολαβών(πίνακας A_real) και χρησιμοποιώντας πάλι minmax βρίσκαμε το σημείο ισορροπίας. Τα αποτελέσματα είναι πολύ καλά και φαίνεται το εκτιμώμενο σημείο ισορροπίας να ταυτίζεται με το αναμενόμενο.

*Υπενθυμίζεται πως στρατηγική σημαίνει η επιλογή ενός συγκεκριμένου κατωφλιού απόφασης κάτω από το οποίο ο παίκτης πυροβολεί.

Algorithm 1 Εκτίμηση με χρήση μέσου όρου

$initial\_distance \leftarrow 10$
$c_1 \leftarrow 1$
$k_1 \leftarrow 0.5$
$c_2 \leftarrow 1$
$k_2 \leftarrow 0.5$
$estimation\_1 \leftarrow zeros(initial\_distance)$ ▷ Estimations o P_1 for the accuracy of P_2 in every position
$estimation\_2 \leftarrow zeros(initial\_distance)$ ▷ Estimations o P_1 for the accuracy of P_2 in every position
$results\_1 \leftarrow$ SIMULATE_PARALLEL$(player = P_1, games = 10000, dist = 10)$ ▷ Accuracies of P_1
$results\_2 \leftarrow$ SIMULATE_PARALLEL$(player = P_2, games = 10000, dist = 10)$ ▷ Accuracies of P_2
$estimation\_1 \leftarrow results\_1$
$estimation\_2 \leftarrow results\_2$
$A \leftarrow$ CALCULATE_A$(dist = 10, estimation_1, estimation_2)$
$A\_real \leftarrow$ CALCULATE_A_REAL$(dist = 10, c_1, k_1, c_2, k_2)$
$estimated\_strategies \leftarrow$ MINMAX(A) ▷ Returns an array with dominant strategies of players
$estimated\_strategy\_1 \leftarrow estimated\_strategies[0]$
$estimated\_strategy\_2 \leftarrow estimated\_strategies[1]$
$real\_strategies \leftarrow$ MINMAX(A_real) ▷ Returns an array with dominant strategies of players
$real\_strategy\_1 \leftarrow real\_strategies[0]$
$real\_strategy\_2 \leftarrow real\_strategies[1]$

Στον παραπάνω ψευδοκώδικα βλέπουμε περίπου τη διαδικασία που ακολουθείτε από τον αλγόριθμο που περιέχεται στο Mean_Estimate.py. Συγκεριμένα τρέχοντας 10000 προσομοιώσεις για κάθε παίκτη κρατάμε το ποσοστό ευστοχίας του σε κάθε απόσταση. Αυτή θεωρούμε στα πλαίσια της εκτίμησης που κάνουν οι παίχτες πως είναι και η αναμενόμενη πραγματική τους ευστοχία. Έτσι υπολογίζουμε τον πίνακα απολαβών που θα προέκυπτε αν οι εκτιμώμενες ευστοχίες ήταν οι πραγματικές. Επίσης υπολογίζουμε και τον πραγματικό πίνακα απολαβών σύμφωνα με τις αρχικοποιήσεις μας. Τέλος βρίσκουμε τις βέλτιστες στρατηγικές εφαρμόζοντας την τεχνική minmax πάνω στους πίνακες απολαβών και συγκρίνουμε τα αποτελέσματα.

Γενικά η συγκεκριμένη προσέγγιση παρότι επιτρέπει στους παίκτες να εκτιμήσουν με ακρίβεια το σημείο ισορροπίας του παίγνιου, απαιτείται να παίξουν αρκετά παίγνια, γεγονός που ανεβάζει αρκετά το χρόνο εκτέλεσης της προσομοίωσης. Για το λόγο αυτό παραλληλοποίησαμε τη διαδικασία και ο απαιτούμενος χρόνος εκτέλεσης είναι μικρός.

2.2.3 Αποτελέσματα

Για να αξιολογήσουμε την απόδοση του αλγορίθμου μας χρησιμοποιήσαμε το κριτήριο της σχετικής εντροπίας. Συγκεκριμένα, υπολογίζουμε το μέτρο του διανύσματος σχετικής εντροπίας της εκτιμώμενης ευστοχίας** και της πραγματικής τιμής της ευστοχίας όπως προκύπτει από τον τύπο στο 1.1 για κάθε παίκτη. Και οι τέσσερις αυτές τιμές είναι διανύσματα πιθανότητων, κάθε στοιχείο δίνει την πιθανότητα ευστοχίας για κάθε τιμή της απόστασης μεταξύ των δύο παικτών. Μεταβάλλοντας τον αριθμό των επαναλαμβανόμενων παιχνιδιών έχουμε τα εξής αποτελέσματα:

**όπως αυτή προκύπτει από τον ψευδοκώδικα του 2.2.2, και συγκεκριμένα για τις μεταβλητές $estimation\_1$ και $estimation\_2$ αντίστοιχα για τους P_1 και P_2

Παίκτης	Αριθμός επαναλήψεων παιγνίου	Μέτρο Σχετικής Εντροπίας
P_1	100	1.8
P_2	100	1.55
P_1	1000	0.56
P_2	1000	1.09
P_1	10000	0.34
P_2	10000	0.21

Παρατηρούμε επομένως πως με την αύξηση του αριθμού επαναλήψεων του παιγνίου μειώνεται σημαντικά το μέτρο της σχετικής εντροπίας. Επίσης μειώνεται και η διακύμανση μεταξύ των μετρήσεων καθώς παρατηρείται πως έχοντας δεδομένα για περισσότερα παιχνίδια έχουμε καλύτερες εκτιμήσεις ευστοχίας και για τους 2 παίκτες με σημαντικά χαμηλό μέτρο σχετικής εντροπίας. Αντίθετα με μικρό αριθμό δεδομένων τυχαίνει κάποιες φορές να έχουμε αρκετά δεδομένα μόνο για ένα παίκτη(π.χ. γιατί αυτός πυροβόλησε στα περισσότερα παιχνίδια) και πολύ λιγότερα για τον άλλο παίκτη, με αποτέλεσμα το μέτρο της σχετικής εντροπίας να είναι αρκετά μεγαλύτερο στον ένα παίκτη σε σύγκριση με τον άλλο. Συχνά βέβαια έχοντας λίγα δεδομένα προκύπτει μεγάλο μέτρο σχετικής εντροπίας και για τους 2 παίκτες.

2.3 Γνωστή συνάρτηση-Άγνωστη παράμετρος k

2.3.1 Περιγραφή

Στη συνέχεια εξετάζουμε την εξής περίπτωση:

- Κάθε παίκτης γνωρίζει τη μορφή της συνάρτησης ευστοχίας(κοινή και για τους 2 παίκτες). Επίσης γνωρίζει το σύνολο των παραμέτρων(c, k) της δικής του συνάρτησης ευστοχίας αλλά γνωρίζει μόνο την παράμετρο c της συνάρτησης ευστοχίας του αντιπάλου του.

- Κάθε παίκτης διαθέτει βέβαια μια a priori γνώση για την παράμετρο k της συνάρτησης ευστοχίας του αντιπάλου του. Η γνώση αυτή είναι στη μορφή ενός συνεχούς διαστήματος τιμών εντός του οποίου βρίσκεται η πραγματική τιμή της παραμέτρου.

- Στόχος κάθε παίκτη είναι κάνει εκτίμηση της τιμής της παραμέτρου k του αντιπάλου του και κατα επέκταση να κάνει μια εκτίμηση της συνάρτησης ευστοχίας του αντιπάλου του αντικάθιστώντας στον τύπο της συνάρτησης ευστοχία που βρίσκεται στο 1.1 τη παράμετρο k που υπολόγισε(η c είναι γνωστή) σύμφωνα με τη διαδικασία που περιγράφεται στο 2.3.2

- Προσομοιώνουμε πολλαπλά παίγνια της μορφής που αναφέτεραι στο 1.1. Σε κάθε τέτοιο παίγνιο η στρατηγική που ακολουθεί κάθε παίκτης είναι η βέλτιστη σύμφωνα με την εκτίμηση του για την ευστοχία του αντιπάλου του. Η εκτίμηση αυτή αξιοποιοεί τη γνώση που έχει αποκτήσει ο παίκτης τόσο a priori, όσο και κατα τη διάρκεια των επαναλαμβανόμενων παιγνίων στα οποία συμμετέχει. Συγκεκριμένα οι παίκτες κάνουν τις εκτιμήσεις τους, και με βάση τις εκτιμήσεις αυτές προκύπτει ένα βέλτιστο σετ στρατηγικών για κάθε παίκτη[†]. Η εκτίμηση των βέλτιστων σετ στρατηγικών για κάθε παίκτη γίνεται ως εξής: Υπολογίζεται ο αναμενόμενος πίνακας απολαβών του παίγνιου σύμφωνα με τις εκτιμώμενες παραμέτρους που διαθέτει ο παίκτης. Στη συνέχεια χρησιμοποιώντας την μέθοδο minmax στον πίνακα αυτό προκύπτει το σετ βέλτιστων στρατηγικών. Ο κάθε παίκτης ακολουθεί τη στρατηγική που του αναλογεί από το σετ που εκτίμησε.

- Αφου επιλέξουν τις στρατηγικές τους οι παίκτες παίζουν 100 παίγνια χρησιμοποιώντας τις στρατηγικές αυτές. Στη συνέχεια ανανεώνουν τις εκτιμήσεις τους για τις παραμέτρους του αντιπάλου σύμφωνα με το 2.3.2. Η διαδικασία αυτή(100 παίγνια και ανανέωση εκτιμήσεων) επαναλαβάνεται κατα μέγιστο

[†]Σκεφτείτε πως ο κάθε παίκτης έχει μια διαφορετική εκτίμηση για τις παραμέτρους του παιγνίου οπότε και καταλήγει σε διαφορετικό σετ σταρτηγικών

20 φορές. Αν ένας παίκτης παρατηρεί πως η ανανέωση της εκτίμησης του k μετα από κάθε γύρο 100 παιγνίων είναι μικρή τότε θεωρεί την εκτίμηση του ικανοποιητική και σταματάει να προσπαθεί να "μάθει" κάτι παραπάνω. Στην περίπτωση που και οι δυο παίκτες σταματήσουν να "μαθαίνουν" η προσομοίωση τερματίζει.

- Στο τέλος της προσομοίωσης εξετάζουμε τρεία βασικά στοιχεία για να αξιολογήσουμε το πόσο καλά η προσέγγιση αυτή λειτούργησε. Πρώτο στοιχείο είναι το σετ στρατηγικών στο οποίο καταλήγουν οι δυο παίκτες ώστε να διαπιστώσουμε αν προσεγγίζει το πραγματικό βέλτιστο σετ στρατηγικών. Δεύτερο στοιχείο είναι η τελική εκτίμηση των παραμέτρων k που έκανε ο κάθε παίκτης για τον αντιπαλό του, όπου εξετάζουμε πόσο κοντά βρίσκεται στην πραγματική τιμή η τελική εκτίμηση. Τρίτο στοιχείο είναι οι σχετική εντροπία των δινανυσμάτων ευστοχίας που προκύπτουν για το πραγματικό και για το εκτιμώμενο k αντίστοιχα.

2.3.2 Εκτίμηση της παραμέτρου k

Κάθε παίκτης έχει στη διάθεση του ένα συνεχές διάστημα τιμών εντός του οποίου γνωρίζει πως βρίσκεται η πραγματική τιμής της εκτίμησης της παραμέτρου k του αντιπάλου του. Αρχικά το διάστημα αυτό προέρχεται από μια a priori γνώση. Στόχος της μεθόδου ανανέωσης της εκτίμησης της παραμέτρου είναι να περιορίσει το διάστημα αυτό σε εύρος αξιοποιώντας την νέα γνώση που αποκτά καθώς παίζει. Η εκτίμηση της παραμέτρου k που χρησιμοποιείται για την εύρεση του σετ βέλτιστων στρατηγικών είναι το μέσο του διαστήματος αυτού.

Όταν κάθε παίκτης επιλέξει την εκτιμώμενη βέλτιστη στρατηγική του(στην αρχή με τη χρήση του a priori γνωστού διαστήματος) τότε όπως έχουμε αναφέρει και πιο πάνω επαναλαμβάνεται ένα παίγνιο με αυτές τις στρατηγικές 100 φορές. Με τον τρόπο αυτό κάθε παίκτης έχει τη δυνατότητα να αξιολογήσει την γνώση του για την τιμή της παραμέτρου k του αντιπάλου του και να προβεί σε ανανέωση της. Η διαδικασία έχει ώς εξής: Στα επαναλαμβανόμενα παίγνια με σταθερές τις στρατηγικές ένα παίκτης αναγκαστικά θα πυροβολεί πρώτος ας τον ονομάσουμε για ευκολία στην ανάλυση μας παίκτη A(ενώ θα αναφερόμαστε στον αντίπαλο του ως παίκτη B). Το γεγονός αυτό επιτρέπει στον παίκτη B να κάνει μια εκτίμηση της ευστοχίας του παίκτη A, έστω P η εκτίμηση αυτή. Λαμβάνοντας υπόψιν και το στατιστικό λάθος (το οποίο θεωρήσαμε πως είναι ± 0.1), ο παίκτης B υπολογίζει ένα νέο συνεχές διάστημα τιμών στο οποίο αναμένει να βρίσκεται η τιμή της παραμέτρου k του αντιπάλου. Το άνω όριο του διαστήματος προκύπτει για τιμή ευστοχίας $(P - 0.1)$. Ο τύπος που επιτρέπει τον υπολογισμό αυτού του άνω ορίου είναι ο εξής:

$$(P-0.1) = \frac{c}{d^{k_{max}}} \Rightarrow d^{k_{max}} = \frac{c}{(P-0.1)} \Rightarrow k_{max} \cdot \log d = \log(\frac{c}{(P-0.1)}) \Rightarrow k_{max} = \frac{\log(\frac{c}{(P-0.1)})}{\log d}$$

Αν θέλουμε υπολογίσουμε το κάτω όριο k_{min} απλά αντικαθιστούμε στη θέση του $(P - 0.1)$ την τιμή $(P + 0.1)$.

Έχοντας υπολογίσει το διάστημα $[k_{min}, k_{max}]$ και έχοντας στη διάθεση του το προηγούμενο διάστημα που γνωρίζει, έστω $[a, b]$, ο παίκτης B προσπαθεί να περιορίσει το εύρος του διαστήματος στο οποίο ψάχνει την πραγματική τιμή της παραμέτρου k. Για να το κάνει αυτό ακολουθεί τον εξής αλγόριθμο:

1. Αν $k_{min} > a$ και $k_{min} < b$ τότε το νέο διάστημα είναι $[k_{min}, (\text{άνω όριο})]$
2. Αν $k_{max} > a$ και $k_{max} < b$ τότε το νέο διάστημα είναι $[(\text{κάτω όριο}), k_{max}]$

Πρακτικά δηλαδή βρίσκουμε την τομή του παλαιού διαστήματος με του νέου ώστε να περιορίσουμε το εύρος του διαστήματος που λαμβάνει υπόψιν του ο παίκτης B. Με τον τρόπο αυτό αξιοποιούμαι την πρότερη γνώση που προέκυψε μέσω της εμπειρίας του παίκτη.

Algorithm 2 Εκτίμηση της παραμέτρου k

$initial\_distance = 10, c_1 = 1, k_1 = 0.7, c_2 = 1, k_2 = 0.4, stops1 = stop2 = 0$

$prior\_k1 = [k1\_min\_init, k1\_max\_init]$ ▷ a priori knowledge of P_2 for k_1
$prior\_k2 = [k2\_min\_init, k2\_max\_init]$ ▷ a priori knowledge of P_1 for k_2
$num\_games = 20$ ▷ Number of different game that'll be played
$num\_samples = 100$ ▷ Number of samples from each game
for game **in** range(num_games) **do**
 for sample **in** range($num\_samples$) **do**
  if $(stop1 != 2 | stop2 != 2)$ **then**
   $estimation\_k1 = \frac{(prior\_k1[0]+prior\_k1[1])}{2}$

   $estimation\_k2 = \frac{(prior\_k2[0]+prior\_k2[1])}{2}$

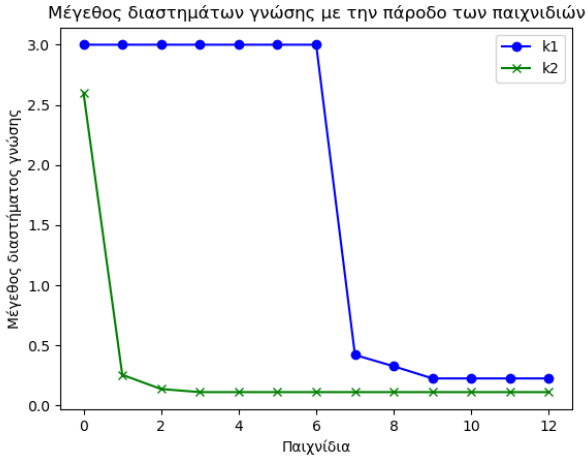
   $wins1$,$wins2$,$distance \leftarrow$ DUEL($k_1, c_1, k_2, c_2, estimation\_k1, estimation\_k2, stop1, stop2$)
   if ($wins1$ & distance!=1) **then** $p_1 = wins1/num\_samples$
    if $p1 > 0.1$ **then**
     $k1_{max} = \frac{\log(\frac{c_1}{(p1-0.1)})}{\log distance}$
     $k1_{min} = \frac{\log(\frac{c_1}{(p1+0.1)})}{\log distance}$
    else
     $k1_{max} = \frac{\log(\frac{c_1}{(p1-\frac{p1}{2})})}{\log d}$
     $k1_{min} = \frac{\log(\frac{c_1}{(p1+0.1)})}{\log distance}$

    if $k1_{min} > prior\_k1[0]$ & $k1_{min} < prior\_k1[1]$ **then**
     $prior\_k1[0] = k1_{min}$
    if $k1_{max} > prior\_k1[0]$ & $k1_{max} < prior\_k1[1]$ **then**
     $prior\_k1[1] = k1_{max}$
    $new\_estimate\_k1 = \frac{(prior\_k1[0]+prior\_k1[1])}{2}$
    if $absolute(new\_estimate\_k1 - estimation\_k1) < 0.001$ **then**
     $stop1 = stop1 + 1$
   if ($wins2$ & distance!=1) **then** $p_2 = wins2/num\_samples$
    if $p2 > 0.1$ **then**
     $k2_{max} = \frac{\log(\frac{c_2}{(p2-0.1)})}{\log d}$
     $k2_{min} = \frac{\log(\frac{c_2}{(p2+0.1)})}{\log distance}$
    else
     $k2_{max} = \frac{\log(\frac{c_2}{(p2-\frac{p2}{2})})}{\log distance}$
     $k2_{min} = \frac{\log(\frac{c_2}{(p2+0.1)})}{\log distance}$

    if $k2_{min} > prior\_k2[0]$ & $k2_{min} < prior\_k2[1]$ **then**
     $prior\_k2[0] = k2_{min}$
    if $k2_{max} > prior\_k2[0]$ & $k2_{max} < prior\_k2[1]$ **then**
     $prior\_k2[1] = k2_{max}$
    $new\_estimate\_k2 = \frac{(prior\_k2[0]+prior\_k2[1])}{2}$
    if $absolute(new\_estimate\_k2 - estimation\_k2) < 0.001$ **then**
     $stop2 = stop2 + 1$

2.3.3 Αποτελέσματα

Η υλοποίηση του παραπάνω βρίσκεται στο αρχείο Unknown_k.py με αρχική απόσταση ίση με 10, παραμέτρους $c1 = c2 = 1, k1 = 0.7, k2 = 0.4$. Επίσης δίνετε στον παίκτη 1 αρχική γνώση για την παράμετρο $k2$ το διάστημα $[0.0, 2.6]$ και στον παίκτη 2 αρχική γνώση για την παράμετρο $k2$ το διάστημα $[0.0, 3.0]$. Παρατηρούμε πως το αποτέλεσμα της εκτίμησης είναι αρκετά ικανοποιητικό με βάση τα κριτήρια που θέσαμε στο 2.3. Συγκεκριμένα, οι εκτιμήσεις για των παραμέτρων γίνεται με πολύ μικρή απόκλιση ($k1 = 0.67$ και $k2 = 0.41$). Επίσης οι τελικώς εκτιμώμενο σημείο ισορροπίας είναι το $(4, 7)$ ενώ το πραγματικό είναι το $(4, 9)$. Στο παρακάτω γράφημα φαίνεται πως περιορίζεται το εύρος των διαστημάτων γνώσης με την πάροδο των παιχνιδιών[‡]



Επίσης, η σχετική εντροπία του διανύσματος πιθανότητας ευστοχίας που προκύπτει μεταξύ του διανύσματος που προκύπτει γνωρίζοντας την πραγματική τιμή των k_1 και k_2 και του διανύσματος που προκύπτει χρησιμοποιώντας τις τελικές εκτιμήσεις του αλγορίθμου 2 είναι αντίστοιχα 0.013 και 0.00013. Παρατηρούμαι πως οι τιμές της σχετικής εντροπίας είναι σημαντικά μικρότερη σε αυτή τη μέθοδο μάθησης σε σύγκριση με τη μέθοδο 2.2.2.

Κρατώντας σταθερές τις πραγματικές τιμές των k παραμέτρων και μεταβάλλοντας το a priori διάστημα γνώσεις δεν παρατηρούμαι σημαντικές διαφορές στο μέτρο της σχετικής εντροπίας. Μπορούμε να πούμε πως το a priori διάστημα γνώσης δεν επηρεάζει την απόδοση του αλγορίθμου 2 αν δεν είναι παραπλανητικό. Παραπλανητικό θα ήταν για παράδειγμα ένα διάστημα το οποίο δεν περιλαμβάνει την πραγματική τιμή του k εντός του. Αντίθετα, ο πειραματισμός με τις πραγματικές τιμές των παραμέτρων k, μπορεί εύκολα να οδηγήσεις τον αλγόριθμο στην ειδική περίπτωση 2.3.4. Επομένως αυτό περιορίζει σε ένα βαθμό την γενικότητα της λύσης που προσφέρει ο αλγόριθμος 2 για την αποτελεσματική σύγκλιση των εκτιμήσεων και των δυο παικτών.

2.3.4 Στρατηγική για την σύγκλιση και των δυο παικτών

Για να είμαστε σίγουροι πως και οι 2 παίκτες θα καταλήξουν σε μια καλή εκτίμηση έχουμε φροντίσει αν ένας παίκτης, έστω ο A, έχει φτάσει σε ικανοποιητική εκτίμηση της τιμής της παραμέτρου k του B τότε ο B επιλέγει να τον αφήσει να πυροβολήσει πρώτος ώστε να μάθει περισσότερα για την ευστοχία του A. Πρακτικά τότε ο B επιλέγει να πυροβολήσει σε κατώφλι μειωμένο κατα μια μονάδα απόστασης από το κατώφλι του B. Ο A συνεχίζει να πυροβολεί στη θέση που πυροβολούσε καθώς πιστεύει πως είναι το βέλτιστο που μπορεί να κάνει καθώς θεωρεί πως γνωρίζει ικανοποιητικά όλες τις παραμέτρους του

[‡]Παιχνίδια εννούμε στη συγκεκριμένη περίπτωση τα διακριτά σετ 100 παιχνιδιών πριν την ανανέωση των παραμέτρων

παίγνιου. Συγκεκριμένα όταν η ανανέωση στην εκτίμηση ενός παίκτη είναι πολύ μικρή(< 0.001) για πάνω από 2 φορές(δηλαδή όταν η μεταβλητή $stop1$ ή η $stop2$ ισούται με 2) τότε ο παίκτης αυτός σταματάει να "μαθαίνει" και πρακτικά σταθεροποιεί την στρατηγική του. Στην περίπτωση που ισχύει $stop1 = stop2 = 2$ τότε τερματίζει ο αλγόριθμος.

Υπάρχει μια ειδική περίπτωση όπου η συγκεριμένη μέθοδος δεν θα δουλέψει: Έστω πως ο παίκτης που παίζει πρώτος, έστω και πάλι ο Α, καταλήξει πως η βέλτιστη στρατηγική του είναι να θέσει το κατώφλι στην απόσταση 2, δηλαδή πρακτικά να μην προβολεί ποτέ πρώτος. Σε αυτή την περίπτωση ο παίκτης Β δεν μπορεί να κάνει κάτι για να αναγκάσει τον παίκτη Α να πυροβολήσει ώστε να μάθει κάτι περισσότερο για την ευστοχία του.

2.4 Γνωστή συνάρτηση-Άγνωστοι παράμετροι k και c

2.4.1 Περιγραφή

Στη συνέχεια δοκιμάσαμε να επεκτείνουμε την περίπτωση 2.3 και να θεωρήσουμε και την παράμετρο c ώς άγνωστη και διαφορετική για κάθε παίκτη. Πλέον ο κάθε παίκτης δεν καλείται απλά να κάνει μια σωστή εκτίμηση μόνο για το k αλλά και για την παράμετρο c. Η λογική που ακολουθούμε για την εκτίμηση του c είναι ακριβώς η ίδια με αυτή της περίπτωσης 2.3, δηλαδή με τομές διαστημάτων. Ο κάθε παίκτης φυσικά έχει μια a priori γνώση για την τιμή της c του αντιπάλου(ένα διάστημα, όπως και για την k). Η μέθοδος με την οποία ο ένας παίκτης αλλάζει την στρατηγική του όταν ο αντίπαλος του σταθεροποιεί την στρατηγική του είναι η ίδια με αυτή που χρησιμοποιείται στο 2.2. Βάση για το να σταματήσει να ένας παίκτης να μαθαίνει είναι η ανανέωση της παραμέτρου k και όχι της c. Παρά το γεγονός αυτό φαίνεται πως η σύγκλιση των διαστημάτων και για την παράμετρο c είναι ικανοποιητική.

2.4.2 Εκτίμηση παραμέτρων και ψευδοκώδικας

Η διαφοροποίηση στην ανανέωση των παραμέτρων σε σύγκριση με το 2.3 είναι ότι όταν ο παίκτης λαμβάνει τα δεδομένα από τα 100 παίγνια με συγκεριμένο σετ στρατηγικών αρχικά χρησιμοποιώντας την εκτίμηση για την παράμετρο k που ήδη έχει από πριν, υπολογίζει μια αναμενόμενη τιμή για το c του αντιπάλου χρησιμοποιώντας τη σχέση $c = P \cdot d^k$. Στη συνέχεια υπολογίζει το άνω και κάτω όριο για την παράμετρο k με τον ίδιο τρόπο όπως και στο 2.3 χρησιμοποιώντας την τιμή της c που εκτίμησε προηγουμένως. Τέλος ανανεώνει το διάστημα για την παράμετρο c χρησιμοποιώντας τον τύπο $c_{min} = P \cdot d^{k_{min}}$ για τον υπολογισμό του κάτω ορίου και τον τύπο $c_{max} = P \cdot d^{k_{max}}$. Η ανανέωση γίνεται με τον εξής αλγόριθμο δεδομένου ότι το προηγούμενο διάστημα που γνωρίζει είναι το $[a, b]$:

1. Αν $c_{min} > a$ και $c_{min} < b$ τότε το νέο διάστημα είναι $[c_{min}, (\text{άνω όριο})]$
2. Αν $c_{max} > a$ και $c_{max} < b$ τότε το νέο διάστημα είναι $[(\text{κάτω όριο}), c_{max}]$

Ο ψευδοκώδικας που υλοποιεί τα παραπάνω φαίνεται στην επόμενη σελίδα

Algorithm 3 Εκτίμηση των παραμέτρων k και c

$initial\_distance = 10, c_1 = 1, k_1 = 0.7, c_2 = 1, k_2 = 0.4, stops1 = stop2 = 0, num\_games = 20, num\_samples = 100$
$prior\_k1 = [k1\_min\_init, k1\_max\_init]$ ▷ a priori knowledge of P_2 for k_1
$prior\_k2 = [k2\_min\_init, k2\_max\_init]$ ▷ a priori knowledge of P_1 for k_2
$prior\_c1 = [c1\_min\_init, c1\_max\_init]$ ▷ a priori knowledge of P_2 for c_1
$prior\_c2 = [c2\_min\_init, c2\_max\_init]$ ▷ a priori knowledge of P_1 for c_2
for game **in** range(num_games) **do**
 for sample **in** range($num\_samples$) **do**
  if $(stop1 != 2 | stop2 != 2)$ **then**
   $estimation\_k1 = \frac{(prior\_k1[0]+prior\_k1[1])}{2}$
   $estimation\_k2 = \frac{(prior\_k2[0]+prior\_k2[1])}{2}$
   $estim\_c1 = \frac{(prior\_c1[0]+prior\_c1[1])}{2}$
   $estim\_c2 = \frac{(prior\_c2[0]+prior\_c2[1])}{2}$
   $wins1, wins2, distance = \text{DUEL}(k_1, c_1, k_2, c_2, estimation\_k1, estimation\_k2, estim\_c1, estim\_c2, stop1, stop2)$
   if ($wins1$ & distance!=1) **then** $p_1 = wins1/num\_samples$
    if $p1 > 0.1$ **then**
     $k1_{max} = \frac{\log(\frac{estim\_c1}{(p1-0.1)})}{\log distance}$
     $k1_{min} = \frac{\log(\frac{estim\_c1}{(p1+0.1)})}{\log distance}$
    else
     $k1_{max} = \frac{\log(\frac{estim\_c1}{(p1-\frac{p1}{2})})}{\log distance}$
     $k1_{min} = \frac{\log(\frac{estim\_c1}{(p1+0.1)})}{\log distance}$
    if $k1_{min} > prior\_k1[0]$ & $k1_{min} < prior\_k1[1]$ **then** $prior\_k1[0] = k1_{min}$
    if $k1_{max} > prior\_k1[0]$ & $k1_{max} < prior\_k1[1]$ **then** $prior\_k1[1] = k1_{max}$
    $c1_{max} = p1 \cdot distance^{(prior\_k1[1])}$
    $c1_{min} = p1 \cdot distance^{(prior\_k1[0])}$
    if $c1_{min} > prior\_c1[0]$ & $c1_{min} < prior\_c1[1]$ **then** $prior\_c1[0] = c1_{min}$
    if $c1_{max} > prior\_c1[0]$ & $c1_{max} < prior\_c1[1]$ **then** $prior\_c1[1] = c1_{max}$
    $new\_estimate\_k1 = \frac{(prior\_k1[0]+prior\_k1[1])}{2}$
    if $absolute(new\_estimate\_k1 - estimation\_k1) < 0.001$ **then** $stop1 = stop1 + 1$
   if ($wins2$ & distance!=1) **then** $p_2 = wins2/num\_samples$
    if $p2 > 0.1$ **then**
     $k2_{max} = \frac{\log(\frac{estim\_c2}{(p2-0.1)})}{\log distance}$
     $k2_{min} = \frac{\log(\frac{estim\_c2}{(p2+0.1)})}{\log distance}$
    else
     $k2_{max} = \frac{\log(\frac{estim\_c2}{(p2-\frac{p2}{2})})}{\log distance}$
     $k2_{min} = \frac{\log(\frac{estim\_c2}{(p2+0.1)})}{\log distance}$
    if $k2_{min} > prior\_k2[0]$ & $k2_{min} < prior\_k2[1]$ **then** $prior\_k2[0] = k2_{min}$
    if $k2_{max} > prior\_k2[0]$ & $k2_{max} < prior\_k2[1]$ **then** $prior\_k2[1] = k2_{max}$
    $c2_{max} = p2 \cdot distance^{(prior\_k2[1])}$
    $c2_{min} = p2 \cdot distance^{(prior\_k2[0])}$
    if $c2_{min} > prior\_c2[0]$ & $c2_{min} < prior\_c2[1]$ **then** $prior\_c2[0] = c2_{min}$
    if $c2_{max} > prior\_c2[0]$ & $c2_{max} < prior\_c2[1]$ **then** $prior\_c2[1] = c2_{max}$
    $new\_estimate\_k2 = \frac{(prior\_k2[0]+prior\_k2[1])}{2}$
    if $absolute(new\_estimate\_k2 - estimation\_k2) < 0.001$ **then** $stop2 = stop2 + 1$

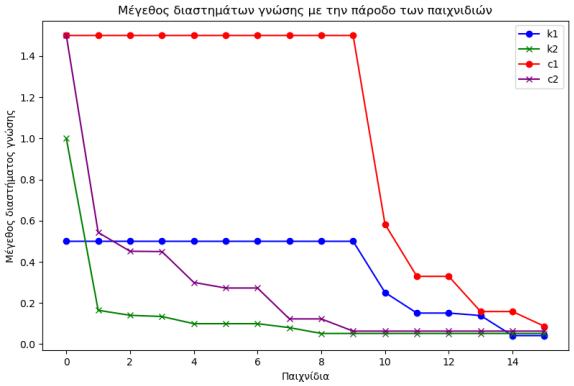
2.4.3 Αποτελέσματα

Η υλοποίηση της παραπάνω προσομοίωσης βρίσκεται στο αρχείο Unknown_kandc.py. Συγκρίνοντας τα αποτελέσματα της προσομοίωσης αυτής με τα αναμενόμενα πραγματικά αποτελέσματα παρατηρούμε πως η προσομοίωση μας είναι αρκετά ευαίσθητη σε αυθαίρετες αρχικοποιήσεις. Γενικά όμως με προσεκτική επιλογή των αρχικών παραμέτρων μπορεί να παράξει αξιόλογα αποτελέσματα, δεδομένης και της πολυπλοκότητας του προβλήματος. Συγκεριμένα με αρχικές συνθήκες $k_1 = 0.7, k_2 = 0.5, c_1 = 1.4, c_2 = 1.7$ ενώ a priori γνώσεις είναι: Για τον παίκτης 1 για την παράμετρο k_2 το διάστημα $[0.0, 1.0]$ ενώ για την παράμετρο c_2 το διάστημα $[0.5, 2.0]$. Για τον παίκτης 2 για την παράμετρο k_1 το διάστημα $[0.5, 1.0]$ ενώ για την παράμετρο c_1 το διάστημα $[0.5, 2.0]$. Το αποτέλεσμα είναι τα τελικά διαστήματα γνώσης να είναι:

- Για τον παίκτης 1 : $k_2 = [0.503, 0.532]$ και $c_2 = [1.577, 1.722]$
- Για τον παίκτης 2 : $k_1 = [0.701, 0.757]$ και $c_1 = [1.454, 1.468]$

Επομένως παρατηρούμε πως οι τελικές εκτιμήσεις είναι πολύ κοντά στις πραγματικές τιμές των παραμέτρων.

Στο παρακάτω γράφημα φαίνεται πως περιορίζεται το εύρος των διαστημάτων γνώσης με την πάροδο των παιχνιδιών[§]

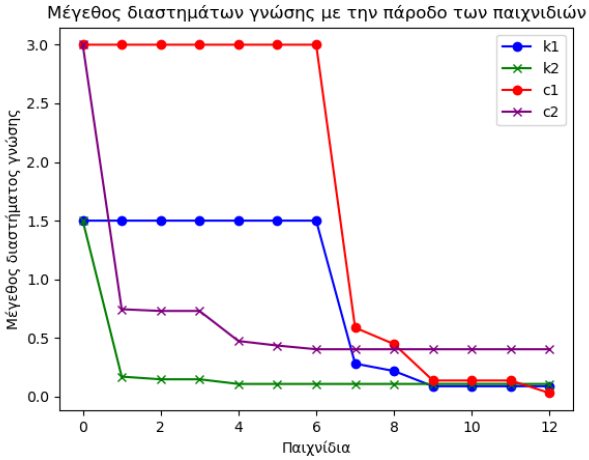


Επίσης, το τελικό εκτιμώμενο σημείο ισορροπίας είναι το (6,7) ενώ το πραγματικό είναι (4,9). Είναι φανερό πως και μικρές αποκλίσεις από τις πραγματικές τιμές οδηγεί σε μεγαλύτερη απόκλιση από το πραγματικό σημείο ισορροπίας σε σύγκριση με τη μια παράμετρο όπως είχαμε δει στο 2.3.

Χρησιμοποιώντας και το μέτρο της σχετικής εντροπίας των διανυσμάτων πιθανότητας ευστοχίας που εμφανίζεται μεταξύ του διανύσματος που προκύπτει γνωρίζοντας τις πραγματικές τιμές των παραμέτρων c, k και του διανύσματος που προκύπτει χρησιμοποιώντας τις τελικές εκτιμήσεις του αλγορίθμου 3 είναι 0.05 και 0.003 για τις ευστοχίες των P_1 και P_2. Παρατηρούμαι πως το μέτρο της σχετικής εντροπίας είναι σχετικά αυξημένο σε σχέση με αυτό που προέκυψε από τον 2. Αυτό βέβαια είναι αναμενόμενο δεδομένου πως υπάρχει εκτίμηση άρα και απόκλιση στις τιμές και των δυο παραμέτρων της συνάρτησης ευστοχίας των παικτών.

Κρατώντας σταθερές τις πραγματικές τιμές των παραμέτρων αλλά μεταβάλλοντας ελαφρώς τα a priori διαστήματα τιμών των παραμέτρων δεν παρατηρείτε σημαντική διαφορά στην απόδοση του αλγορίθμου. Συγκεριμένα για a priori διαστήματα: $k1 = [0, 1.5], c1 = [0, 3], c2 = [0, 3], k2 = [0.1.5]$ το γράφημα του εύρους των διαστημάτων γνώσης με την πάροδο των παιχνιδιών είναι το παρακάτω:

[§]Παιχνίδια εννούμε στη συγκεκριμένη περίπτωση τα διακριτά σετ 100 παιχνιδιών πριν την ανανέωση των παραμέτρων



Επίσης το μέτρο της σχετικής εντροπίας είναι 0.02 και 0.036 αντίστοιχα για τις ευστοχίες των P_1 και P_2, επομένως δεν παρατηρούμαι σημαντική διαφορά παρά την σχετική διεύρυνση των ορίων των a priori διαστημάτων.

Γενικά μπορεί κανείς να πειραματιστεί με πληθώρα επιλογών τόσο για τις πραγματικές τιμές των παραμέτρων όσο και με τα a priori διαστήματα γνώσης και μπορούν να υπάρξουν μικρές διαφοροποιήσεις όσο αφορά την σχετική εντροπία και γενικά του πόσο καλά προσεγγίζεται η πραγματική ευστοχία των παικτών αλλά τα αποτελέσματα για λογικές αρχικοποιήσεις φαίνεται να είναι ικανοποιητικά(κόντα σε αυτά των παραδειγμάτων που αναφέρθηκαν). Φυσικά καλό θα είναι μελλοντικά να γίνει και πιο διεξοδικό testing του αλγορίθμου σε μεγάλο αριθμό διαφορετικών αρχικοποιήσεων.

Το σημαντικό μειονέκτημα του αλγορίθμου 3 είναι το ίδιο με αυτό του 2 και το έχουμε αναφέρει στο 2.3.4. Μελλοντικά θα είχε ιδιαίτερο ενδιαφέρον να μελετηθούν δυνατότητες βελτίωσης της προβληματικής λειτουργίας του αλγορίθμου σε αυτές τις ειδικές περιπτώσεις.

2.5 Άγνωστη συνάρτηση -Χρήση κατανομών

2.5.1 Περιγραφή και ψευδοκώδικας

Τέλος εξετάσαμε την περίπτωση που ο κάθε παίκτης δεν προσπαθεί να βρεί τις παραμέτρους της συνάρτησης ευστοχίας αλλά να βρεί τη βέλτιστη στρατηγική. Συγκεκριμένα κάθε παίκτης ξεκινά επιλέγοντας κάθε στρατηγική με ίση πιθανότητα, στόχος είναι στο τέλος να επιλέγει μια και μόνο στραηγική με υψηλή πιθανότητα και αυτή να είναι η βέλτιστη. Αυτό επιυγχάνεται με ένα απλό μηχανισμό επιβράβευσης-τιμωρίας. Συγκεκριμένα αυξάνεται η πιθανότητα να επιλέξει κάθε παίκτης μια στρατηγική όταν αυτή καταλήγει να είναι νικηφόρα σε ένα παίγνιο, ενώ αντίθετα μειώνεται η πιθανότητα να επιλέξει κάποια στρατηγική όταν αυτή τον οδηγεί σε ήττα.

Πιο συγκριμένα, αρχικά κάθε παίκτης διαθέτει ένα διάνυσμα πιθανοτήτων που στην αρχή εκφράζει ομοιόμορφη κατανομή μεταξύ των πιθανών στρατηγικών. Κάνοντας δειγματοληψία σε αυτό το διάνυσμα επιλέγει στο ξεκίνημα κάθε παιγνίου την στρατηγική του. Αν η στρατηγική που επέλεξε ο παίκτης τον οδηγήσει σε νίκη, τότε η πιθανότητα επιλογής της συγκεκριμένης σταρτηγικής αυξάνεται πολλαπλασιάζοντας την αντίστοιχη θέση του διανύσματος πιθανοτήτων κατα ένα σταθερό παράγοντα μεγαλύτερο από 1. Αντίθετα, όταν η επιλογή του τον οδηγεί σε ήττα η πιθανότητα επιλογής της στρατηγικής πολλαπλασιάζεται με ένα σταθερό παράγοντα μικρότερο του 1. Τέλος, το νέο διάνυσμα πιθανοτήτων που θα προκύψει κανονικοποιείται έτσι ώστε να εκφράζει και πάλι κατανομή πιθανοτήτων(το άθροισμα των στοιχείων του

να ισούται με 1). Παρουσιάζουμε τον παραπάνω αλγόριθμος και στον σχετικό ψευδοκώδικα παρακάτω.

Algorithm 4 Εκτίμηση στρατηγικής μέσω κατανομών

$init\_dist = 10, c_1 = 1, k_1 = 0.7, c_2 = 1, k_2 = 0.4$

▷ Every index represents a strategy of the player, for example $p1\_distribution[i]$ represent the probability player 1 shoots in for distance less than i steps

$p1\_distribution = initial\_distance * [\frac{1}{initial\_distance}]$

$p2\_distribution = initial\_distance * [\frac{1}{initial\_distance}]$
$num\_games = 3000$ ▷ Number of different game that'll be played
for game **in** range(num_games) **do**

$p1\_strategy = \text{RANDOM\_CHOICE}(p1\_distribution)$ ▷ Returns random index on the distribution
$p2\_strategy = \text{RANDOM\_CHOICE}(p2\_distribution)$ ▷ Returns random index on the distribution

$win1, win2 = \text{DUEL}(init\_dist = 10, p1\_strategy, p2\_strategy, c_1 = 1, k_1 = 0.7, c_2 = 1, k_2 = 0.4)$

if $win1$ **then** $p1\_distribution[p1\_strategy] = p1\_distribution[p1\_strategy] \cdot 1.1$
else $p1\_distribution[p1\_strategy] = p1\_distribution[p1\_strategy] \cdot 0.8$

if $win2$ **then** $p2\_distribution[p2\_strategy] = p2\_distribution[p1\_strategy] \cdot 1.1$
else $p2\_distribution[p2\_strategy] = p2\_distribution[p2\_strategy] \cdot 0.8$

$p1\_distribution = \text{NORMALIZE}(p1\_distribution)$
$p2\_distribution = \text{NORMALIZE}(p2\_distribution)$

$p1\_final = \text{INDEXOFMAX}(p1\_distribution)$ ▷ Returns the index of max element of distribution
$p2\_final = \text{INDEXOFMAX}(p2\_distribution)$

$A = \text{CALCULATE\_A}(init\_dist = 10, p1\_strategy, p2\_strategy, c_1 = 1, k_1 = 0.7, c_2 = 1, k_2 = 0.4)$

$real\_strategies \leftarrow \text{MINMAX}(A)$ ▷ Returns an array with dominant strategies of players
$real\_strategy\_1 \leftarrow real\_strategies[0]$
$real\_strategy\_2 \leftarrow real\_strategies[1]$

2.5.2 Αποτελέσματα

Η παραπάνω προσέγγιση υλοποιείται στο αρχείο Distributions.py. Συγκεκριμένα, όταν ένας παίκτης επιλέγει μια στρατηγική και αυτή τον οδηγεί σε νίκη, τότε η πιθανότητα επιλογής της πολλαπλασιάζεται με ένα παράγοντα μεγαλύτερο από 1 και έχουμε επιλέξει να ισούται με 1.1. Αντίθετα όταν οδηγείται σε ήττα, η πιθανότητα επιλογής της στρατηγικής μειώνεται κατά 20%. Η επιλογή τον παραπάνω παραγόντων "επιβράβευσης" και "τιμωρίας" γίνανε έπειτα από ορισμένες δοκιμές καθώς φάνηκε να λειτουργούν λίγο καλύτερα από άλλες αρχικοποιήσεις. Παρατηρούμαι πως στην τελική κατανομή που προκύπτει μετά από σχετικά μεγάλο αριθμό επαναλήψεων(> 1000) μια από τις πιθανές στρατηγικές έχει σημαντικά μεγαλύτερη πιθανότητα να επιλεγεί. Θεωρητικά έχουμε μια σύγκλιση θα λέγαμε στην βέλτιστη στρατηγική. Παρόλα αυτά ο συγκεκριμένος αλγόριθμος χρειάζεται σημαντικές βελτιώσεις καθώς εμφανίζει μεγάλη τυχαιότητα

στα αποτελέσματα του. Αυτό οφείλετε πιθανότατα στο γεγονός πως οι αρχικές επιλογές που θα γίνουν(με τυχαίο αρχικά τρόπο) καθορίζουν σημαντικά την μετέπειτα απόδοση του αλγορίθμου. Μικρές μεταβολές στους παράγοντες "επιβράβευσης-τιμωρίας" δεν φαίνεται να βελτιώνουν ιδιαίτερα την κατάσταση αυτή, τουλάχιστον με τις δοκιμές που έγιναν στα παλίσια της εργασίας αυτής. Παραλλαγές και βελτιώσεις του αλγορίθμου 4 ενδεχομένως να μπορούν να προσφέρουν αξιόπιστα αποτελέσματα. Αφήνουμε το ενδεχόμενο αυτό ανοιχτό προς περαιτέρω μελέτη.