ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ

# [PLH 423] - Reinforcement Learning & Dynamic Optimization

## Αναφορά 1ης φάσης τελικού project : Poker Playing Agent


***Διδάσκων :***
Θρασύβουλος Σπυρόπουλος

***Group 22 :***
Γεώργιος Σκουλάς : Α.Μ. 2018030148
Ιωάννης Περίδης : Α.Μ. 201803069


1η Ιουλίου 2023

## Περιεχόμενα:

# 1 Εισαγωγή:

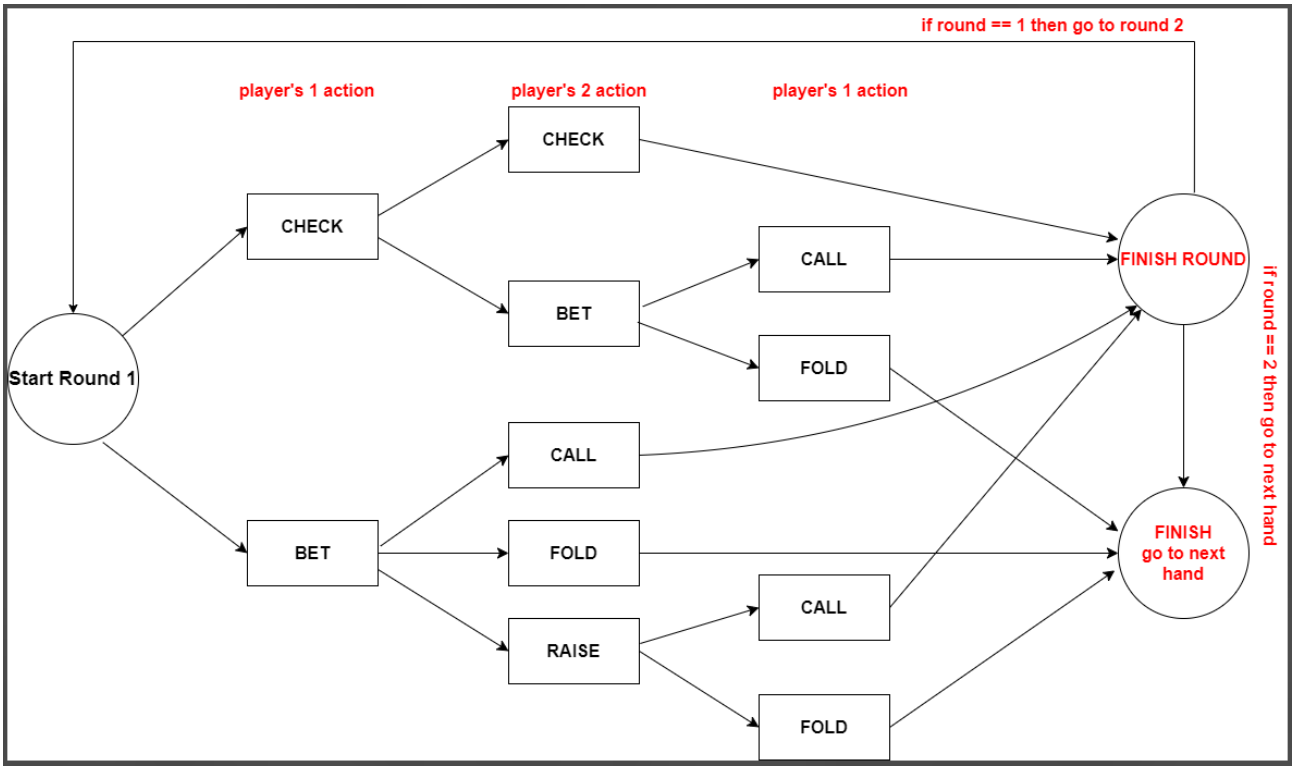
Στην εργασία αυτή, μας ζητήθηκε η δημιουργία ενός περιβάλλοντος που θα υποστηρίζει την λειτουργία και την γραφική αναπαράσταση μιας απλουστευμένης εκδοχής του παιχνιδιού Heads Up , Limit ,Texas Holde'm Poker. Έπειτα, στο περιβάλλον αυτό, γίνεται η υλοποίηση δύο διαφορετικών αλγορίθμων ενισχυτικής μάθησης. Το Policy Iteration με τον τύπο του αντιπάλου να είναι γνωστός και το Q-Learning με τον τύπου του αντιπάλου να είναι άγνωστος. Οι διαφορετικοί αντίπαλοι agents που μοντελοποιήθκαν , κατηγοριοποιούνται ανάλογα με την στρατηγική που ακολουθεί ο καθένας τους στο παιχνίδι, επιθετική ή παθητική. Ακόμη, παραδίδονται αναλυτικά αποτελέσματα και γραφικές παραστάσεις, τα οποία αναδεικνύουν και συγκρίνουν το μέσο reward του κάθε αλγορίθμου για κάθε διαφορετικό αντίπαλο. Τέλος, γίνεται επίδειξη της ταχύτητας σύγκλισης των αλγορίθμων στην βέλτιστη λύση και σχολιασμός των αποτελεσμάτων της απόδοσής τους.

# 2 Υλοποίηση Περιβάλλοντος Παιχνιδιού:

### Βασική Λειτουργία:

Στην απλοποιημένη εκδοχή αυτή του πόκερ: Παίζουν 2 παίκτες, οι οποίοι υποχρεούνται να ποντάρουν 0,5 blinds στην αρχή κάθε χεριού και το μέγιστο ποντάρισμα τους είναι 1 blind. Η τράπουλα έχει 20 συνολικά φύλλα, συγκεκριμένα τα 4 suits (club,spade,heart,diamond) των καρτών T,J,Q,K,A. Στους παίκτες μοιράζεται 1 φύλλο στο χέρι τους και 2 στο board. Οι γύροι ποντάρισμάτων είναι 2 preflop και flop και μόνο στον 2$^{ο}$ γύρω , εμφανίζονται οι κάρτες του board. Οι εφικτοί συνδυασμοί των καρτών είναι 3 και η δύναμη τους αυξάνεται με την παρακάτω σειρά high card (κανένα ζεύγος), one pair ( 1 ζεύγος καρτών) και triplet (3 ίδιες κάρτες). Τα πιθανά actions είναι check (δράση περνάει στον αντίπαλο), bet 1 blind (ποντάρισμα), call 1 blind (αντιστοίχιση ποντάρίσματος του αντιπάλου), raise 1 blind (αντιστοίχιση του ποντάρίσματος του αντιπάλου και αύξηση κατά 1 blind), fold (τέλος της δράσης με δική σου ήττα). Το παιχνίδι τελειώνει όταν το αρχικό stack ενός από τους 2 παίκτες φτάσει τα 0 blinds. Παρακάτω φαίνεται το διάγραμμα όλων των επιτρεπτών ακολουθιών των ενεργειών των παικτών , σε μια παρτίδα.

***Actions Flow Diagram representing one game hand:***



### Κώδικας:

Για την υλοποίηση του κώδικα δεν χρησιμοποιήθηκε το RLcard, δημιουργήθηκε το περιβάλλον από την αρχή. Αρχικά, για την μοντελοποίηση των παικτών, υλοποιήθηκε η ***κλάση player***, η οποία περιέχει το όνομα, το χέρι, το stack, και την ενέργεια που εκτελεί ένας παίκτης. Το πρόγραμμα ξεκινάει δημιουργώντας 2 παίκτες και την τράπουλα με τις συναρτήσεις ***create_players*** και ***create_deck*** αντίστοιχα. Έπειτα, εισέρχεται σε έναν βρόγχο while, ο οποίος εκτελείται μέχρι να μηδενιστεί το stack ενός από των 2 παικτών, ο έλεγχος πραγματοποιείται με την συνάρτηση ***game_not_over***. Κάθε επανάληψη του βρόγχου, συμβολίζει την παρτίδα ενός χεριού. Πριν από κάθε επανάληψη, μοιράζονται και εκτυπώνονται οι κάρτες στους παίκτες, με τις συναρτήσεις ***deal_cards*** και ***print_card***, αντίστοιχα. Οι κάρτες του board, εκτυπώνονται κανονικά και εμφανίζονται στους παίκτες, μόνο σε περίπτωση που μεταβούν στον γύρο 2, μέσω της ***print_board***. Ακόμη, διαλέγεται ο παίκτης που θα παίζει πρώτος σε αυτόν τον γύρο, με την σειρά να εναλλάσσεται σε κάθε χέρι. Στο τέλος κάθε επανάληψης, μέσω της συνάρτησης ***hand_end*** εκτυπώνονται το ολικό pot ,τα stacks των 2 παικτών και μια ένδειξη για το αν το χέρι έληξε επειδή ένας παίκτης έκανε fold ή αν οι παίκτες πήγαν σε showdown. Επίσης, οι κάρτες επιστρέφουν στο deck με την συνάρτηση ***restore_deck***.

Το βασικό σώμα του περιβάλλοντος είναι η υλοποίηση με εμφωλευμένες συνθήκες ελέγχου if, του παραπάνω διαγράμματος ροής ενεργειών των παικτών. Η περιήγηση στο διάγραμμα γίνεται μέσω της μεταβλητής ***round*** που υποδεικνύει αν οι παίκτες βρίσκονται στον γύρο 1 ή 2 και της συνάρτησης ***ask_action***. Η συνάρτηση αυτή, παίρνει σαν είσοδο , τα ***legal actions*** ,δηλαδή μια λίστα με τις επιτρεπτές ενέργειες που μπορεί να διαλέξει ο παίκτης, έναν δείκτη για το αν η ενέργεια που πρόκειται να επιλεχθεί είναι για εμάς ή για τον αντίπαλο και την δύναμη του χεριού/συνδυασμού του παίκτη. Σαν έξοδο επιστρέφει την αντίστοιχη ενέργεια που επιλέγει ο παίκτης ανάλογα με το μοντέλο στρατηγικής που χρησιμοποιεί ο ίδιος. Όσον αφορά την δύναμη του χεριού του παίκτη, υπολογίζεται με έναν πολύ απλό τρόπο από την συνάρτηση ***calculate_strength.*** Αν ο παίκτης βρίσκεται στον γύρο 1, τότε η δύναμη του χεριού του θα είναι μεταξύ 1 έως 5, ανάλογα την κάρτα στο χέρι του ***T=1,J=2,Q=3,K=4,A=5***. Όταν μεταβαίνει στον γύρο 2, η δύναμη θα κυμαίνεται μεταξύ 1 έως 15 ,ανάλογα τον συνδυασμό που έχει το χέρι του με το board. Συγκεκριμένα είναι η αρχική δύναμη του γύρου 1 ***,+ 0 για high card, +5 για one pair, +10 για triplet***. Σε περίπτωση που οι παίκτες φτάσουν σε showdown, ο νικητής υπολογίζεται από την συνάρτηση ***calculate_winner*** και θα είναι εκείνος που έχει το μεγαλύτερο strength ή επιστρέφει split σε περίπτωση ισοπαλίας. Τέλος, κάθε φορά που τελειώνει ένας γύρος (fold ή showdown), η συνάρτηση ***calculate_blinds***, ανανεώνει τα stacks των παιχτών αντίστοιχα με το πόσα blinds κέρδισαν ή έχασαν. Το ελάχιστο που μπορεί να χάσει/κερδίσει κάποιος παίκτης σε έναν γύρο είναι 0,5 blinds (τα υποχρεωτικά, ακολουθία:R1:bet-fold) και το μέγιστο 4,5 blinds (ακολουθία: R1:bet-raise-call, R2:bet-raise-call).

# 3 Μοντελοποίηση Αντίπαλων Πρακτόρων:

Για τον σωστό έλεγχο λειτουργίας και αποδοτικότητας των αλγορίθμων που υλοποιήθηκαν, χρειάστηκε να δημιουργηθούν αρκετοί αντίπαλοι agents που προσομοιάζουν διαφορετικές συμπεριφορές και στρατηγικές παικτών πόκερ. Συγκεκριμένα ένας αντίπαλος κρίνεται από το πόσο επιθετικά ή παθητικά παίζει.

Ένας επιθετικός παίκτης, παίζει περισσότερα χέρια, ανεξάρτητα της δύναμης τους, δηλαδή μεταβαίνει με υψηλή συχνότητα στον γύρο 2 (άρα κάνει bets και calls) και με χαμηλή συχνότητα κάνει fold στον γύρο 1. Ακόμη, κάνει πολλά πονταρίσματα και αυξήσεις ακόμα και αν δεν έχει πολύ δυνατό συνδυασμό , με σκοπό να μπλοφάρει και να κερδίσει το χέρι , με μικρότερη δύναμη από αυτή του αντιπάλου του. Η στρατηγική του είναι, να κερδίζει όσα περισσότερα χέρια μπορεί και συχνά να αναγκάζει τον αντίπαλό του να πηγαίνει fold ποντάροντας, ανεξαρτήτως της δύναμης που έχει ο ίδιος.

Αντίθετα, ένας παθητικός παίκτης, παίζει λιγότερα χέρια, μόνο εκείνα που έχουν υψηλή δύναμη, δηλαδή μεταβαίνει με χαμηλή συχνότητα στον γύρο 2 (άρα κάνει αρκετά folds), αλλά όταν μεταβαίνει σίγουρα θα έχει υψηλή δύναμη. Επίσης, κάνει πολλά checks και περισσότερα calls από bets, καθώς και παίζει συντηρητικά , μόνο όταν είναι σίγουρος ότι θα κερδίσει. Η στρατηγική του είναι να αφήνει τον αντίπαλο του να κερδίζει πολλά μικρά pots, αλλά στα λίγα που θα παίζει να νικάει με μεγάλη πιθανότητα ,έχοντας το ξεκάθαρο μαθηματικό πλεονέκτημα δύναμης χεριού.

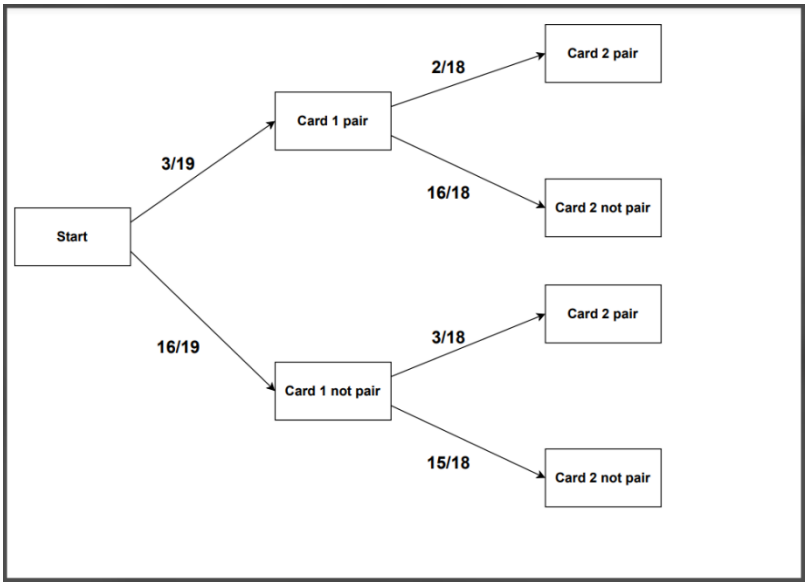
### Random (τυχαίος):
Σε κάθε απόφαση επιλέγει τελείως τυχαία μια ενέργεια από την λίστα των επιτρεπτών ενεργειών της κάθε περίστασης.

### Threshold_loose (επιθετικός):
Στον 1ο γύρο, κάνει bet/raise με τα χέρια Q,K,A ενώ κάνει check/call με τα T,J. Πάντα θα πηγαίνει στον 2ο γύρο. Στον 2ο γύρο, με δύναμη από 4 και πάνω , δηλαδή με K,A high card, οποιοδήποτε one pair και οποιαδήποτε triplet, κάνει bet/raise. Ενώ με δύναμη κάτω από 4, δηλαδή T,J,Q high card, κάνει επιθετικό ποντάρισμα/μπλόφα bet/raise με πιθανότητα 50%, και τις υπόλοιπές φορές κάνει check/fold.

### Threshold_tight (παθητικός):
Στον 1ο γύρο, με τα χέρια K,A κάνει bet/raise, με την Q κάνει bet/call , ενώ με T,J κάνει check/fold. Θα μπαίνει στον 2ο γύρο σίγουρα με τα Q,K,A ενώ με T,J θα μεταβαίνει στον 2ο γύρο μόνο αν η ενέργεια του αντιπάλου είναι check. Στον 2ο γύρο, κάνει bet/raise με δύναμη πάνω από 5, δηλαδή αν έχει οποιοδήποτε one pair και οποιαδήποτε triplet. Με δύναμη ίση με 5, δηλαδή A high card, κάνει check/call και σε οποιαδήποτε άλλη περίπτωση με δύναμη κάτω από 5 κάνει check/fold.

# 4 Μοντελοποίηση Πόκερ με Markov Decision Process

Ξεκινώντας, πριν να υλοποιηθούν οι αλγόριθμοι ενισχυτικής μάθησης, είναι αναγκαίο να αναπαρασταθεί το παιχνίδι σαν Markov Decision Process. Ορίστηκαν οι ενέργειες, οι καταστάσεις , τα έπαθλα και οι πιθανότητες μετάβασης. Οι συνολικές καταστάσεις χωρίστηκαν ανάλογα με την δύναμη του χεριού του παίκτη και ανάλογα τον γύρο που βρίσκεται. Υπάρχουν 5 καταστάσεις για γύρο 1, 15 καταστάσεις για γύρο 2 και 1 παραπανίσια τερματική κατάσταση. Για τις καταστάσεις του γύρου 1, χρειάστηκε να υπολογιστούν με μαθηματικό τρόπο και οι πιθανότητες μετάβασης στις καταστάσεις του γύρου 2. Παρακάτω δίνεται το δέντρο των πιθανοτήτων των μεταβάσεων:




Οι πιθανότητες μετάβασης από κάποια κατάσταση του γύρου 1, χωρίζονται σε 3 μέρη, των οποίων η πιθανότητα υπολογίζεται από το παραπάνω δέντρο. Στην πιθανότητα ο παίκτης να φτιάξει συνδυασμό one pair : $\frac{3}{19} * \frac{16}{18} + \frac{16}{19} * \frac{3}{18} = 0{,}283$ , στο να φτιάξει συνδυασμό triplet : $\frac{3}{19} * \frac{2}{18} = 0{,}017$ και τέλος, στο να μην φτιάξει κανέναν συνδυασμό και να παραμείνει με high card : $1 - 0{,}283 - 0{,}017 = 0{,}7$ . Ενώ αν ο παίκτης βρίσκεται ήδη στον γύρο 2, η μόνη πιθανή μετάβαση είναι στην τερματική κατάσταση με πιθανότητα 1. Τα rewards, θα διαφέρουν ανάλογα τον αντίπαλο και θα αναλυθούν στην επόμενη παράγραφο.

**Actions:** Fold (0), Check (1), Call (2), Bet (3), Raise (4)
**States:**
*S0-S4*: preflop states (round 1): *S0*=T, *S1*=J, *S2*=Q, *S3*=K, *S4*=A
*S5-S19*: afterflop states (round 2): *S5-S9* = high card, *S10-S14*= one pair, *S15-S19* = triplet
*S20* : terminal state
**Transitions Probabilities:**
from *S0-S4*: to *S5-S9* →Pr = 0,668 , to *S10-S14* →Pr = 0,315 , to *S15-S19* →Pr = 0,017
from *S5-S19*: to *S20* →Pr = 1

# 5 Αλγόριθμος Policy Iteration:

Ο αλγόριθμος policy iteration, είναι ένας αλγόριθμος μηχανικής μάθησης που ακολουθεί 2 βήματα, το βήμα αξιολόγησης και το βήμα βελτιστοποίησης.

### 1.Βήμα αξιολόγησης:

Στο πρώτο βήμα, αξιολογείται η πολιτική του αλγορίθμου. Αυτό συμβαίνει υπολογίζοντας την τιμή κάθε κατάστασης (state) με βάση την τρέχουσα πολιτική. Αυτή η τιμή αντιπροσωπεύει το αναμενόμενο συνολικό κέρδος που μπορεί να προκύψει από αυτήν την κατάσταση, λαμβάνοντας υπόψη την τρέχουσα πολιτική και την πιθανότητα μετάβασης σε άλλες καταστάσεις.

### 2.Βήμα βελτιστοποίησης:

Στο δεύτερο βήμα, βελτιστοποιείται η πολιτική με βάση τις αξιολογήσεις που προκύπτουν από το προηγούμενο βήμα. Αυτό συμβαίνει επιλέγοντας για κάθε κατάσταση την ενέργεια (action) που μεγιστοποιεί την αναμενόμενη ανταμοιβή, όπως προκύπτει από τις αξιολογήσεις. Η βελτιστοποίηση της πολιτικής επαναλαμβάνεται μέχρις ότου η πολιτική συγκλίνει στη βέλτιστη πολιτική.

Παρακάτω φαίνονται οι 2 εξισώσεις που χρησιμοποιούνται σε κάθε βήμα:
**V** = συνάρτηση αξιολόγησης, **R** = έπαθλα, **Pr** = πιθανότητες μετάβασης
**a** = ενέργεια , **π** = πολιτική, **s** = κατάσταση, **s'** = επόμενη κατάσταση
**γ** = παράγοντας έκπτωσης

1. Policy evaluation

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} \Pr(s'|s, \pi(s))\, V^{\pi}(s') \;\; \forall s$$

2. Policy improvement

$$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}}\, R(s, a) + \gamma \sum_{s'} \Pr(s'|s, a)\, V^{\pi}(s') \;\; \forall s$$

Ο αλγόριθμος συνεχίζει να επαναλαμβάνει αυτά τα δύο βήματα αναδρομικά μέχρι η πολιτική να συγκλίνει στη βέλτιστη πολιτική. Συγκλίνει όταν η βελτιστοποίηση της πολιτικής δεν παράγει πλέον αλλαγές στις ενέργειες που επιλέγονται για κάθε κατάσταση. Το Policy iteration εγγυάται να συγκλίνει στη βέλτιστη πολιτική εφόσον ικανοποιούνται ορισμένες προϋποθέσεις, όπως η προβλεψιμότητα και η περιορισμένη επανάληψη των βημάτων. Για την επαλήθευση της σύγκλιση, χρησιμοποιείται ένα κριτήριο σύγκλισης, το οποίο συγκρίνει την παλιά με την νέα πολιτική σε διαδοχικά βήματα για να διαπιστωθεί εάν έχει συμβεί αλλαγή ή όχι.

Παρακάτω φαίνεται η υλοποίηση του αλγορίθμου σε ψευδοκώδικα :

modifiedPolicyIteration(MDP)
Initialize $\pi_0$ and $V_0$ to anything
$n \leftarrow 0$
Repeat
  Eval: Repeat $k$ times
    $V_n \leftarrow R^{\pi_n} + \gamma T^{\pi_n} V_n$
  Improve: $\pi_{n+1} \leftarrow argmax_a \; R^a + \gamma T^a V_n$
    $V_{n+1} \leftarrow max_a \; R^a + \gamma T^a V_n$
  $n \leftarrow n + 1$
Until $\left|\left|V_n - V_{n-1}\right|\right|_{\infty} \leq \epsilon$
Return $\pi_n$

Στην συνέχεια θα αναδειχθεί ο τρόπος με τον οποίο δημιουργήθηκαν οι πίνακες μεταβάσεων και επάθλων και οι τιμές των παραμέτρων. Στο σημείο αυτό, είναι πολύ σημαντικό να γίνει αντιληπτό πως ο policy iteration, λειτουργεί έχοντας γνώση του μοντέλου στρατηγικής του αντιπάλου. Αυτό επιτρέπει στον αλγόριθμο να αντιμετωπίζει διαφορετικά κάθε αντίπαλο και να βρίσκει την συγκεκριμένη βέλτιστη πολιτική που δίνει τα καλύτερα αποτελέσματα εναντίον του.

Η παράμετρος γ (discount factor) τέθηκε ίση με 0,95 διότι οι μελλοντικές/μακρινές τιμές των reward, έχουν την ίδια αξία με τις αρχικές. Για αυτό τον λόγο πρέπει να είναι πολύ κοντά στο 1, αλλά όχι ακριβώς 1 ,για να υπάρξει σίγουρα η σύγκλιση .Ενώ για τους 2 διαφορετικούς αντιπάλους threshold_loose και threshold_tight, δημιουργήθηκαν 2 διαφορετικοί πίνακες

P_vs_loose και P_vs_tight, οι οποίοι περιέχουν τις πιθανότητες μετάβασης όλων των καταστάσεων και τα αντίστοιχα rewards για κάθε μια ενέργεια που μπορεί ο παίκτης να πραγματοποιήσει από την κάθε κατάσταση. Τα rewards της κάθε ενέργειας (από κάθε κατάσταση), είναι πολύ διαφορετικά στους 2 πίνακες, γιατί οι στρατηγικές που χρησιμοποιήθηκαν για να νικηθεί ο αντίστοιχος αντίπαλος, είναι πολύ διαφορετικές.

Συγκεκριμένα, ενάντιων του loose (και του random) πράκτορα, επιβραβεύονται θετικά οι παθητικές ενέργειες και αρνητικά οι επιθετικές. Δηλαδή ο πράκτορας, θα παίζει επιθετικά μόνο τα χεριά και τους συνδυασμούς με μεγάλη δύναμη, με σκοπό όταν ο αντίπαλος του ποντάρει να είναι σίγουρος ότι θα νικήσει, ενώ τα πιο αδύναμα χέρια θα τα παίζει παθητικά. Πρακτικά, υλοποιεί μια υπομονετική συμπεριφορά που μπορεί να χάνει 0,5 blinds πολλές φορές επειδή δεν έπαιξε το αρχικό χέρι, αλλά όταν παίζει το χέρι θα έχει πάντα το μαθηματικό πλεονέκτημα να κερδίσει όλα τα μεγάλα (με πολλά blinds στο κέντρο) pots.

Αντίθετα, ενάντιων του tight πράκτορα, επιβραβεύονται θετικά οι επιθετικές ενέργειες και αρνητικά οι παθητικές. Δηλαδή ο πράκτορας, θα παίζει επιθετικά όλα τα χέρια ανεξαιρέτως της δύναμης τους και θα κάνει πάντα πονταρίσματα. Πρακτικά υλοποιεί μια πολύ επιθετική συμπεριφορά που έχει σκοπό να νικάει όσα πιο πολλά pots γίνεται , χωρίς να έχει σημασία το χέρι που του δόθηκε. Βασίζεται στο να αναγκάζει τον αντίπαλο με μπλόφες , να πηγαίνει πολλά folds, ακόμα και αν έχει μεγαλύτερη δύναμη (δηλαδή καλύτερο χέρι) από το δικό του.

Ο αλγόριθμος , φαίνεται πως κάνει converge στην optimal λύση(δηλαδή δεν βρίσκει κάποια καλύτερη πολιτική) πάντα μετά από 3 επαναλήψεις (σπάνια μετά από 2) και παίρνει τις αναμενόμενες σωστές αποφάσεις για την στρατηγική που υλοποιείται κάθε φορά. Γεγονός που ήταν αναμενόμενο να κάνει converge σε τόσες λίγες επαναλήψεις, διότι ο χώρος των καταστάσεων , είναι μικρός ,αποτελείται μόνο από 21 καταστάσεις.

# 6 Αλγόριθμος Q-Learning:

Ο Q Learning είναι ένας αλγόριθμος ο οποίος μαθαίνει την βέλτιστη πολιτική για την επιλογή της επόμενης ενέργειας σε ένα δοσμένο περιβάλλον. Σε αντίθεση με τον Policy Iteration ο Q Learning λειτουργεί χωρίς γνώση για την στρατηγική του αντιπάλου. Αυτό σημαίνει ότι οι κινήσεις που διαλέγει σαν βέλτιστες προσαρμόζονται σύμφωνα με τον αντίπαλο που έχει κάθε φορά.

Για τον υπολογισμό της βέλτιστης αυτής πολιτικής χρησιμοποιείται ένας πίνακας Q. Το μέγεθος του πίνακα αυτού εξαρτάται από το state space του περιβάλλοντος και τα διαθέσιμα actions που δίνονται. Η χρησιμότητα του πίνακα αυτού είναι για να αποθηκεύει τις αναμενόμενες ανταμοιβές για κάθε ζεύγος κατάστασης-ενέργειας. Ο πίνακας αυτός ενημερώνεται επαναληπτικά, και πιο συγκεκριμένα κάθε φορά που ο πράκτορας μας λαμβάνει κάποια ενέργεια. Με αυτόν τον τρόπο θα μάθει σταδιακά την βέλτιστη πολιτική για κάθε δυνατή κατάσταση που μπορεί να βρεθεί μέσα στο περιβάλλον.

Το περιβάλλον το οποίο χρησιμοποιήθηκε έχει περιγράφει πλήρως σε παραπάνω ενότητα, και δεν θα αναλυθεί περαιτέρω. Οι δυνατές ενέργειες οι οποίες έχουν δοθεί σύμφωνα με το περιβάλλον είναι οι check, fold, raise, call, και bet. Για την αναπαράσταση του παιχνιδιού και του state-space, actions χρησιμοποιήθηκαν οι αναπαραστάσεις που προαναφέρθηκαν,

Όταν δημιουργείται το object του QLearning πράκτορα αρχικοποιείται και ο πίνακας q βάζοντας 0 σε κάθε ζεύγος κατάστασης-ενέργειας. Εν συνεχεία, θα πρέπει να γίνει η

εκπαίδευση του πράκτορα παίζοντας πολλούς γύρους με έναν συγκεκριμένο αντίπαλο. Σε κάθε γύρο, ο πράκτορας επιλέγει μια ενέργεια χρησιμοποιώντας την πολιτική του ε-greedy (ο οποίος έχει αναλυθεί σε προηγούμενη εργασία ). Με το πέρασμα του χρόνου η τιμή του ε θα μειώνεται σταδιακά, έτσι ώστε να εκμεταλλευτούμε την γνώση που αποκτά ο πράκτορας. Αρχικά, πρέπει να εξερευνά περισσότερο για να ανακαλύψει τις βέλτιστες στρατηγικές . Στην συνέχεια όσο αποκτάει γνώση θα πρέπει να τις εκμεταλλεύεται, επιλέγοντας ενέργειες βασισμένες στις τιμές Q. Μετά την εκτέλεση κάθε ενέργειας θα πρέπει να λάβει την κατάλληλη ανταμοιβή και να ανανεωθεί ο πίνακας q με βάση την παρακάτω εξίσωση:

Παρακάτω φαίνεται η υλοποίηση του αλγορίθμου σε ψευδοκώδικα :

**Q** = συνάρτηση/πίνακας αξιολογισης, **r** = έπαθλα
**a** = ενέργεια , **π** = πολιτική, **s** = κατάσταση, **s'** = επόμενη κατάσταση
**γ** = παράγοντας έκπτωση, **α** = ρυθμός μάθησης
**max(Q(s', a'))** είναι η μέγιστη τιμή Q για την επόμενη κατάσταση s' και όλες τις πιθανές ενέργειες

Qlearning$(s, Q^*)$
Repeat
Select and execute $a$
Observe $s'$ and $r$
Update counts: $n(s,a) \leftarrow n(s,a) + 1$
Learning rate: $\alpha \leftarrow 1/n(s,a)$
Update Q-value:
$$Q^*(s,a) \leftarrow Q^*(s,a) + \alpha\left(r + \gamma \max_{a'} Q^*(s',a') - Q^*(s,a)\right)$$
$s \leftarrow s'$
Until convergence of $Q^*$
Return $Q^*$

Για τον υπολογισμό του ε σε κάθε γύρο χρησιμοποιήθηκε η εξής συνάρτηση:

$$\boldsymbol{\varepsilon = (episode + 1)^{-\frac{1}{4}}}$$

Είναι αναγκαίο να αναφερθεί ότι όταν τελειώσει η εκπαίδευση του πράκτορα θα οριστεί η τιμή του ε σε 0, έτσι ώστε να επιλέγονται οι ενέργειες που ο πράκτορας θεωρεί βέλτιστες.

# 7 Πειραματικές Μετρήσεις , Αποτελέσματα & Συμπεράσματα:

Παρακάτω φαίνονται τα πειράματα που έτρεξαν με τον αλγόριθμο Policy iteration, εναντίον των 3ιων πρακτόρων random,threshold_loose και threshold_tight. Κρατήθηκαν μετρήσεις από 10 παιχνίδια με αρχικό stack των παικτών στα 20 Blinds, έτσι ώστε οι παρτίδες να μην κρίνονται τόσο από τον παράγοντα της τύχης του πόκερ, αλλά οι πράκτορες να προλαβαίνουν να εφαρμόσουν τις στρατηγικές τους σε πολλαπλά χέρια. Οι μετρήσεις που καταγράφηκαν είναι αν ο παίκτης μας νικάει/χάνει και τον μέσο αριθμό blinds/hand που κερδίζει και το πλήθος χεριών που παίχτηκαν μέχρι να τερματίσει το παιχνίδι.

| Policy Iteration Algorithm<br>(10 games, 20 blinds starting stack) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **vs Random** | | | **vs Threshold_loose** | | | **vs Threshold_tight** | | |
| **Win/ Loose** | **#Hands** | **Average Blidns/Hand ratio** | **Win/ Loose** | **#Hands** | **Average Blidns/Hand ratio** | **Win/ Loose** | **#Hands** | **Average Blidns/Hand ratio** |
| W | 24 | 0,80 | W | 98 | 0,20 | W | 48 | 0,40 |
| W | 52 | 0,60 | W | 99 | 0,20 | W | 27 | 0,70 |
| W | 72 | 0,27 | W | 8 | 2,20 | W | 28 | 0,68 |
| W | 23 | 0,83 | W | 119 | 0,16 | W | 38 | 0,51 |
| W | 56 | 0,35 | W | 104 | 0,19 | W | 54 | 0,35 |
| W | 77 | 0,25 | W | 23 | 0,83 | W | 36 | 0,54 |
| W | 33 | 0,59 | W | 73 | 0,27 | W | 32 | 0,60 |
| W | 15 | 1,25 | W | 88 | 0,22 | W | 31 | 0,62 |
| W | 16 | 1,17 | W | 44 | 0,44 | W | 17 | 1,11 |
| W | 56 | 0,35 | W | 77 | 0,25 | W | 19 | 1 |
| **Win ratio:** **100%** | **Average #Hands:** **42,4** | **Avg.Avg Blinds/Hand ratio:** **0,64** | **Win ratio:** **100%** | **Average #Hands:** **73,3** | **Avg.Avg Blinds/Hand ratio:** **0,49** | **Win ratio:** **100%** | **Average #Hands:** **33** | **Avg.Avg Blinds/Hand ratio:** **0,65** |

Παρατηρείται από τα αποτελέσματα καταρχάς πως νικάει πάντα όλους τους πράκτορες σε δείγμα 10 παιχνιδιών. Αυτό θα ισχύει πάντα εναντίον του πράκτορα random, αλλά εναντίον των άλλων 2 πρακτόρων έχει παρατηρηθεί ότι για μεγαλύτερο πλήθος δοκιμών ο παίκτης μας ,μπορεί να χάσει με μια πολύ μικρή πιθανότητα (1/30 παιχνίδια).Γεγονός που είναι λογικό, καθώς και το πόκερ περιέχει τυχαιότητα και δεν είναι ντετερμινιστικό παιχνίδι.

Συνεχίζοντας, παρατηρείται πως εναντίον του random, δεν παρατηρούνται μοτίβα συμπεριφορών ,που είναι λογικό. Ενώ ,μεταξύ των άλλων 2, υπάρχουν ξεκάθαρα μοτίβα που σχετίζονται με τις στρατηγικές των παιχτών.

Παίζοντας εναντίον του loose, ο αριθμός χεριών που χρειάζονται οι παρτίδες για να τερματίσουν είναι αυξημένος. Αυτό αιτιολογείται από το γεγονός ότι ο πράκτοράς μας, αφήνει τον loose να κερδίζει πολλά χέρια με pot 0,5 , διότι κάνει πολλά folds στον γύρο 1, μέχρι να του έρθει κάποιο καλό χέρι. Επίσης, παρατηρούνται με μικρή συχνότητα, φορές που ο αντίπαλος χάνει εξαιρετικά γρήγορα. Αυτό ερμηνεύεται σε ακολουθίες ενεργειών που σε

κάποιο χέρι πολύ νωρίς, ο αντίπαλος έπαιξε πολύ επιθετικά, όταν ο πράκτοράς μας είχε κάποιον πολύ δυνατό συνδυασμό και έτσι κατέληξε να χάνει κάποιο πολύ μεγάλο pot.

Αντίθετα, εναντίον στον tight,παρατηρείται πως τα χέρια που παίζονται ανα παρτίδα, έχουν έναν πολύ σταθερό αριθμό και δεν υπάρχουν μεγάλες αυξομειώσεις. Γεγονός που είναι αναμενόμενο, διότι εφόσον ο tight, παίζει πολύ παθητικά, ο πράκτοράς μας, νικάει κάθε pot συνεχόμενα. Εκτός από εκείνα που με μικρή πιθανότητα έχει κάποιον δυνατό συνδυασμό ο tight, στα οποία θα χρειαστεί λίγη παραπάνω ώρα μέχρι να τον κερδίσει.

Παρακάτω φαίνονται τα αντίστοιχα πειράματα με τον αλγόριθμο Q-Learning αυτή τη φορά:

| **Q-Learning Algorithm**<br>**(10 games, 20 blinds starting stack)** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **vs Random** | | | **vs Threshold_loose** | | | **vs Threshold_tight** | | |
| **Win/ Loose** | **#Hands** | **Average Blidns/Hand ratio** | **Win/ Loose** | **#Hands** | **Average Blidns/Hand ratio** | **Win/ Loose** | **#Hands** | **Average Blidns/Hand ratio** |
| W | 58 | 0,33 | L | 38 | -0,52 | W | 54 | 0,36 |
| W | 296 | 0,06 | L | 44 | -0,43 | W | 217 | 0,09 |
| W | 91 | 0,21 | L | 33 | -0,58 | W | 45 | 0,44 |
| W | 65 | 0,31 | L | 77 | -0,25 | W | 126 | 0,16 |
| W | 111 | 0,18 | L | 19 | -1 | W | 35 | 0,56 |
| W | 48 | 0,40 | L | 33 | -0.58 | W | 55 | 0,36 |
| W | 70 | 0,27 | L | 45 | -0,43 | W | 207 | 0,10 |
| W | 80 | 0,25 | L | 63 | -0,31 | W | 121 | 0,17 |
| W | 57 | 0,35 | L | 23 | -0,83 | L | 58 | -0,33 |
| W | 49 | 0,4 | L | 31 | -0,62 | W | 57 | 0,34 |
| **Win ratio:**<br>**100%** | **Average #Hands:**<br>**82,5** | **Avg.Avg Blinds/Hand ratio:**<br>**0,27** | **Win ratio:**<br>**0%** | **Average #Hands:**<br>**40,6** | **Avg.Avg Blinds/Hand ratio:**<br>**negative0,5** | **Win ratio:**<br>**90%** | **Average #Hands:**<br>**97,5** | **Avg.Avg Blinds/Hand ratio:**<br>**0,22** |

Από τα αποτελέσματα παρατηρήθηκε πως ενώ ο αλγόριθμος παίρνει τις σωστές αποφάσεις και ενημερώνει το Q-table σωστά, εναντίον του random και του tight, για κάποιο λόγο χάνει από τον loose. Γεγονός που μας οδηγεί στην πιθανότητα ύπαρξης κάποιου προγραμματιστικού λάθους στον κώδικα, το οποίο δεν μπόρεσε να γίνει αντιληπτό.

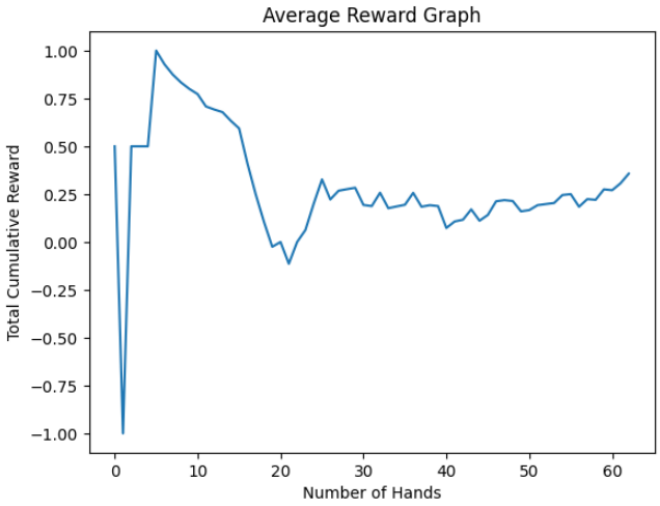
Η σύγκριση των 2 αλγορίθμων , δεν είναι εφικτή καθώς και δεν θα απεικονίζεται στην πραγματικότητα, λόγω της μερικής δυσλειτουργίας του Q-Learning. Παρόλα αυτά, όσον αφορά τον αλγόριθμο policy iteration, γνωρίζοντας την στρατηγική του αντιπάλου κάθε φορά, έχει εξαιρετικά αποτελέσματα.

# 8 Γραφικές Παραστάσεις:

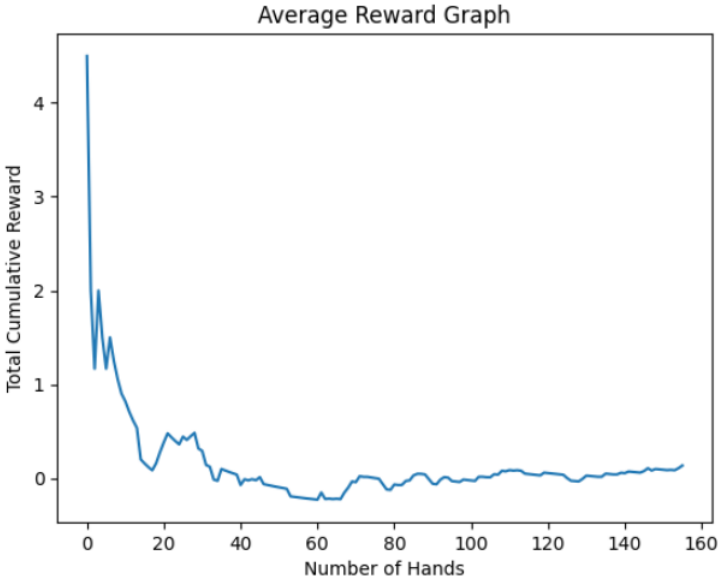
Παρακάτω αποδίδονται οι γραφικές παραστάσεις του average reward των 2 αλγορίθμων. Το average reward, είναι το αθροιστικό ποσό blinds που κερδίζει ή χάνει ο παίκτης σε κάθε χέρι δια το πλήθος χεριών που έχει παίζει.

## 1.Policy Iteration Average Reward:

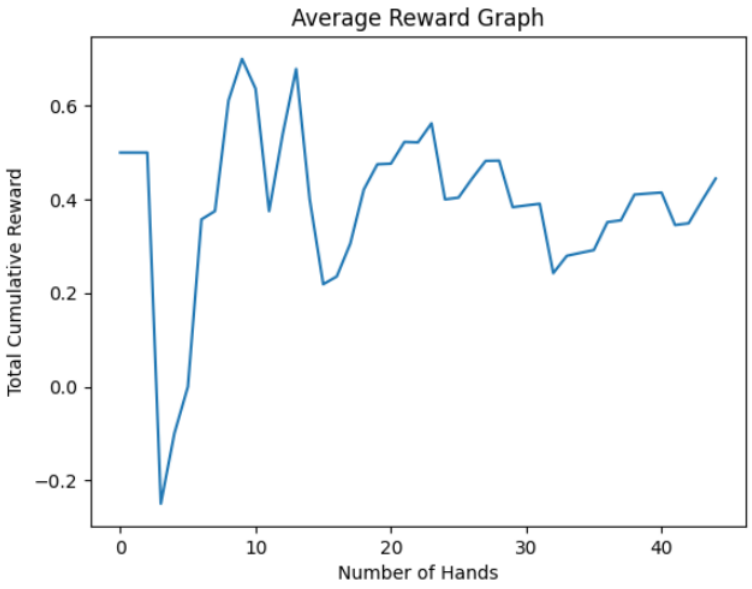
### vs Random:




### vs Threshold_Loose:

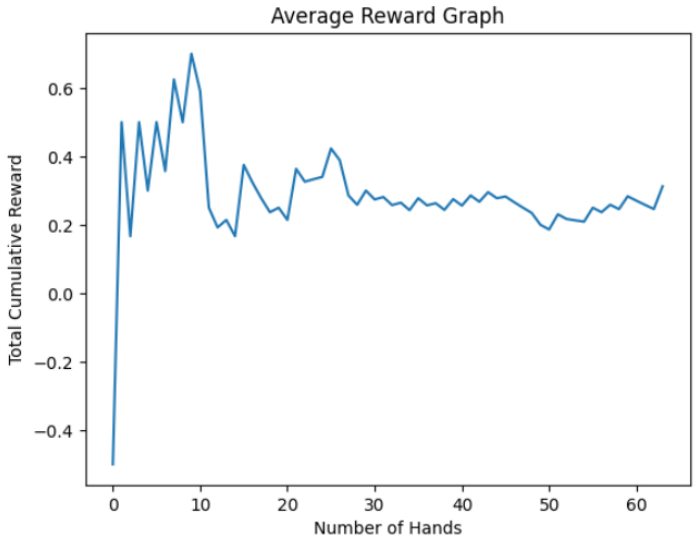



### vs Threshold_Tight:

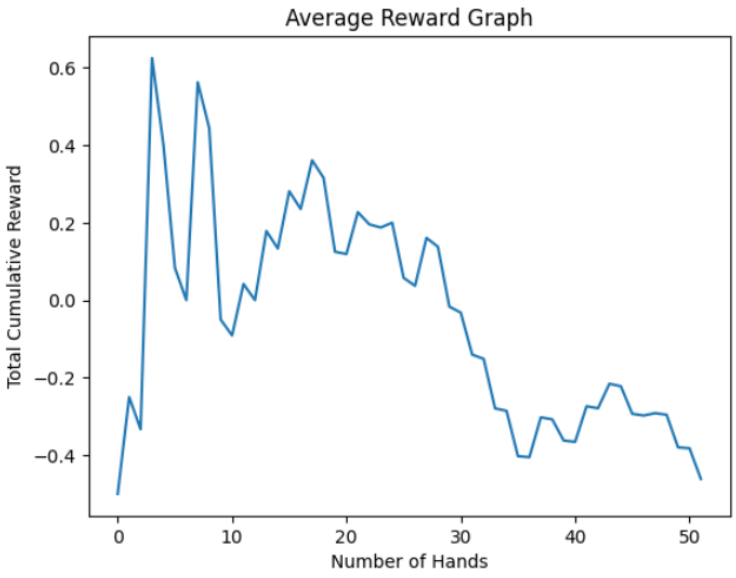


# 2.Q-Learning Average Reward:

## vs Random:




## vs Threshold_Loose:




## vs Threshold_Tight: