ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΓΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ PROLOG ΠΟΥ ΘΑ ΥΛΟΠΟΙΕΙ ΕΝΑ ΠΑΙΧΝΙΔΙ ΚΑΙ ΘΑ ΕΝΣΩΜΑΤΩΝΕΙ ΣΤΟΙΧΕΙΑ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

Του φοιτητή

Κωτσαλίδη Θεόδωρου

Αρ. Μητρώου: 03/2389

Επιβλέπων καθηγητής

Βοζαλής Εμμανουήλ

Θεσσαλονίκη 2013

ΠΡΟΛΟΓΟΣ

Στα πλαίσια αυτής της πτυχιακής εργασίας γίνεται μια προσπάθεια ανάδειξης της συμπεριφοράς των προγραμμάτων τεχνητής νοημοσύνης στο πόκερ, αναλύοντας τα βασικότερα χαρακτηριστικά και τις λειτουργίες τους. Αρχικά γίνεται μια εκτενής θεωρητική ανάλυση όσον αφορά το χώρο του πόκερ και μετέπειτα μελετούνται τα κύρια στοιχεία που οφείλει να έχει ένας παίκτης πόκερ τεχνητής νοημοσύνης. Τελικά, εξάγονται χρήσιμα συμπεράσματα για τα πλεονεκτήματα αλλά και για τις αδυναμίες του παίκτη τεχνητής νοημοσύνης που δημιουργήθηκε.

Κωτσαλίδης Θεόδωρος

ΠΕΡΙΛΗΨΗ

Το πεδίο των παιχνιδιών ελλιπούς πληροφορίας, στο οποίο ανήκει το πόκερ, έχει απασχολήσει τους ερευνητές για πολλά χρόνια, ωστόσο ο τομέας μόλις τα τελευταία χρόνια κατάφερε να παράσχει καλούς ανταγωνιστικούς παίκτες τεχνητής νοημοσύνης (ΤΝ) για να παίξει μερικά από τα πιο περίπλοκα παιχνίδια καρτών σε ύψιστο επίπεδο. Σε αυτήν την εργασία εξετάζεται το παιχνίδι του πόκερ, με την δημιουργία του RezAI, ενός παίκτη ΤΝ που αναπτύχθηκε σε Prolog, του οποίου η συμπεριφορά βασίζεται σε στρατηγικές παιξίματος, στην επιλογή των οποίων υπάρχει μια πτυχή τυχαιότητας.

Για τον παίκτη αυτόν δημιουργήθηκαν συνδυασμοί τεσσάρων διαφορετικών στρατηγικών (Loose, Tight, Aggressive, Passive) οι οποίοι επιλέγονται από τον παίκτη ΤΝ, ανάλογα με την κατάσταση που βρίσκεται το παιχνίδι, προσομοιώνοντας τις βασικές συμπεριφορές παικτών που συναντώνται σε ένα παιχνίδι πόκερ μεταξύ ανθρώπων. Οι στρατηγικές αυτές δοκιμάστηκαν ενάντια σε ανθρώπινους παίκτες, φτάνοντας στο συμπέρασμα πως ο συνδυασμός των Loose και Aggressive έχουν την μεγαλύτερη αποτελεσματικότητα εναντίον ανθρώπινων αντιπάλων μεσαίου επιπέδου.

Επίσης, δημιουργήθηκε ένα γραφικό περιβάλλον σε Java, το οποίο συνδέεται με τον κώδικα της Prolog, καθιστώντας έτσι πιο εύκολη την χρήση της εφαρμογής.

Τέλος, εξάγονται ενδιαφέροντα συμπεράσματα σχετικά με τις στρατηγικές πόκερ και τους ανθρώπινους ευριστικούς κανόνες που παρουσιάζονται σε αυτή την εργασία.

ABSTRACT

The field of Imperfect Information Games, which includes poker, has interested researchers for many years, yet the sector has only recently managed to provide good competitive artificial intelligence (AI) players to play some of the most complex card games at master level. In this thesis we observe the game of poker, along with the creation of RezAI, a strategy-based AI player written in Prolog language, demonstrating an aspect of randomness in his choices.

The AI player can choose between combinations of four different strategies which were created for him (Loose, Tight, Aggressive, Passive), depending on the game status, simulating the key behaviors of human players found in a poker game. These strategies were tested against human players, to reach the conclusion that the combination of Loose-Aggressive has the greatest efficiency against mid-level human opponents.

To facilitate the use of the Prolog application, a GUI was created in Java, that is linked with the Prolog project.

Finally, some interesting conclusions are made about the poker strategies and human heuristics presented in this thesis.

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον καθηγητή **κ. Βοζαλή Εμμανουήλ** για την υπομονή του, την κατανόηση που έδειξε και την πολύτιμη βοήθεια που μου παρείχε κατά την εκπόνηση αυτής της πτυχιακής εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο σχημάτων

Ευρετήριο πινάκων

ΕΙΣΑΓΩΓΗ

Η εργασία αυτή έχει ως στόχο την δημιουργία ενός προγράμματος σε Swi-Prolog, στο οποίο ένας παίκτης ΤΝ θα μπορεί να επιδείξει βασικά στοιχεία ΤΝ, στο εξειδικευμένο παιχνίδι πόκερ Limit Heads-Up Texas Hold'em (παιχνίδι 2 ατόμων με όριο πονταρίσματος). Για την επιτυχή ανάπτυξη του παίκτη ΤΝ πρέπει να ικανοποιούνται δύο επιμέρους στόχοι:

- ✓ **Αξιολόγηση της παρτίδας:** Καταρχήν ο παίκτης ΤΝ πρέπει να μπορεί να αξιολογήσει την αξία του φύλλου σε όλους τους γύρους της παρτίδας. Η αξιολόγηση των φύλλων πρέπει να λαμβάνει υπόψιν όλα τα στοιχεία που εμφανίζονται σε ένα παιχνίδι, όπως η δύναμη του φύλλου, οι δυνατότητες για βελτίωση του, το μέγεθος των πονταρισμάτων του κάθε γύρου, τον γύρο που βρίσκεται η παρτίδα κ.ά. Με βάση αυτά πρέπει να μπορεί να διαμορφώνει μία καλή στρατηγική παιχνιδιού.

- ✓ **Μη-προβλεψιμότητα:** Το απρόβλεπτο δίνει πλεονέκτημα σε έναν παίκτη πόκερ. Ένας παίκτης ΤΝ που παίζει με βάση μία σταθερή στρατηγική είναι αδύναμος απέναντι σε έναν παίκτη ο οποίος μπορεί να μοντελοποιεί τον αντίπαλο. Μία σταθερή πολιτική θα έχει πάντα κάποιες αδυναμίες που μπορούν να ανακαλυφθούν από αντιπάλους. Αντίθετα ένας παίκτης που παίζει, σε έναν βαθμό, απρόβλεπτα μπορεί να παραπλανήσει τον αντίπαλο του και να αυξήσει το κέρδος του.

Επίσης είναι αναγκαία η δημιουργία ενός γραφικού περιβάλλοντος για την εφαρμογή, το οποίο επιλέχθηκε να αναπτυχθεί σε Java. Η σύνδεση των δύο γλωσσών θα γίνεται μέσω της βιβλιοθήκης JPL, η οποία είναι πλέον ενσωματωμένη στην τελευταία έκδοση της SWI-PROLOG.

Το κεφάλαιο 1, παρουσιάζει το **Θεωρητικό Υπόβαθρο** του θέματος των παιχνιδιών ελλιπούς πληροφορίας, καθώς και τις προηγούμενες λύσεις που χρησιμοποίησαν οι παίκτες ΤΝ στο πρόβλημα του πόκερ. Παρέχει επίσης πληροφορίες σχετικά με τους τρεις πιο καλά τεκμηριωμένους παίκτες ΤΝ στον τομέα, δηλαδή τον Loki, Poki και PsOpti.

Κεφάλαιο 2, το **Παιχνίδι του Πόκερ**, ασχολείται με ένα εξειδικευμένο παιχνίδι πόκερ, το Limit Heads-Up Texas Hold'em. Το παιχνίδι, οι κανόνες ροής του παιχνιδιού, οι συνθήκες νίκες και ήττας, παρουσιάζονται όλα, μαζί με τις βασικές, αλλά και πιο σύνθετες στρατηγικές που χρησιμοποιούνται από τους ανθρώπινους παίκτες κατά την διάρκεια του παιχνιδιού.

Κεφάλαιο 3, **Σχεδιασμός και Μεθοδολογία**, παρουσιάζει το σχεδιασμό και τη δομή του συστήματος του παίκτη ΤΝ που δημιουργήθηκε. Η αρχιτεκτονική του συστήματος απεικονίζεται μαζί με τις στρατηγικές και τις προδιαγραφές τους.

Η τεκμηρίωση των δοκιμών, τα αποτελέσματα και η αξιολόγηση του παίκτη ΤΝ παρουσιάζονται στο κεφάλαιο 4, **Πειράματα και Αποτελέσματα**.

ΚΕΦΑΛΑΙΟ 1 - ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια το πόκερ έχει γίνει ένα από τα πιο δημοφιλή παιχνίδια στην παγκόσμια κοινότητα των παιχνιδιών, με πολλές εφαρμογές κάθε τύπου. Παρά το υψηλό ενδιαφέρον για το παιχνίδι αυτό, μόλις τα τελευταία πέντε χρόνια, τα υπολογιστικά προγράμματα και η υπάρχουσα τεχνητή νοημοσύνη (ΤΝ) έφτασε σε ένα ικανοποιητικό επίπεδο. Μετά από δεκαετίες έρευνας και εξελίξεων, τα καλύτερα προγράμματα πόκερ είναι πλέον ικανά να αντιμετωπίσουν επιτυχώς έναν παίκτη ύψιστου επιπέδου.

Τα παιχνίδια τα οποία ενδιαφέρουν την ΤΝ κατατάσσονται ανάλογα με το αν είναι διαθέσιμη η κατάσταση του παιχνιδιού στους παίκτες. Αν ολόκληρη η κατάσταση του παιχνιδιού είναι διαθέσιμη σε όλους τους παίκτες τότε είναι παιχνίδι πλήρους πληροφορίας (perfect information). Για παράδειγμα, το σκάκι και το τάβλι είναι παιχνίδια πλήρους πληροφορίας, αφού μπορεί ο κάθε παίκτης κοιτώντας την σκακιέρα να έχει όλες τις πληροφορίες για την κατάσταση του παιχνιδιού. Η πλήρης γνώση της κατάστασης του παιχνιδιού επιτρέπει σε brute-search αλγορίθμους να υπολογίζουν τα σενάρια και τις πιθανές μελλοντικές κινήσεις. Η επιτυχία του τομέα αυτού έχει επιτευχθεί μέσω της βελτίωσης της ταχύτητας των αναζητήσεων, για παράδειγμα, ο Deep Blue έλεγχε πάνω από 250 εκατομμύρια θέσεις το δευτερόλεπτο, για να μπορέσει να κάνει την βέλτιστη κίνηση σε μια παρτίδα σκάκι. Τα παιχνίδια πλήρους πληροφορίας έχουν μια καλά καθορισμένη βέλτιστη κίνηση, δηλαδή υπάρχει πάντα μια κίνηση που είναι τουλάχιστον τόσο καλή όσο μια άλλη κίνηση. Εκτός από αυτό, αν ο αντίπαλος μάθει την κίνηση αυτή, δεν θα υπάρξει καμία διαφορά στην τρέχουσα στρατηγική, καθώς είναι εξ’ ορισμού η βέλτιστη κίνηση. Αυτό είναι και το αντικείμενο έρευνας των αλγορίθμων αναζήτησης στο δέντρο ενός παιχνιδιού.

Από την άλλη μεριά, υπάρχουν τα παιχνίδια ελλιπούς πληροφορίας (imperfect information) όπως το πόκερ και το μπρίτζ, όπου ο κάθε παίκτης κρατάει κρυφά από τους άλλους παίκτες τα φύλλα του. Η ανάλυση αυτού του είδους παιχνιδιών γίνεται στο παρακάτω υποκεφάλαιο.

Μια άλλη κατάταξη των παιχνιδιών γίνεται με βάση την ύπαρξη στοχαστικότητας, σε αιτιοκρατικά (deterministic), και μη-αιτιοκρατικά (non-deterministic) παιχνίδια. Στο σκάκι δεν υπάρχει καθόλου στοχαστικότητα, είναι δηλαδή ένα αιτιοκρατικό παιχνίδι. Αντίθετα, το τάβλι είναι μη-αιτιοκρατικό αφού το ρίξιμο του ζαριού εισάγει τυχαιότητα στο παιχνίδι. Αν έπρεπε να κατατάξουμε το πόκερ σε μια από αυτές τις κατηγορίες τότε θα το βάζαμε στα μη-αιτιοκρατικά παιχνίδια, καθώς η ενέργεια του κάθε παίκτη εξαρτάται από ένα μεγάλο σύνολο παραγόντων, όπως για παράδειγμα η επιθυμία να εξαπατήσει τον αντίπαλο (μπλόφα), όπου στην ουσία παραβλέπει τα φύλλα τα οποία έχει.

1.1 ΠΑΙΧΝΙΔΙΑ ΕΛΛΙΠΟΥΣ ΠΛΗΡΟΦΟΡΙΑΣ

Αν και η αναζήτηση του δέντρου ενός παιχνιδιού λειτουργεί καλά στα παιχνίδια πλήρους πληροφορίας, υπάρχουν προβλήματα στην προσπάθεια να χρησιμοποιηθεί για παιχνίδια ελλιπούς πληροφορίας όπως το πόκερ. Η έλλειψη γνώσης σχετικά με τις πιθανές κινήσεις των αντιπάλων μεγαλώνει υπερβολικά την διακλάδωση του δέντρου, κάνοντας την αναζήτηση σε αυτό ανέφικτη.

Τα παιχνίδια ελλιπούς πληροφορίας παίζονται με μια συνεχή έλλειψη γνώσης μεταξύ των παικτών έχοντας μερική γνώση της κατάστασης του παιχνιδιού. Σε αντίθεση με τα παιχνίδια πλήρους γνώσης, μια αιτιοκρατική στρατηγική (deterministic strategy), δεν μπορεί να χρησιμοποιηθεί σ' αυτόν τον τύπο παιχνιδιών, καθώς θα επέτρεπε στον αντίπαλο να έχει το πλεονέκτημα να μαντέψει την πλήρη γνώση του παιχνιδιού. Ο λόγος για τον οποίο τα παιχνίδια ελλιπούς πληροφορίας θεωρούνται δύσκολα είναι επειδή απαιτεί από τους παίκτες, ανθρώπινους και TN, να αντιμετωπίσουν την αβεβαιότητα, να ρισκάρουν και να αναπτύξουν στρατηγικές. Επίσης, εκτός από τις καλές στρατηγικές και τον καλό τρόπο παιξίματος, είναι αναγκαία η ενσωμάτωση μιας πτυχής τυχαιότητας στην στρατηγική του παίκτη TN. Για τον λόγο αυτόν, οι βέλτιστες στρατηγικές στα παιχνίδια ελλιπούς πληροφορίας μπορούν να περιγραφούν ως ένας τυχαίος συνδυασμός στρατηγικών, έχοντας κάνει την βέλτιστη αξιολόγηση, για μια κατάσταση παιχνιδιού όπου υπάρχει μερική γνώση.

Έχει γίνει μεγάλη έρευνα στο χώρο της ελλιπούς πληροφορίας που επέφερε θεωρητικές λύσεις ή παίκτες TN για συγκεκριμένα είδη πόκερ. Μέχρι το 2007 ο στατιστικά καλύτερος παίκτης TN, ο PsOpti του πανεπιστημίου της Αλμπέρτα, θεωρούνταν καλύτερος από τους περισσότερους ανθρώπινους παίκτες αλλά αδυνατούσε να αντιμετωπίσει επιτυχώς κορυφαίους παίκτες. Επίσης, ο PsOpti λειτουργεί μόνο με μια περιορισμένη μορφή του Texas Hold'em Poker, βελτιστοποιημένη για 2 παίκτες. Ωστόσο, μετά τη δημιουργία του Polaris το 2007, ήρθαν τα πρώτα θετικά αποτελέσματα ενάντια σε κορυφαίους ανθρώπινους παίκτες. Όπως είδαμε στο δεύτερο Man-Machine Poker Championship [16], που έγινε το 2008, το Polaris νίκησε έξι κορυφαίους παίκτες με 3 νίκες, 2 ήττες και μία ισοπαλία, έχοντας βέβαια αρκετό δρόμο ακόμα για να φτάσει στο επιθυμητό επίπεδο.

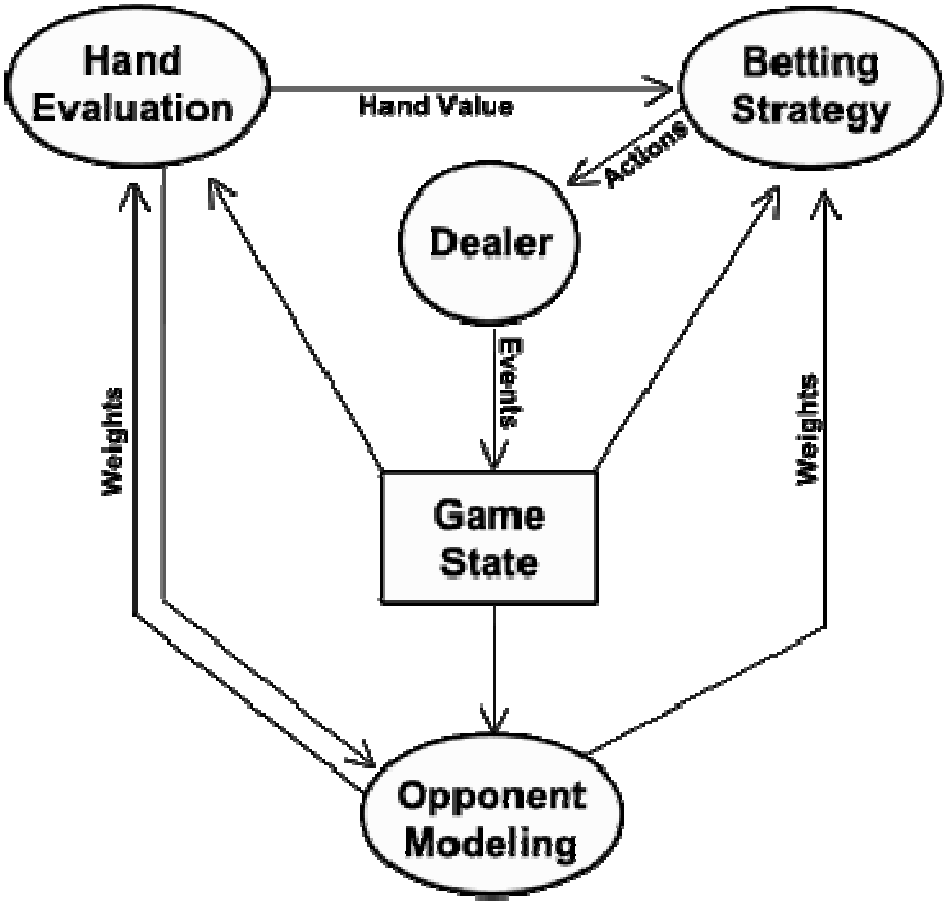
Παρακάτω παρουσιάζονται οι σύντομες περιγραφές των Loki, PsOpti και Poki, τριών δημοφιλών προγραμμάτων τεχνητής νοημοσύνης για το πόκερ που αναπτύχθηκαν από το πανεπιστήμιο της Αλμπέρτα.

1.2 LOKI

Ο Loki ήταν ένας από τους πρώτους ολοκληρωμένους παίκτες πόκερ τεχνητής νοημοσύνης. Αναπτύχθηκε το 1997 από τον Darse Billings και τεκμηριώνεται στο [2]. Ενσωματώνει μία προηγμένη βάση γνώσης και λειτουργικότητα που βασίζεται σε εμπειρικούς κανόνες του Billings, με τους οποίους σχηματίστηκε το δέντρο του παιχνιδιού. Το δέντρο αυτό αποτελείται από σενάρια και στρατηγικές τα οποία προτάθηκαν από παίκτες υψηλού επιπέδου και θα επιτρέπουν στη μηχανή του παιχνιδιού να βρίσκει μία σχεδόν βέλτιστη λύση, ή μία τυχαία λύση.

Η αρχική έκδοση του Loki στηρίχθηκε αποκλειστικά σε εμπειρικούς κανόνες για να παίξει ενάντια σε άλλους παίκτες, δημιουργώντας έτσι αμφιβολίες για το αν είναι πραγματικά ένας παίκτης TN, ή απλά μία μεταφορά σε κώδικα, της συμπεριφοράς ενός παίκτη υψηλού επιπέδου. Στις στρατηγικές δίνεται μία κατάταξη προτεραιότητας (priority ranking), και επιλέγονται με την χρήση μιας σήμανσης βαρύτητας (weights), επιτρέποντας τους να είναι σχετικά τυχαίες.

Η απόδοση του προγράμματος στην αρχική του έκδοση ήταν ικανοποιητική μόνο ενάντια σε αρχάριους παίκτες, καθώς οποιαδήποτε μορφή σταθερής στρατηγικής, μπορούσε να χρησιμοποιηθεί προς όφελος ενός πιο έμπειρου παίκτη. Στο Σχήμα 1 φαίνεται η βασική αρχιτεκτονική του Loki και οι λειτουργίες της [2].



Σχήμα 1 "Αρχιτεκτονική Loki"

1.3 PSOPTI

Το πρόγραμμα PsOpti στο οποίο βασίζεται ένας από τους κορυφαίους παίκτες TN, το Sparbot, έχει σχεδιαστεί να παίζει μόνο limit heads-up πόκερ. Η ανάπτυξη του

έγινε το 2002, χρησιμοποιώντας ιδέες της θεωρίας παιγνίων (game theory). Η θεωρία παιγνίων είναι ένας τομέας των εφαρμοσμένων μαθηματικών που εφαρμόζεται σε πολλούς τομείς επιστημών και προσπαθεί να καταγράψει τις μαθηματικές σχέσεις στις οποίες η επιτυχία ενός ατόμου στις επιλογές που κάνει, εξαρτάται από τις επιλογές των άλλων ατόμων. Παιχνίδια όπως το πόκερ, με ανταγωνισμό μεταξύ των παικτών και στα οποία ένας παίκτης κερδίζει σε βάρος ενός άλλου, ήταν τα πρώτα που μελετήθηκαν από την θεωρία παιγνίων.

Ιδιαίτερο ενδιαφέρον στην θεωρία παιγνίων παρουσιάζει η ισορροπία Nash (Nash equilibrium) που πήρε την ονομασία της από τον John Nash [11], ο οποίος και την πρότεινε το 1950. Ισορροπία Nash είναι η αντίληψη της επίλυσης ενός παιχνιδιού, δύο ή περισσότερων παικτών, στην οποία κάθε παίκτης ξέρει την ισορροπία μεταξύ των στρατηγικών των άλλων παικτών και όμως κανένας δεν μπορεί να κερδίσει κάτι αλλάζοντας μόνο αυτός την στρατηγική του. Αν υποθέσουμε ότι ένας από τους παίκτες μαθαίνει τις στρατηγικές των άλλων παικτών και έχοντας γνώση αυτών των στρατηγικών, που παραμένουν αναλλοίωτες, δεν μπορεί να αλλάξει την στρατηγική του με τρόπο ώστε να ωφεληθεί, τότε εμφανίζεται ισορροπία Nash.

Μία βέλτιστη (optimal) στρατηγική είναι αυτή που μεγιστοποιεί το κέρδος και ελαχιστοποιεί την απώλεια απέναντι σε οποιονδήποτε αντίπαλο. Η βέλτιστη στρατηγική είναι ο ασφαλέστερος τρόπος που μπορεί να παίξει κανείς όταν η στρατηγική του αντιπάλου είναι άγνωστη. Με άλλα λόγια, αν ο αντίπαλος παίζει με μία βέλτιστη στρατηγική, τότε ο καλύτερος τρόπος να τον αντιμετωπίσουμε είναι να παίξουμε με την ίδια (βέλτιστη) στρατηγική.

Το PsOpti προσεγγίζει το πρόβλημα του πόκερ με αρχές της θεωρίας παιγνίων, δηλαδή αναπτύχθηκε πάνω στην ιδέα ότι υπάρχει μία στρατηγική ισορροπίας και για τους δύο παίκτες. Στόχος της ανάπτυξης ήταν να βρεθεί μία στρατηγική πολύ κοντά στη βέλτιστη. Επειδή ο χώρος καταστάσεων στο πόκερ είναι τεράστιος, απαιτούνται να γίνουν προσεγγίσεις. Λόγω αυτών των προσεγγίσεων η στρατηγική του PsOpti θεωρείται ψευδο-βέλτιστη(pseudo-optical). Δεν χρησιμοποιεί μοντελοποίηση του αντιπάλου αλλά επιλέγει κινήσεις που βελτιστοποιούν το κέρδος του σε κάθε περίπτωση και για κάθε αντίπαλο. Γι’ αυτό δεν εγγυάται ότι θα έχει το μέγιστο κέρδος, αλλά σίγουρα εξασφαλίζει ότι θα χάσει πολύ δύσκολα.

Σε όλα τα πειράματα που έχει συμμετάσχει, το PsOpti πήγε πολύ καλά. Έχει καταφέρει να αντέξει σε ένα παγκόσμιας κλάσης διαγωνισμό 7000 παιχνιδιών απέναντι σε κορυφαίους παίκτες. Όμως, το PsOpti έχει δύο προβλήματα που είναι έμφυτα σε παίκτες ΤΝ που σχεδιάζονται και παίζουν με ψευδο-βέλτιστες στρατηγικές:

- Οι προσεγγίσεις που απαιτούνται για την μείωση του χώρου καταστάσεων εισάγουν αδυναμίες και αυτές οι αδυναμίες είναι μόνιμες. Αυτό σημαίνει ότι αν ο αντίπαλος ενός ψευδο-βέλτιστου

παίκτη TN ανακαλύψει μία αδυναμία του, τότε θα μπορεί να την εκμεταλλεύεται συνέχεια.

- Επειδή δεν προσπαθούν να εκμεταλλευτούν τον αντίπαλο, ένας δυνατός παίκτης μπορεί να παίζει με τέτοιο στυλ που να του επιτρέπει να εντοπίσει τις αδυναμίες τους. Έτσι, ο αντίπαλος μπορεί να τους «εξερευνήσει» χωρίς να τιμωρείται για αυτήν την αρκετά προβλέψιμη συμπεριφορά του.

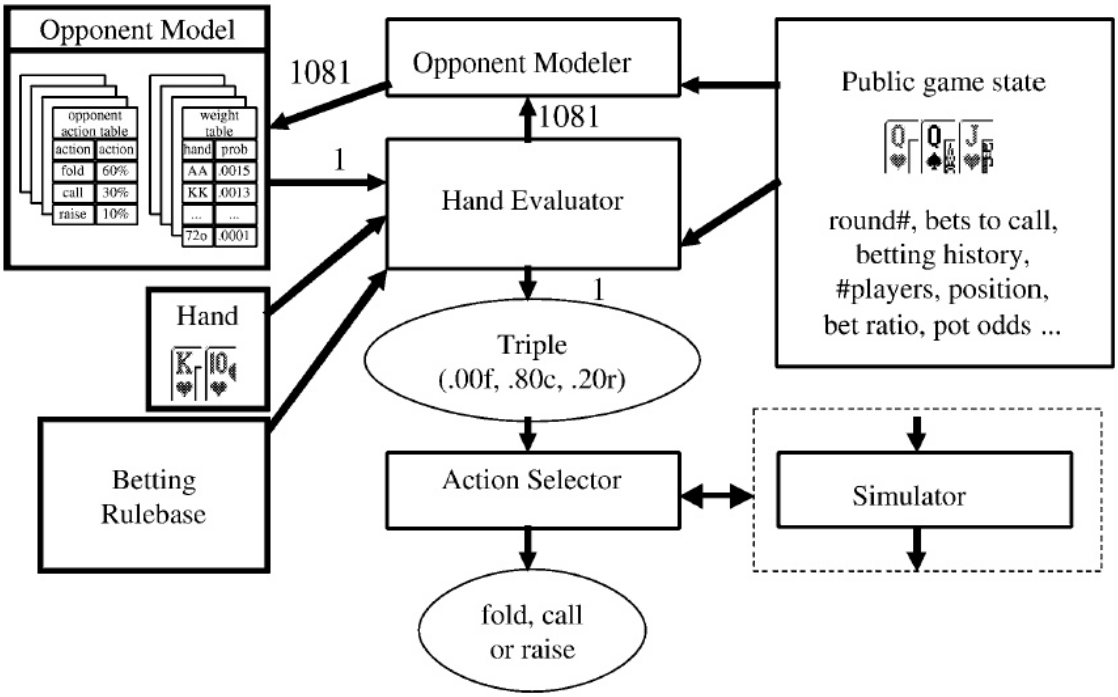
Παρά τα προβλήματα που θεωρητικά υπάρχουν σε αυτήν την κατηγορία προγραμμάτων, το PsOpti αποδεικνύει στην πράξη ότι είναι ένας ισχυρός αντίπαλος.

1.4 POKI

Το Poki σχεδιάστηκε το 1999, για να παίζει σε full-ring limit τραπέζια (δηλαδή σε τραπέζια με 10 παίκτες περιορισμένου πονταρίσματος). Έχει αποδείξει ότι είναι σταθερός νικητής σε διαγωνισμούς απέναντι σε ανθρώπους με εικονικά χρήματα που διεξάγονται είτε σε Internet Relay Chat (IRC) είτε στον server της ερευνητικής ομάδας πόκερ του πανεπιστημίου της Αλμπέρτα. Σε full-ring παιχνίδια θεωρείται ότι παίζει με ένα ενδιάμεσο επίπεδο δύναμης. Σε παιχνίδια με λιγότερους παίκτες γίνεται πιο αδύναμος αλλά εξακολουθεί να είναι ένας από τους κορυφαίους πράκτορες πόκερ.

Ενώ το PsOpti προσπαθεί να προσεγγίσει μία βέλτιστη στρατηγική το Poki προσπαθεί να βρει μία στρατηγική μέγιστου κέρδους. Όταν ο αντίπαλος μας δεν παίζει με τη βέλτιστη στρατηγική τότε μπορεί να υπάρχει μία μη-βέλτιστη στρατηγική η οποία θα μεγιστοποιεί το κέρδος μας. Το Poki χρησιμοποιεί μοντελοποίηση του αντιπάλου (opponent modeling) σε συνδυασμό με κανόνες που δημιουργήθηκαν από ειδικούς (expert rules), ώστε να αναγνωρίζεται η στρατηγική του αντιπάλου και να εκμεταλλεύονται οι αδυναμίες της [3][4][13].

Στο Σχήμα 2 βλέπουμε την αρχιτεκτονική του προγράμματος Poki [3].



Σχήμα 2 "Αρχιτεκτονική Poki"

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό εξετάστηκαν οι έννοιες των παιχνιδιών πλήρους και ελλιπούς πληροφορίας. Επίσης έγινε μια συνοπτική ανάλυση τριών γνωστών προγραμμάτων πόκερ τεχνητής νοημοσύνης και παρουσιάστηκαν οι αρχιτεκτονικές τους. Στο επόμενο κεφάλαιο θα δούμε τις στρατηγικές που ακολουθούν οι παίκτες του πόκερ, καθώς και τους κανόνες του παιχνιδιού.

ΚΕΦΑΛΑΙΟ 2 - ΠΑΙΧΝΙΔΙ ΤΟΥ ΠΟΚΕΡ

ΕΙΣΑΓΩΓΗ

Το Texas Hold'em είναι η πιο δημοφιλής παραλλαγή πόκερ και στο εξής θα αναφερόμαστε σε αυτό απλά ως πόκερ. Είναι παιχνίδι που απαιτεί πολύ καλή ικανότητα και λίγη τύχη. Ο παίκτης που έχει καλή στρατηγική σπάνια θα χρειαστεί τύχη για να έχει κέρδος. Το παιχνίδι παίζεται με την γνωστή τράπουλα των πενήντα δύο φύλλων, έχει τέσσερα χρώματα και δεκατρία φύλλα ανά χρώμα. Θα χρησιμοποιούνται οι εξής συμβολισμοί για τα χρώματα: μπαστούνια (♠), σπαθιά (♣), κούπες (♥), καρό (♦) και για την αξία των φύλλων: δύο(2), τρία(3), τέσσερα(4), πέντε(5), έξι(6), εφτά(7), οκτώ(8), εννέα(9), δέκα(10), βαλές(J), ντάμα(Q), παπάς(K) και άσσος (A).

Μπορούν να συμμετέχουν στο ίδιο τραπέζι από δύο έως δώδεκα παίκτες. Στα πλαίσια της πτυχιακής εργασίας θα εξετασθεί το heads up παιχνίδι, δηλαδή το παιχνίδι με δύο μόνο παίκτες.

2.1 ΚΑΝΟΝΕΣ ΡΟΗΣ

Η κάθε παρτίδα πόκερ χωρίζεται στα εξής στάδια:

Pre-Flop

Στην αρχή της παρτίδας και εναλλάξ ορίζεται ο ένας παίκτης ως dealer, δηλαδή ο παίκτης που μιλάει τελευταίος. Στους παίκτες μοιράζονται από δύο κρυφά προς τον αντίπαλο φύλλα και τοποθετούνται τα blinds που είναι τα υποχρεωτικά πονταρίσματα που πρέπει να μπουν σε αυτό το στάδιο. Ο dealer βάζει το big blind (π.χ. 20 chips) και ο αντίπαλος βάζει το small blind (π.χ. 10 chips) το οποίο είναι το μισό του big blind.

Ακολουθεί ένας γύρος πονταρίσματος όπου ο παίκτης που δεν είναι dealer μιλάει πρώτος και μπορεί είτε να ισοφαρίσει το ποσό του αντιπάλου (να κάνει call), είτε να αυξήσει το ποσό του αντιπάλου του (να κάνει raise), ή να πάει πάσο (fold) χάνοντας την παρτίδα. Αν ο παίκτης αυτός κάνει call τότε ο dealer μπορεί να κάνει check, δηλαδή να μην τοποθετήσει άλλο ποσό αφού είναι ίσα τα πονταρίσματα του γύρου, ή να κάνει raise και να ξαναμιλήσει ο αντίπαλος. Αυτό επαναλαμβάνεται μέχρι να ισοφαριστούν τα πονταρίσματα και έτσι ολοκληρώνεται ο Pre-Flop γύρος.

Flop (Pre-Turn)

Στη συνέχεια, περνάμε στον flop γύρο όπου ανοίγουν τρία κοινά φύλλα στο τραπέζι και ακολουθεί νέος γύρος πονταρίσματος. Μόλις ολοκληρωθεί και αυτός ο γύρος πονταρίσματος περνάμε στο Turn.

Turn (Pre-River)

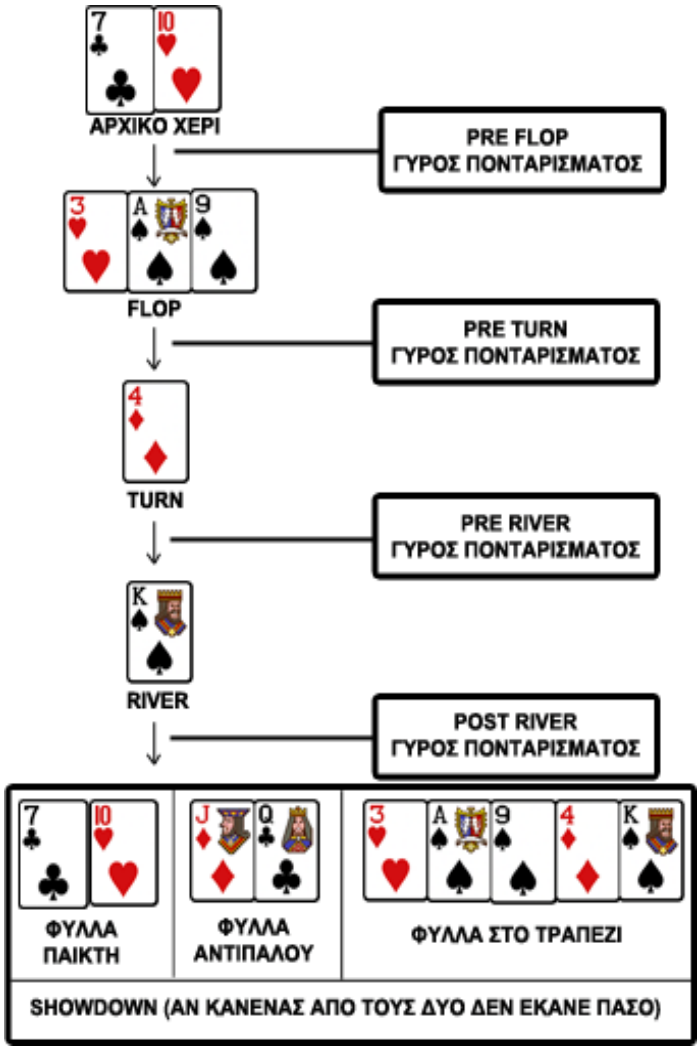
Σε αυτόν τον γύρο ανοίγει άλλο ένα κοινό φύλλο στο τραπέζι (το τέταρτο) και ακολουθεί νέος γύρος ποντάρισματος. Μόλις ολοκληρωθεί και αυτός ο γύρος ποντάρισματος περνάμε στο Turn.

River

Σε αυτόν τον γύρο ανοίγει το πέμπτο και τελευταίο κοινό φύλλο στο τραπέζι και ακολουθεί νέος γύρος ποντάρισματος.

Showdown

Αν και μετά την ολοκλήρωση του river έχουν παραμείνει και οι δύο παίκτες στην παρτίδα, τότε δείχνουν τα φύλλα τους και ο παίκτης που έχει την καλύτερη πεντάδα φύλλων από τα εφτά φύλλα (τα δύο κρυφά προς τον αντίπαλο και τα πέντε κοινά) κερδίζει το pot, δηλαδή κερδίζει το συνολικό ποσό που έχει ποντάριστεί σε όλους τους γύρους. Η αξία μιας πεντάδας φύλλων καθορίζεται από τον συνδυασμό που σχημάτισαν, όπως παρουσιάζεται στο παρακάτω υποκεφάλαιο.



Σχήμα 3 "Ροή ενός Παιχνιδιού Πόκερ"

2.2 ΣΥΝΔΥΑΣΜΟΙ ΦΥΛΛΩΝ

Η δύναμη ενός χεριού πέντε φύλλων απεικονίζεται στον Πίνακα 1. Τα πέντε φύλλα του χεριού είναι ο ισχυρότερος συνδυασμός που σχηματίζεται από το σύνολο των δύο φύλλων που κρατάει ο παίκτης στο χέρι και τα πέντε κοινά φύλλα που υπάρχουν στο τραπέζι. Ο καλύτερος συνδυασμός είναι το Straight Flush, ενώ ο χειρότερος το High Card. Για τον καθορισμό του νικητή στο Showdown, ο παίκτης με τον συνδυασμό που βρίσκεται υψηλότερα στον Πίνακα 1 κερδίζει.

Πίνακας 1 "Κατάταξη συνδυασμών φύλλων σε φθίνουσα σειρά δύναμης και συχνότητα εμφάνισης τους"

Συνδυασμός Φύλλων	Συχνότητα Εμφάνισης	Παράδειγμα
Straight Flush (Κέντα Χρώμα)	0.0015%	7♥ 6♥ 5♥ 4♥ 3♥
Four of a Kind (Καρέ)	0.0240%	9♦ 9♥ 9♣ 9♠ K♥
Foul House (Φουλ)	0.1441%	Q♣ Q♥ Q♠ 2♦ 2♥
Flush (Χρώμα)	0.1965%	4♣ J♣ 9♣ K♣ 6♣
Straight (Κέντα)	0.3925%	A♠ K♥ Q♥ J♣ 10♦
Three of a Kind (Τρία όμοια)	2.1128%	2♣ 2♥ 2♠ A♠ 7♦
Two Pairs (Δύο ζεύγη)	4.7539%	A♣ A♦ 5♦ 5♠ J♥
One Pair (Ένα ζεύγος)	42.257%	J♦ J♦ 4♥ 8♠ 2♣
High Card (Μεγαλύτερο Φύλλο)	49.882%	A♠ 6♦ K♥ 3♣ 8♣

Παρακάτω γίνεται μια αναλυτική επεξήγηση των συνδυασμών.

- Straight Flush: Ο δυνατότερος συνδυασμός στο πόκερ, πέντε φύλλα που σχηματίζουν Flush και Straight μαζί (δες παρακάτω). Η δύναμη του Straight Flush κατηγοριοποιείται ανάλογα με το υψηλότερο φύλλο του Straight (ο άσσος μπορεί να χρησιμοποιηθεί και ως χαμηλότερο φύλλο σε ένα Straight από άσσο μέχρι το πέντε, αλλά και ως υψηλότερο στο Straight από δέκα έως άσσο). Το Straight Flush από το δέκα έως τον άσσο είναι ο πιο δυνατός συνδυασμός του παιχνιδιού και ονομάζεται Royal Flush.

- Four of a Kind: Τέσσερα φύλλα ίδιας αξίας και ένα οποιοδήποτε πέμπτο (kicker). Η σύγκριση γίνεται βάση της αξίας του φύλλου της τετράδας. Ο αξία του kicker ελέγχεται σε περίπτωση που και οι δύο αντίπαλοι έχουν στον συνδυασμό καρτών της ίδιας αξίας.

- Full House: Τρία φύλλα ίδιας αξίας και ένα ζευγάρι. Η αξία του υπολογίζεται από την αξία της τριάδας, ενώ σε περίπτωση ισοπαλίας ελέγχεται και η αξία του ζευγαριού.

- Flush: Πέντε οποιαδήποτε φύλλα του ίδιου χρώματος. Μεγαλύτερο θεωρείται το Flush με το υψηλότερο φύλλο, ενώ σε περίπτωση ισοπαλίας ελέγχονται και τα επόμενα. Στο πόκερ όλα τα χρώματα έχουν την ίδια αξία.

- Straight: Πέντε συνεχόμενα φύλλα. Η αξία του υπολογίζεται βάσει του φύλλου με την μεγαλύτερη αξία.

- Three of a Kind: Τρία φύλλα ίσης αξίας, και δύο οποιοιδήποτε kickers. Πρώτα ελέγχεται η αξία της τριάδας και αν χρειαστεί οι δύο kickers(ο υψηλότερος πρώτα).

- Two Pairs: Ένα ζευγάρι φύλλων κάποιας αξίας, άλλο ένα ζευγάρι φύλλων κάποιας άλλης αξίας και ένα kicker. Πάντα ελέγχεται το ζευγάρι με την μεγαλύτερη αξία, μετά αν χρειαστεί το ζευγάρι χαμηλότερης αξίας και μετά ο kicker.

- One Pair: Ένα ζευγάρι και τρία kickers διαφορετικής αξίας μεταξύ τους. Ελέγχεται πρώτα η αξία του ζευγαριού και έπειτα ένας-ένας ο κάθε kicker, από τον υψηλότερο στον χαμηλότερο.

- High Card: Πέντε κάρτες οι οποίες δεν συγκροτούν κανέναν από τους παραπάνω συνδυασμούς. Η σύγκριση γίνεται με βάση το υψηλότερο φύλλο, όπως στο Flush.

2.3 ΣΤΡΑΤΗΓΙΚΕΣ ΑΝΘΡΩΠΙΝΩΝ ΠΑΙΚΤΩΝ

Όταν προσπαθούμε να κατηγοριοποιήσουμε τους παίκτες σε ένα παιχνίδι πόκερ, είναι χρήσιμο να σκεπτόμαστε με όρους που περιγράφουν το στυλ παιξίματος τους, δηλαδή τις στρατηγικές τους. Οι βασικοί όροι που χρησιμοποιούνται για το

σκοπό αυτόν είναι, χαλαρός (loose) ή σφιχτός (tight), και επιθετικός (aggressive) ή παθητικός (passive).

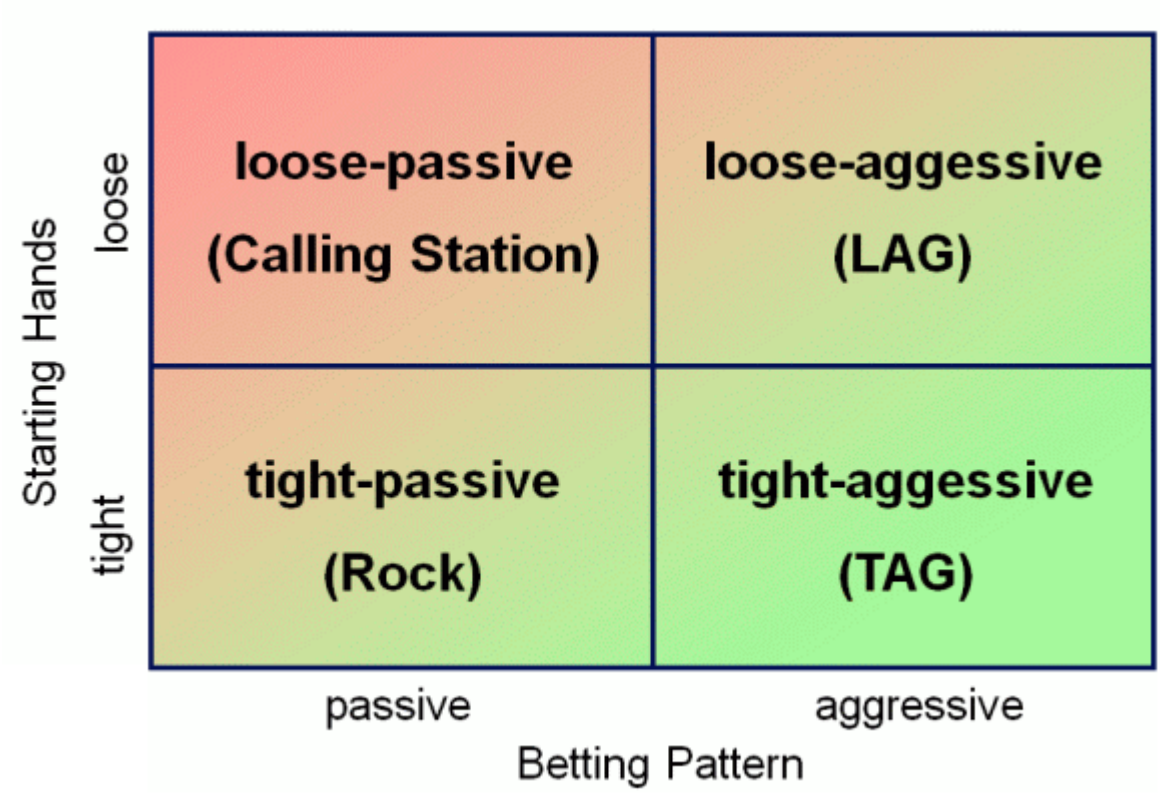
Για την διάκριση των όρων αυτών εξετάζονται δύο χαρακτηριστικά στο στυλ παιξίματος του παίκτη:

- Αρχικό χέρι (starting hand): Ο παίκτης παίζει πολλά αρχικά χέρια (loose) ή μόνο λίγα (tight);
- Μοτίβο ποντάρισματος (betting pattern): Ο παίκτης ποντάρει συχνά (aggressive) ή σπάνια (passive);

Με τον όρο “αρχικό χέρι”, εννοούμε τα δύο κλειστά φύλλα που μοιράζονται στον παίκτη στο Pre-Flop. Ένας loose παίκτης θα παίξει πολλά από τα αρχικά χέρια που θα του μοιραστούν, δηλαδή θα φτάσει στην FLOP κατάσταση. Αντίθετα, ένας tight παίκτης έχει μεγάλες απαιτήσεις από την δύναμη του αρχικού χεριού. Στατιστικά ένας loose παίκτης, παίζει το 40% των αρχικών χεριών, ενώ ο tight μόνο το 15%.

Το μοτίβο ποντάρισματος είναι η σχέση μεταξύ του Bet, του Raise και του Call. Αν ένας παίκτης κάνει Bet ή Raise τις διπλάσιες φορές από τις οποίες κάνει Call τότε θεωρείται aggressive, αλλιώς θεωρείται passive.

Όπως αναφέρει ο Billings [6], από τις απαιτήσεις στα αρχικά χέρια και τα μοτίβα ποντάρισματος μπορούμε να διακρίνουμε τέσσερα στυλ παιξίματος, όπως φαίνεται στο Σχήμα 4.



Σχήμα 4 "Βασικές Στρατηγικές Πόκερ"

2.3.1 Tight-Passive (Rock)

Στο στυλ Rock (βράχος), ο παίχτης παίζει ελάχιστα αρχικά χέρια και σπάνια θα κάνει bet ή raise. Σε περίπτωση όμως που το κάνει, είναι σχεδόν σίγουρο ότι έχει ένα δυνατό χέρι.

Πλεονεκτήματα του tight-passive:

- Περιορισμός στα καλά αρχικά χέρια, ο οποίος συνήθως μειώνει τις απώλειες.
- Σε τραπέζια περισσότερων των 2 ατόμων όπου κάθονται πολλοί επιθετικοί παίκτες, δε χρειάζεται να κάνει ο ίδιος bet ή raise προδίδοντας το δυνατό του φύλλο.

Μειονεκτήματα του tight-passive:

- Με τον παθητικό τρόπο του παιχνιδιού του, δεν παίρνει το μέγιστο κέρδος όταν παίζει τα καλά χέρια. Επιπλέον, δίνει στους αντιπάλους του την δυνατότητα να αποκτήσουν ένα καλύτερο χέρι, καθώς δεν προστατεύει επαρκώς τα δυνατά του χέρια (π.χ. υπάρχει κίνδυνος ο αντίπαλος να ολοκληρώσει Straight ή Flush).

Βέλτιστη στρατηγική εναντίον του tight-passive:

- Αν ο Rock κάνει bet ή raise στο Pre-Flop, τότε είναι καλύτερα να κάνεις fold, εκτός κι αν έχεις ένα πάρα πολύ καλό αρχικό χέρι.

2.3.2 Loose-Passive (Calling Station)

Ο παίκτης αυτού του στυλ παίζει πολλά αρχικά χέρια, αλλά σπάνια κάνει bet ή raise. Πολλοί αρχάριοι παίκτες του πόκερ έχουν αυτό το στυλ. Είναι σχεδόν αδύνατο για έναν παίκτη να έχει κέρδος μακροπρόθεσμα, χρησιμοποιώντας το loose-passive στυλ.

Πλεονεκτήματα του loose-passive:

- Όπως και στο tight-passive, σε τραπέζια περισσότερων των 2 ατόμων όπου κάθονται πολλοί επιθετικοί παίκτες, δε χρειάζεται να κάνει ο ίδιος bet ή raise προδίδοντας το δυνατό του φύλλο.

Μειονεκτήματα του loose-passive:

- Αφού παίζει πολλά αδύναμα αρχικά χέρια, υπάρχει συνεχώς ο κίνδυνος να έχει το δεύτερο ισχυρότερο χέρι στο παιχνίδι (π.χ. A♥10♠ ενάντια σε A♣K♦).
- Με τον παθητικό τρόπο του παιχνιδιού του, δεν παίρνει το μέγιστο κέρδος όταν παίζει τα καλά χέρια.

Βέλτιστη στρατηγική ενάντια σε loose-passive:

- Πρέπει να κάνεις bet μόνο όταν έχεις καλό αρχικό χέρι, καθώς ένας Calling Station θα κάνει call ούτως η άλλως.
- Αν κάνει bet και το αρχικό σου χέρι είναι μέτριας αξίας τότε είναι καλύτερα να κάνεις fold.

2.3.3 Tight-Aggressive (TAG)

Ένας παίκτης TAG παίζει λίγα αρχικά χέρια και συνηθίζει να κάνει bet ή raise και σπάνια θα κάνει call. Γενικά πρόκειται για το πιο κερδοφόρο στυλ παιξίματος.

Πλεονεκτήματα του tight-aggressive:

- Περιορισμός στα καλά αρχικά χέρια, ο οποίος συνήθως μειώνει τις απώλειες.
- Λόγω του επιθετικού στυλ και της σφικτής στρατηγικής, ο TAG συχνά κερδίζει έχοντας χέρι υψηλότερης αξίας, ή αναγκάζοντας τον αντίπαλο να κάνει fold.

Μειονεκτήματα του tight-aggressive:

- Λόγω του σφικτού στυλ παιχνιδιού οι αξία του χεριού που αποφασίζει να παίξει ο TAG, γίνεται εύκολα αντιληπτή σε έναν έμπειρο αντίπαλο, ο οποίος θα προσαρμόσει το στυλ παιχνιδιού του.
- Υπάρχει μεγάλος κίνδυνος να χάσει ολόκληρο το pot, όταν έχει ένα καλό, αλλά όχι τόσο ισχυρό χέρι (π.χ. υψηλό kicker).

Βέλτιστη στρατηγική ενάντια σε tight-aggressive:

- Καθώς ο TAG παίζει σφιχτά, προσπάθησε να κερδίσεις τα blinds, αναγκάζοντας τον να κάνει fold.
- Αν κάνει bet και το αρχικό σου χέρι είναι μέτριας αξίας τότε είναι καλύτερα να κάνεις fold.

2.3.4 Loose-Aggressive (LAG)

Ένας παίκτης LAG παίζει πολλά αρχικά χέρια και προτιμάει να κάνει bet και raise, παρά call. Πολλοί επαγγελματίες παίκτες πόκερ χρησιμοποιούν αυτήν τη στρατηγική σε τουρνουά με μεγάλη επιτυχία, αλλά είναι μια στρατηγική υψηλού κινδύνου που πρέπει να αποφεύγεται από άπειρους παίκτες.

Πλεονεκτήματα του loose-aggressive:

- Επειδή παίζει μεγάλο ποσοστό των αρχικών χεριών που του μοιράζονται, είναι δύσκολο για τον αντίπαλο να καταφέρει να “διαβάσει” τα φύλλα ενός LAG παίκτη.
- Λόγω του επιθετικού στυλ και της σφικτής στρατηγικής, ο LAG συχνά κερδίζει έχοντας χέρι υψηλότερης αξίας, ή αναγκάζοντας τον αντίπαλο να κάνει fold.
- Η επιθετικότητα του μπορεί να οδηγήσει τους αντιπάλους του στο λανθασμένο συμπέρασμα στο ότι ο LAG πάντα μπλοφάρει.

Μειονεκτήματα του loose-aggressive:

- Απαιτεί υψηλό επίπεδο ικανοτήτων, ώστε να μπορεί ο παίκτης να καταλάβει πότε είναι σε μειονεκτική θέση και να κάνει fold, αλλιώς θα χάσει μεγάλο ποσό ακολουθώντας αυτήν τη στρατηγική.

Βέλτιστη στρατηγική ενάντια σε loose-aggressive:

- Αν το αρχικό σου χέρι είναι μεγάλης αξίας, και ο LAG κάνει bet στο Pre-Flop, τότε καλό είναι να κάνεις raise για να τον εγκλωβίσεις.

2.3.5 Nit και Maniac

Επιπλέον, υπάρχουν άλλα δύο στυλ παιξίματος που εντοπίζονται και αυτά συχνά σε ένα παιχνίδι πόκερ.

Ο Nit παίζει δειλά, κάνοντας fold τα λιγοστά αρχικά χέρια που παίζει (<15%), λόγω του ότι ο αντίπαλος κάνει bet. Πολλοί Rock και TAG παίκτες είναι Nits. Εάν ένας Nit αντίπαλος κάνει call σε ένα bet σου, πρέπει να υποθέσεις πως έχει ένα πολύ υψηλό χέρι.

Ο Maniac είναι μία ακραία περίπτωση ενός loose-aggressive παίκτη. Είναι σημαντικό να μην πέσεις στην παγίδα να τον ακολουθήσεις αν δεν είσαι σίγουρος για το χέρι σου. Οι maniacs δεν μπορούν να ανταπεξέλθουν σε ένα παιχνίδι μεγάλης διάρκειας.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό παρουσιάστηκαν οι κανόνες ροής ενός παιχνιδιού πόκερ, καθώς και οι συνδυασμοί των φύλλων που μπορεί να επιτύχει ένας παίκτης, μαζί με τον τρόπο υπολογισμού της αξίας του καθένα. Επίσης έγινε μια αναλυτική περιγραφή των διαφορετικών στρατηγικών που ακολουθούν οι ανθρώπινοι παίκτες σε μια παρτίδα πόκερ, καταλήγοντας πως ο συνδυασμός των loose-aggressive είναι η καλύτερη επιλογή για έναν παίκτη με σχετική εμπειρία στο πόκερ, ενώ θα πρέπει να αποφεύγονται οι στρατηγικές passive, καθώς αποδεικνύονται οι λιγότερο αποτελεσματικές.

Στο επόμενο κεφάλαιο θα δούμε πως σχεδιάστηκαν οι κανόνες του πόκερ και οι συνδυασμοί νίκης για το πρόγραμμα που δημιουργήθηκε, και επίσης πώς οι παραπάνω στρατηγικές προσαρμόστηκαν στον παίκτη TN που αναπτύχθηκε σε αυτήν την εργασία.

ΚΕΦΑΛΑΙΟ 3 - ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΜΕΘΟΔΟΛΟΓΙΑ

ΕΙΣΑΓΩΓΗ

Για την σωστή λειτουργία της εφαρμογής είναι αναγκαία η ενσωμάτωση σε αυτήν, όλων των εννοιών και των κανόνων ροής του παιχνιδιού, όπως αυτά παρουσιάστηκαν στο Κεφάλαιο 2. Όσον αφορά τη συμπεριφορά του παίκτη TN, είναι απαραίτητος ο ορισμός κάποιων επιπλέον χαρακτηριστικών, τα οποία θα επιτρέπουν τη σωστή ανάγνωση της κατάστασης του παιχνιδιού. Επίσης πρέπει να οριστούν οι στρατηγικές του παίκτη TN, αντιπροσωπεύοντας τις συμπεριφορές των ανθρώπινων παικτών, προσθέτοντας βέβαια μία πτυχή τυχαιότητας στην επιλογή τους. Τα χαρακτηριστικά αυτά και ο σχεδιασμός γενικότερα του παιχνιδιού παρουσιάζονται στα παρακάτω υποκεφάλαια.

3.1 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

Εδώ θα παρουσιαστεί ο τρόπος που γίνεται η κωδικοποίηση των χαρακτηριστικών του παιχνιδιού, περιγράφοντας το νόημα των εισόδων και θα επεξηγηθεί ο τρόπος προσδιορισμού τους.

3.1.1 Αναπαράσταση Φύλλων

Για την αναπαράσταση ενός φύλλου λαμβάνονται υπόψιν δύο παράγοντες, η αξία του φύλλου (RANK), από το δύο μέχρι τον άσσο και το χρώμα του (SUIT) ♠, ♥, ♣, ♦. Για την επιλογή της βέλτιστης μορφής αναπαράστασης των φύλλων δοκιμάστηκαν πολλές μορφές, οι οποίες κατά τη διάρκεια των δοκιμών της εφαρμογής μεταλλάχθηκαν ή απορρίφθηκαν. Παρακάτω παρουσιάζονται δυο μορφές αναπαράστασης φύλλων που τελικά απορρίφθηκαν καθώς και η βέλτιστη μορφή αναπαράστασης, η οποία επιλέχθηκε.

<u>Μορφή A, card(Rank, Suit) :</u>

card(2, d), card(3, d), … card(k, d), card(a, d)

card(2, h), card(3, h), … card(k, h), card(a, h)

card(2, c), card(3, c), … card(k, c), card(a, c)

card(2, s), card(3, s), … card(k, s), card(a, s)

Το γεγονός card έχει ως πρώτο όρισμα την αξία του φύλλου, η οποία μπορεί να πάρει τις τιμές 2,3,4,5,6,7,8,9,10,j,q,k,a και ως δεύτερο όρισμα το χρώμα του φύλλου, έχοντας ως τιμή ένα από τα d,h,c,s, τα οποία συμβολίζουν αντίστοιχα τα καρό ♦, κούπα ♥, σπαθί ♣, μπαστούνι ♠. Η μορφή A αν και είναι μια απλή και κατανοητή αναπαράσταση των φύλλων, απορρίφθηκε διότι η χρήση του γεγονότος card για τον προσδιορισμό του κάθε φύλλου αποτελεί πλεονασμό, περιπλέκοντας ανούσια τον κώδικα Prolog αυτής της εργασίας που αγγίζει τις 2000 γραμμές.

Μορφή Β, (Rank-Suite) :

(2-d), (3-d), … (k-d), (a-d)

(2-h), (3-h), … (k-h), (a-h)

(2-c), (3-c), … (k-c), (a-c)

(2-s), (3-s), … (k-s), (a-s)

Η μορφή Β χρησιμοποιήθηκε στα πρώτα στάδια ανάπτυξης που προγράμματος με ικανοποιητικά αποτελέσματα. Η αναπαράσταση ενός φύλλου γίνεται με μια ενιαία αναπαράσταση της αξίας και του χρώματος του φύλλου, τα οποία διαχωρίζονται μεταξύ τους με μια παύλα(-). Οι τιμές που μπορεί να πάρει η αξία και το χρώμα του φύλλου, είναι οι ίδιες με αυτές της μορφής Α. Το μειονέκτημα της ήταν ότι για την έξοδο των δεδομένων από την Prolog σε Java μέσω του JPL, απαιτούνταν επιπλέον μηχανισμοί μορφοποίησης των δεδομένων από πλευράς Java, γεγονός το οποίο οδήγησε στην μετατροπή της σε μια απλούστερη μορφή Head|Tail όπως βλέπουμε παρακάτω.

Βέλτιστη μορφή:

Στο prolog πρόγραμμα που δημιουργήθηκε στην εργασία αυτή, επιλέχθηκε η αναπαράσταση του κάθε φύλλου με τη μορφή ΑΞΙΑ|ΧΡΩΜΑ (RANK|SUIT).

Η αξία του φύλλου μπορεί να πάρει τις τιμές: 2,3,4,5,6,7,8,9,10,j,q,k,a.

Ενώ το χρώμα: s για το ♠, h για το ♥, c για το ♣, d για το ♦

Παρακάτω δίνονται μερικά παραδείγματα αναπαράστασης φύλλων:

5♦ συμβολίζεται ως 5|d

A♣ συμβολίζεται ως a|c

Q♠ συμβολίζεται ως q|s

7♥ συμβολίζεται ως 7|h

3♦ συμβολίζεται ως 3|d

10♣ συμβολίζεται ως 10|c

3.1.2 Χρήσιμες Μεταβλητές

Στον Πίνακα 2 παρουσιάζονται μερικές μεταβλητές του προγράμματος, οι οποίες αντιπροσωπεύουν κάποια βασικά χαρακτηριστικά που χρησιμοποιούνται από τον παίκτη ΤΝ, ή είναι χρήσιμα στον έλεγχο ροής της εφαρμογής.

Πίνακας 2 "Μεταβλητές παιχνιδιού"

ΟΝΟΜΑ ΜΕΤΑΒΛΗΤΗΣ	ΠΕΡΙΓΡΑΦΗ
Strategy	Η στρατηγική που ακολουθεί ο παίκτης ΤΝ.
GameState	Η τρέχουσα κατάσταση στην οποία βρίσκεται το παιχνίδι (Pre-Flop, Pre-Turn κτλ).
HandStrength	Η δύναμη του χεριού του παίκτη.
PotentialStrength	Δυνατότητα βελτίωσης του χεριού.
OpponentAction	Ο ενέργεια του αντιπάλου.
Action	Η ενέργεια του παίκτη ΤΝ.
ActionsLeft	Ο αριθμός των ενεργειών που απομένουν συνολικά στους δύο παίκτες στην τρέχουσα κατάσταση παιχνιδιού. Κάθε κατάσταση αρχίζει με τον μέγιστο αριθμό ActionsLeft = 6.
Pot	Το συνολικό ποσό που έχουν ποντάρει και οι δύο παίχτες στο τραπέζι από το Pre-Flop μέχρι την τρέχουσα κατάσταση παιχνιδιού.
MyChips	Το ποσό των chips που κατέχει ο παίκτης ΤΝ.
OpponentChips	Το ποσό των chips που κατέχει ο αντίπαλος του παίκτη ΤΝ.
MyChipStatus	Η τιμή της ακέραιας διαίρεσης του MyChips προς το Pot.
OpponentChipStatus	Η τιμή της ακέραιας διαίρεσης του OpponentChips προς το Pot.
ChipsLost	Το σύνολο των chips που έχασε ο παίκτης ΤΝ με την τρέχουσα στρατηγική.

3.1.3 Δύναμη του Χεριού (Hand Strength)

Το hand strength είναι μία καλή πρώτη προσέγγιση για την αξιολόγηση του χεριού του παίκτη ΤΝ. Αποτελεί μία από τις βασικότερες εισόδους του παίκτη ΤΝ για όλους τους γύρους της παρτίδας.

Ο τρόπος υπολογισμού του hand strength όταν η παρτίδα βρίσκεται στο Pre-Flop στάδιο αντιστοιχεί στην “απόλυτη” δύναμη του χεριού (μόνο δύο κάρτες στο Pre-Flop). Σε αυτή την εργασία θα αναφερόμαστε σε αυτό ως hand pair strength. Η δύναμη του εξαρτάται φυσικά από την αξία της κάθε κάρτας, αλλά κι από το αν οι δύο αυτές κάρτες είναι του ίδιου χρώματος (suited) ή όχι (unsuited). Να θυμίσουμε πως στο πόκερ όλα τα χρώματα έχουν την ίδια αξία.

Η τιμή του hand pair strength υπολογίζεται από ένα σύνολο γεγονότων (facts) και κανόνων (rules), ένα μέρος των οποίων παρουσιάζεται στο Σχήμα 5.

```
handPairStrength(C1, C2, S):-
    handPair(C1, C2, S1),
    suited(C1, C2, S1, S2),
    S is S2.

suited(_|X, _|Y, S1, S2):-
    X == Y,
    S2 is S1 + 10.
suited(_|X, _|Y, S1, S2):-
    X \== Y,
    S2 is S1.

handPair(a|_, k|_, 87).
handPair(a|_, q|_, 80).
handPair(a|_, j|_, 65).
handPair(a|_, 10|_, 50).
handPair(a|_, 9|_, 45).
handPair(a|_, 8|_, 40).
handPair(a|_, 7|_, 35).
```

Σχήμα 5 “Τρόπος υπολογισμού του handPairStrength”

Όπως φαίνεται και στο παραπάνω σχήμα, η τιμή του hand pair strength μεγαλώνει κατά 10 μονάδες όταν οι δύο κάρτες είναι suited. Στον Πίνακα 3 και Πίνακα 4 που ακολουθούν, αναγράφεται η δύναμη του hand pair για όλους τους συνδυασμούς φύλλων, για suited και unsuited αντίστοιχα.

Πίνακας 3 "Τιμές του handPairStrength για unsuited φύλλα"

	A	K	Q	J	10	9	8	7	6	5	4	3	2
A	100												
K	87	90											
Q	80	75	85										
J	65	62	60	75									
10	50	48	46	45	55								
9	45	43	42	41	40	50							
8	40	38	37	36	35	34	45						
7	35	33	32	31	30	29	28	40					
6	30	28	27	26	25	24	23	22	35				
5	25	23	22	21	20	19	18	17	16	30			
4	20	18	17	16	15	14	13	12	11	10	25		
3	15	13	12	11	10	9	8	7	6	5	4	20	
2	10	7	6	5	4	3	2	1	2	3	3	3	15

Πίνακας 4 "Τιμές του handPairStrength για suited φύλλα"

	A	K	Q	J	10	9	8	7	6	5	4	3	2
A													
K	97												
Q	90	85											
J	75	72	70										
10	60	58	56	55									
9	55	53	52	51	50								
8	50	48	47	46	45	44							
7	45	43	42	41	40	39	38						
6	40	38	37	36	35	34	33	32					
5	35	33	32	31	30	29	28	27	26				
4	30	28	27	26	25	24	23	22	21	20			
3	25	23	22	21	20	19	18	17	16	15	14		
2	20	17	16	15	14	13	12	11	12	13	13	13	

Από την άλλη πλευρά, όταν η παρτίδα πόκερ βρίσκεται σε οποιοδήποτε άλλο στάδιο εκτός του Pre-Flop, δηλαδή σε ένα εκ των Pre-Turn, Pre-River ή Post-River, ο τρόπος υπολογισμού του hand strength γίνεται με διαφορετικό τρόπο. Σε αυτήν την περίπτωση εξετάζεται αν έχει δημιουργηθεί κάποιος συνδυασμός με τα κρυφά φύλλα του παίκτη και τα φύλλα που έχουν ανοιχτεί στο τραπέζι. Σε κάθε πιθανό συνδυασμό δίνεται μία τιμή, με τον High Card να έχει την μικρότερη και το Straight Flush την υψηλότερη. Επίσης, ελέγχεται η αξία των φύλλων που συμμετέχουν στον συνδυασμό αυτόν, και αποδίδεται μία δευτερεύουσα τιμή ανάλογα με την αξία τους (minor strength). Αυτές οι δύο τιμές αθροίζονται και συνιστούν το hand strength.

Στον Πίνακα 5 βλέπουμε τις αρχικές τιμές που αποδίδονται στους συνδυασμούς πριν προστεθεί σε αυτές το minor strength.

Πίνακας 5 “Αρχικές τιμές συνδυασμών χωρίς το minorStrength”

Συνδυασμός	Τιμή(ΑΤΣ)
Straight Flush	1020
Four of a Kind	1000
Full House	815
Flush	800
Straight	785
Three of a Kind	600
Two Pairs	400
One Pair	200
High Card	0

Οι αρχικές αυτές τιμές των συνδυασμών (ΑΤΣ) δεν προκύπτουν από κάποιον τύπο, αλλά δόθηκαν αυθαίρετα με γνώμονα μόνο την ευκολία χρήσης τους, καθώς θα τις χρειαστούμε και στο στάδιο του showdown όπου θα γίνεται η επιλογή του νικητή.

Για τον υπολογισμό του minor strength δίνονται οι τιμές από 1 έως 13, στα φύλλα από το 2 έως τον άσσο αντίστοιχα. Για τους συνδυασμούς Straight-Flush, Flush, Straight, και Four of a Kind, το minor strength αντιστοιχεί στην τιμή του φύλλου με την υψηλότερη αξία που συμμετέχει στον συνδυασμό. Σε αυτήν την περίπτωση έχουμε τον τύπο (3.1):

HandStrength = ΑΤΣ + minorStrength (3.1)

Για όλους τους υπόλοιπους συνδυασμούς, είναι αναγκαίο να υπολογιστεί και ένα δεύτερο minor strength που θα ικανοποιεί τον τύπο (3.2):

HandStrength = ΑΤΣ + (minorStrengthA * 13) + minorStrengthB (3.2)

Για παράδειγμα αν ο παίκτης κρατάει τα φύλλα K♠ 10♥ και στο τραπέζι έχουν ανοίξει τα φύλλα 4♣ K♥ K♦, τότε ο υπολογισμός του Hand Strength γίνεται ως εξής:

Three of a Kind (K♠ K♥ K♦) άρα ΑΤΣ = 600

minorStrengthA = (<αξία του K> * 13) = 12 * 13 = 156

minorStrengthB = <αξία του 10> = 9

HandStrength = 600 + 156 + 9 = 765

Ένα άλλο παράδειγμα όπου ο παίκτης κρατάει τα φύλλα 9♠ 2♠ και στο τραπέζι έχουν ανοίξει τα φύλλα 5♠ Q♠ 10♠.

Flush (9♠ 2♠ 5♠ Q♠ 10♠) άρα ΑΤΣ = 800

minorStrength = <αξία του Q> = 11

HandStrength = 800 + 11 = 811

Στο Σχήμα 6, παρατίθεται ένα απόσπασμα του κώδικα για τον υπολογισμό του hand strength στην περίπτωση του Flush.

```
handStrength(L, Strength):-
    isFlush(L, _, Highest) ->
        checkMinorStrength(Highest, Str),
        Strength is 800+Str;

checkMinorStrength(X, S):-
    isAce(X) -> S is 13;
    isKing(X) -> S is 12;
    isQueen(X) -> S is 11;
    isJack(X) -> S is 10;
    isTen(X) -> S is 9;
    isNine(X) -> S is 8;
    isEight(X) -> S is 7;
    isSeven(X) -> S is 6;
    isSix(X) -> S is 5;
    isFive(X) -> S is 4;
    isFour(X) -> S is 3;
    isThree(X) -> S is 2;
    isTwo(X) -> S is 1;
    S is 0.
```

Σχήμα 6 "Υπολογισμός hand strength στην περίπτωση του flush"

Σε αυτό το σημείο, αξίζει να σημειωθεί πως ο πιο γνωστός αλγόριθμος πόκερ για τον υπολογισμό της δύναμης του χεριού είναι ο EHS (Effective Hand Strength), ο οποίος αναπτύχθηκε από τους επιστήμονες Darse Billings, Dennis Papp, Jonathan Schaeffer και Duane Szafron [4]. Δημοσιοποιήθηκε για πρώτη φορά το 1998 και από τότε θεωρείται σημείο αναφοράς για την τεχνητή νοημοσύνη στο πεδίο του πόκερ και είναι η βάση για περαιτέρω έρευνες.

Ο αλγόριθμος αυτός είναι μια αριθμητική προσέγγιση για τον προσδιορισμό της δύναμης ενός χεριού πόκερ, όπου η τιμή του κυμαίνεται από το 0 έως το 1, σε σύγκριση με όλα τα άλλα πιθανά χέρια. Η βασική παραδοχή είναι ότι ο EHS αποτελείται από την τρέχουσα δύναμη του χεριού (Hand Strength) και τις δυνατότητες της να βελτιωθεί (PPOT) ή να επιδεινωθεί (NPOT) όπως φαίνεται στον τύπο (3.3).

$$EHS = HS \times (1 - NPOT) + (1 - HS) \times PPOT \quad (3.3)$$

Όπου:

EHS είναι η πραγματική δύναμη του χεριού.

HS είναι η τρέχουσα δύναμη του χεριού (δηλαδή χωρίς να λαμβάνει υπόψη τις δυνατότητες να βελτιωθεί ή να χειροτερέψει, ανάλογα με τα επερχόμενα κοινά φύλλα).

NPOT είναι η αρνητική δυνατότητα (δηλαδή η πιθανότητα το τρέχον χέρι, αν και ισχυρότερο, να επιδεινωθεί και γίνει ένα χαμένο χέρι).

PPOT είναι η θετική δυνατότητα (δηλαδή η πιθανότητα το τρέχον χέρι, αν και αδύναμο, να βελτιωθεί και να γίνει ένα νικητήριο χέρι).

Ο EHS είναι εφαρμόσιμος σε ένα μεγάλο εύρος παιχνιδιών πόκερ, όπως το Texas Holdem, το Omaha και άλλα. Δεδομένης της πολυπλοκότητας του αλγορίθμου, δεν μπορεί να υπολογισθεί με το χέρι και πρέπει να χρησιμοποιείται στο πλαίσιο της τεχνητής νοημοσύνης. Οι γνωστοί αλγόριθμοι πόκερ Loki και Poki που είδαμε στο πρώτο κεφάλαιο της εργασίας βασίζονται στον EHS.

Στο σχήμα 7 βλέπουμε τον ψευδοκώδικα για τον υπολογισμό του HandStrength με βάση τον αλγόριθμο EHS.

```
HandStrength(ourcards, boardcards) {

    ahead = tied = behind = 0
    ourrank = Rank(ourcards, boardcards)
    for each case(oppcards) {

        opprank = Rank(oppcards, boardcards)
        if (ourrank>opprank) ahead += 1
        else if (ourrank==opprank) tied += 1
        else behind += 1

    }
    handstrength=(ahead+tied/2)/ahead+tied+behind)
    return(handstrength)

}
```

Σχήμα 7 "Ψευδοκώδικας υπολογισμού HandStrength για τον EHS"

3.1.4 Δυνατότητα του Χεριού (Hand Potential)

Αν το hand strength ήταν το μόνο χαρακτηριστικό αξιολόγησης του χεριού, ο παίκτης TN δεν θα είχε την ικανότητα να προβλέψει πιθανές εκβάσεις της παρτίδας που θα άλλαζαν το παιχνίδι. Μετά το flop μένουν να ανοίξουν τα κοινά φύλλα του turn και του river. Αυτά τα φύλλα μπορεί να μεταβάλλουν σημαντικά την δύναμη του χεριού.

Γι' αυτόν το λόγο, χρησιμοποιούμε και το hand potential που στην ουσία χαρακτηρίζει την δυνατότητα του χεριού να βελτιωθεί. Ο έλεγχος για την ύπαρξη ή όχι του hand potential γίνεται μόνο σε δύο καταστάσεις του παιχνιδιού, στο Pre-Turn, όπου υπολείπονται άλλα δύο φύλλα στο τραπέζι, και στο Pre-River όπου υπολείπεται ένα. Στους ελέγχους αυτούς διαπιστώνεται αν το υπάρχον χέρι χαμηλής αξίας μπορεί να μετατραπεί σε Straight ή Flush στην αμέσως επόμενη κατάσταση παιχνιδιού.

Για παράδειγμα, αν υποθέσουμε ότι βρισκόμαστε στο Pre-Turn, τα κρυφά μας φύλλα είναι τα 6♦ 8♠ , και στο τραπέζι υπάρχουν τα φύλλα 5♣ 4♦ K♥, τότε υπάρχει hand potential, καθώς αν στο Turn το φύλλο είναι 7, θα δημιουργεί Straight. Σε ένα άλλο παράδειγμα, υποθέτουμε ότι είμαστε στο Pre-River τα φύλλα μας είναι A♣ 4♣ , και στο τραπέζι υπάρχουν τα φύλλα 8♦ 3♣ Q♣ 7♥, τότε επίσης υπάρχει hand potential, επειδή αν στο River το φύλλο είναι χρώματος ♣, θα δημιουργεί Flush.

Η χρήση του Hand Potential από τον παίκτη TN είναι αναγκαία καθώς είναι μια τεχνική που μειώνει ως ένα σημείο την προβλεψιμότητα που εμφανίζει ο

παίκτης στις επιλογές του. Αν σκεφτούμε πως χωρίς τον έλεγχο για Hand Potential, τα χέρια που θα παίζονταν από τον παίκτη TN θα ήταν μόνο τα ισχυρά, καταλαβαίνουμε πως μακροπρόθεσμα, η πιθανότητα κέρδους για αυτόν μειώνεται δραματικά, καθώς δεν είναι δυνατό να ελπίζει αποκλειστικά στο ότι το χέρι του είναι καλύτερο από του αντιπάλου, σε ένα τόσο πολύπλοκο παιχνίδι όπως το πόκερ.

Επίσης, θα πρέπει να γνωρίζουμε πως η εμμονή σε ένα χέρι το οποίο εμφανίζει Potential Strength, με συνεχή αύξηση των ποντάρισμάτων (Raise-reRaise), μπορεί να αποδειχθεί καταστροφική, αν στις επόμενες καταστάσεις του παιχνιδιού δεν σχηματιστεί ο επιθυμητός συνδυασμός φύλλων. Στο πρόγραμμα μας η ένταση με την οποία θα επιμείνει ο παίκτης TN σε ένα τέτοιο χέρι, καθορίζεται από την στρατηγική που ακολουθεί στο συγκεκριμένο σημείο.

Ενδιαφέρον παρουσιάζει και η κατάσταση κατά την οποία και οι δύο παίκτες έχουν ένα χέρι με δυνατότητα βελτίωσης, όπως φαίνεται παρακάτω:

Παίκτης TN: 6♥ 7♠

Αντίπαλος: J♥ Q♣

Κοινά φύλλα: 8♦ 9♣ 2♥

Αν στην επόμενη κατάσταση παιχνιδιού, στο Pre-River, έχουμε τα κοινά φύλλα 8♦ 9♣ 2♥ 10♠ , τότε και οι δύο παίκτες θα έχουν σχηματίσει Straight, με διαφορετική όμως δύναμη για τον καθένα, ο παίκτης TN στο κάτω άκρο 6♥ 7♠ 8♦ 9♣ 10♠, και ο αντίπαλος στο άνω άκρο 8♦ 9♣ 10♠ J♥ Q♣. Σε αυτήν την περίπτωση ο παίκτης TN κινδυνεύει να χάσει ένα μεγάλο ποσό των chips του.

Σε γενικές γραμμές, αν και πρόκειται για μια τεχνική με μεγάλο ρίσκο, ο καθορισμός των επιλογών που κάνει ο παίκτης TN συνυπολογίζοντας την ύπαρξη του Potential Strength του χεριού του, είναι απαραίτητος και συνήθως υποχρεώνει τον αντίπαλο να κάνει πάσο ή τον συμπαρασύρει σε ένα γύρο επικίνδυνων ποντάρισμάτων, αν δεν έχει ιδιαίτερα δυνατό χέρι, κερδίζοντας πολλές φορές σημαντικά ποσά.

Στο Σχήμα 8 παρατίθεται ένα απόσπασμα κώδικα για τον έλεγχο της ύπαρξης potential strength στο Pre-Turn.

```
potentialStrength(C1, C2, BC1, BC2, BC3, Strength):-
    isStraight(C1, C2, BC1, BC2, BC3, _, _) ->
        Strength is 1;
    isFlush(C1, C2, BC1, BC2, BC3, _, _) ->
        Strength is 1;
    Strength = 0.
```

Σχήμα 8 "Έλεγχος για potential strength στο Pre-Turn"

3.2 ΣΤΡΑΤΗΓΙΚΕΣ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΠΑΙΚΤΗ ΤΝ

Στο Κεφάλαιο 2.3, Στρατηγικές Ανθρώπινων Παικτών, αναλύθηκαν οι βασικές κατηγορίες στρατηγικών που ακολουθούν οι ανθρώπινοι παίκτες στο πόκερ. Μια παραλλαγή των στρατηγικών αυτών κωδικοποιήθηκε και προσαρμόστηκε στον παίκτη ΤΝ που αναπτύχθηκε σε αυτήν την εργασία, διατηρώντας τις βασικές αρχές της κάθε στρατηγικής. Θα αναφερόμαστε σε αυτές ως στρατηγικές ΤΝ όπως φαίνεται παρακάτω:

- Tight-Passive TN
- Loose-Passive TN
- Tight-Aggressive TN
- Loose-Aggressive TN

Για την διαμόρφωση των στρατηγικών ΤΝ έχει γραφεί ένα σύνολο κανόνων που έχουν ως είσοδο τις μεταβλητές που παρουσιάστηκαν στο Κεφάλαιο 3.1.2. Η μεταβλητή με την μεγαλύτερη σημασία για όλες τις στρατηγικές είναι το HandStrength (HandPairStrength για το Pre-Flop), η τιμή του οποίου ορίζει σε πολύ μεγάλο βαθμό την λειτουργία τους.

Στον Πίνακα 6 και Πίνακα 7, θα δούμε τη συμπεριφορά που έχουν οι στρατηγικές ΤΝ στο Pre-Flop, ανάλογα με την τιμή του HandPairStrength.

Πίνακας 6 "Τιμές του HandPairStrength για κάθε ενέργεια των στρατηγικών ΤΝ στο Pre-Flop"

Στρατηγικές ΤΝ	Τιμές HandPairStrength για κάθε ενέργεια				
	FOLD	CHECK	CALL	BET	RAISE
Tight Passive TN	< 35	< 35	35 – 44	> 44	>44
Loose Passive TN	< 6	< 45	6 – 44	>44	>44
Tight Aggressive TN	< 35	< 35	35 – 44	>34	>44
Loose Aggressive TN	< 6	< 24	6 - 24	>23	>24

Πίνακας 7 "Τιμές του HandPairStrength για τις οποίες γίνεται fold στο Pre-Flop"

	A	K	Q	J	10	9	8	7	6	5	4	3	2
A	100												
K	87	90											
Q	80	75	85										
J	65	62	60	75									
10	50	48	46	45	55								
9	45	43	42	41	40	50							
8	40	38	37	36	35	34	45						
7	35	33	32	31	30	29	28	40					
6	30	28	27	26	25	24	23	22	35				
5	25	23	22	21	20	19	18	17	16	30			
4	20	18	17	16	15	14	13	12	11	10	25		
3	15	13	12	11	10	9	8	7	6	5	4	20	
2	10	7	6	5	4	3	2	1	2	3	3	3	15

LOOSE AGGRESSIVE
LOOSE PASSIVE

TIGHT AGGRESSIVE
TIGHT PASSIVE

Από τους παραπάνω πίνακες, βλέποντας το πλήθος των αρχικών χεριών που απορρίπτονται, γίνεται σαφές πως οι στρατηγικές ΤΝ έχουν γίνει πιο ελαστικές σε σχέση με τις ανάλογες που συναντώνται στους ανθρώπινες παίκτες, εννοώντας πως τα χέρια τα οποία θα παίζονται στο Pre-Flop από τον παίκτη ΤΝ, θα είναι πολλά περισσότερα, ιδιαίτερα στις loose στρατηγικές.

Με τον ίδιο τρόπο εξετάζεται η ενέργεια του παίκτη ΤΝ στα επόμενα στάδια του παιχνιδιού ελέγχοντας κατά βάση το HandStrength. Για κάθε στρατηγική ΤΝ ορίζονται οι κανόνες isFold/8, isCheck/8, isCall/8, isBet/8, isRaise/8, με ορίσματα τις μεταβλητές Strategy, GameState, HandStrength, PotentialStrength, OpponentAction, MyChipStatus, OpponentChipStatus και ActionsLeft. Ο έλεγχος των κανόνων αυτών γίνεται μέσω του checkAction/9, καθορίζοντας την ενέργεια του παίκτη ΤΝ.

Η μέθοδος με την οποία ομαδοποιούνται οι αξίες του χεριού σε σύνολα, ονομάζεται bucketing [8][9]. Είναι η πιο σημαντική αφαιρετική μέθοδος για τον υπολογισμό ψευδο-βέλτιστων στρατηγικών. Το σύνολο των χεριών είναι χωρισμένο σε κλάσεις ή buckets, όπου σε κάθε μία, τα χέρια από τα οποία αποτελείται είναι παρόμοιας στρατηγικής αξίας. Το πρόγραμμα της εργασίας αυτής πραγματοποιεί ένα υποτυπώδες bucketing.

Στον Πίνακα 8 παρουσιάζονται μερικά χαρακτηριστικά των στρατηγικών ΤΝ.

Πίνακας 8 "Χαρακτηριστικά στρατηγικών ΤΝ"

	Loose Passive ΤΝ	Loose Aggressive ΤΝ	Tight Passive ΤΝ	Tight Aggressive ΤΝ
Πλήθος αρχικών χεριών που παίζονται	Πάρα πολλά	Πάρα πολλά	Λίγα	Λίγα
Αξία αρχικού χεριού που παίζεται	Αδύναμο	Αδύναμο	Δυνατό	Δυνατό
Αγαπημένη ενέργεια	Call/Fold	Bet/Raise	Check/Fold	Fold/Bet/Raise
Συχνότητα μπλόφας	Σπάνια	Συχνά	Σπάνια	Σπάνια

Ο παίκτης ΤΝ που δημιουργήθηκε, ο RezAI, έχει σχεδιαστεί να επιλέγει τυχαία μία από τις τέσσερις στρατηγικές μόλις αρχίζει ένα νέο παιχνίδι. Ακολουθεί την επιλεγμένη στρατηγική ελέγχοντας κάθε φορά αν έχει χάσει chips στο τέλος του κάθε γύρου, δηλαδή στο showdown ή σε περίπτωση που κάνει fold. Η μεταβλητή που αντιπροσωπεύει το πλήθος των χαμένων αυτών chips είναι η ChipsLost και έχει ιδιαίτερη σημασία για τον παίκτη ΤΝ.

Σχήμα 9 "Μηχανισμός αλλαγής στρατηγικής του RezAI"

Στο Σχήμα 9 παρουσιάζεται η διαδικασία αλλαγής στρατηγικής του RezAI. Για να γίνει αυτό πρέπει να τηρούνται δύο προϋποθέσεις:

- ✓ Τα χαμένα chips να φτάσουν σε μια συγκεκριμένη τιμή (π.χ. το 1/5 των αρχικών chips).
- ✓ Τα chips του RezAI να είναι λιγότερα ή ίσα από αυτά του αντιπάλου.

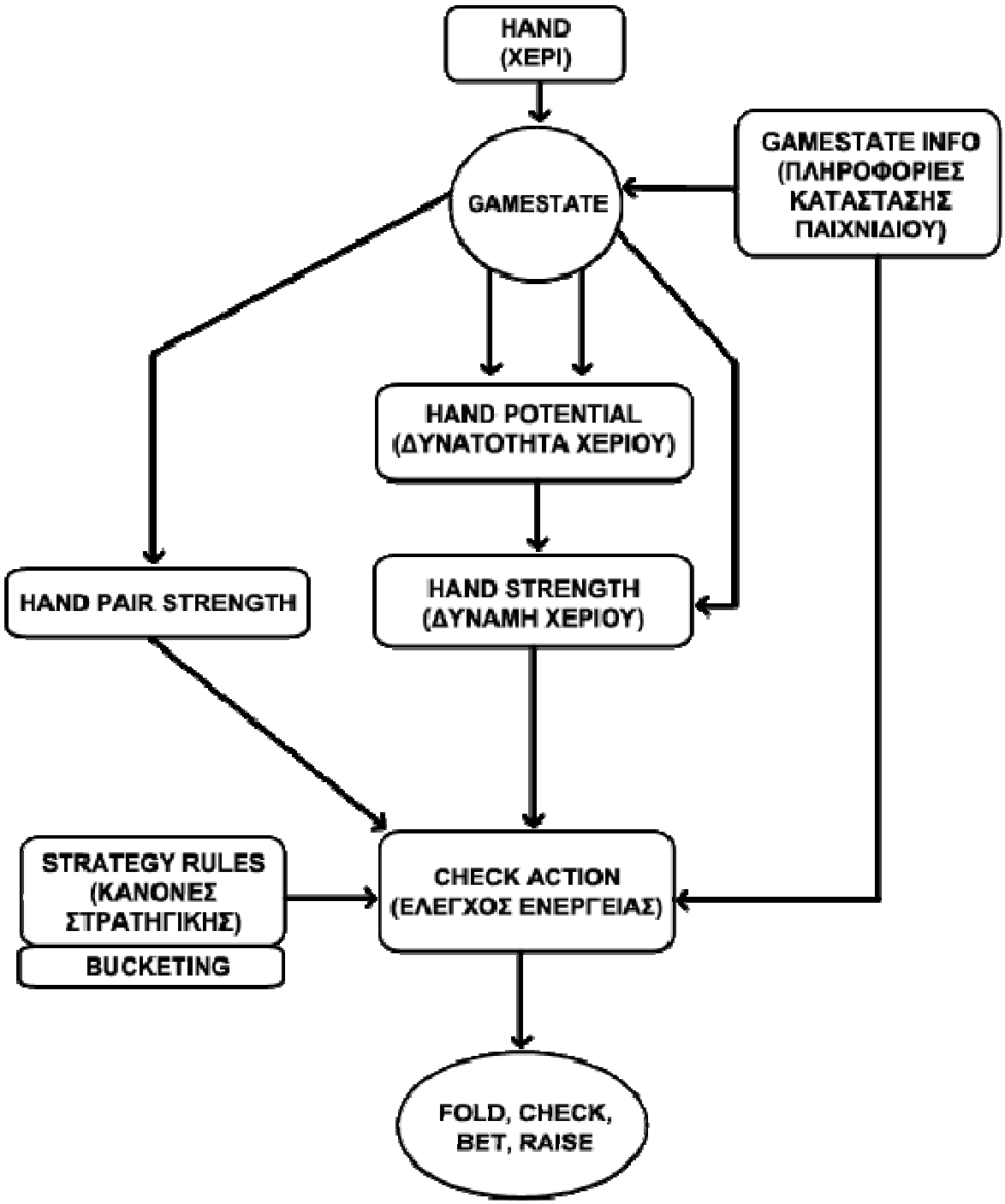
Εφόσον ισχύουν οι δύο αυτές προϋποθέσεις τότε η τιμή του ChipsLost μηδενίζεται και επιλέγεται τυχαία μια νέα στρατηγική ΤΝ από τις τέσσερις, χωρίς να σημαίνει αυτό πως δε θα είναι η ίδια με την προηγούμενη. Στο Σχήμα 10 βλέπουμε ένα μέρος του κώδικα για την αλλαγή στρατηγικής του RezAI.

```
changeStrategy(MyChips, OpponentChips, NewStrategy) :-
    MyChips =< OpponentChips,
    nb_getval(chipsLost, ChipsLost),
    ChipsLost >= 100,
    randomStrategy(NS),
    nb_setval(chipsLost, 0),
    NewStrategy is NS.

randomStrategy(X) :-
    random(1, 5, X).
```

Σχήμα 10 ”Κώδικας μηχανισμού αλλαγής στρατηγικής του RezAI”

Μια γενική εικόνα της αρχιτεκτονικής του RezAI φαίνεται στο Σχήμα 11.



Σχήμα 11 "Αρχιτεκτονική του RezAI"

3.3 ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΚΑΙ JPL

Το πρόγραμμα Prolog που αναπτύχθηκε στην εργασία αυτή αφορά ένα παιχνίδι πόκερ μεταξύ ενός ανθρώπινου παίκτη και του παίκτη TN (RezAI). Για τον λόγο αυτόν ήταν αναγκαίο να δημιουργηθεί ένα γραφικό περιβάλλον που θα επιτρέπει σε οποιονδήποτε χρήστη να μπορεί να παίξει ενάντια στον RezAI με σχετική ευκολία. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη του είναι η Java, η σύνδεση της οποίας με την Prolog γίνεται μέσω της βιβλιοθήκης JPL, που είναι ενσωματωμένη στην τελευταία έκδοση της Swi-Prolog (έκδοση 6.2).

Για την σύνδεση της Java με την Prolog εξετάστηκαν διάφορες λύσεις, όπως τα InterProlog, GNU Prolog for Java, JIProlog, καταλήγοντας στο JPL το οποίο καλύπτει απόλυτα τις ανάγκες της παρούσας εργασίας. Το JPL είναι ένα σύνολο από κλάσεις Java και C, παρέχοντας μια διεπαφή μεταξύ της Java και της Prolog. Χρησιμοποιεί το Java Native Interface (JNI) για να συνδεθεί στην μηχανή Prolog μέσω του Foreign Language Interface (FLI). Δεν είναι μια καθαρή υλοποίηση της Prolog σε Java, κάνει εκτεταμένη χρήση native υλοποιήσεων της Prolog στις υποστηριζόμενες πλατφόρμες. Η τρέχουσα έκδοση του JPL λειτουργεί μόνο με την Swi-Prolog. Είναι επίσης χωρισμένο σε δύο στρώματα, το χαμηλό επίπεδο διασύνδεσης με την Prolog FLI και μια υψηλού επιπέδου διεπαφή Java με τον προγραμματιστή, που δεν ασχολείται με τις λεπτομέρειες της Prolog FLI.

Η χρήση του προϋποθέτει σαφώς να είναι εγκατεστημένη η Swi-Prolog, αλλά και να έχει οριστεί η διαδρομή του φακέλου εγκατάστασης <PROLOG>\bin στην μεταβλητή συστήματος PATH.

Το πρόγραμμα αποτελείται από τρεις κλάσεις Java:

MainFrame

PokerGame

PrologQueries

και ένα αρχείο Prolog, το rules.pl.

Στην MainFrame δημιουργούνται όλα τα στοιχεία του γραφικού περιβάλλοντος και οι λειτουργίες τους. Στην κλάση PrologQueries ορίζονται όλα τα queries που θα γίνουν στο αρχείο rules.pl σε μορφή String, ενώ η κλάση PokerGame αντιπροσωπεύει το παιχνίδι του πόκερ με όλα τα χαρακτηριστικά και τις ιδιότητες τους. Επίσης, από την PokerGame εκτελούνται όλα τα queries προς το εξωτερικό αρχείο rules. Οι Java κλάσεις ασχολούνται μόνο με τις λειτουργίες που είναι απαραίτητες για την σωστή ροή του παιχνιδιού και την ορθή σύνδεση των δύο γλωσσών, αφήνοντας στο Prolog αρχείο rules την ευθύνη για την υλοποίηση όλων των λειτουργιών που αφορούν την τεχνητή νοημοσύνη.

Για την δημιουργία του γραφικού περιβάλλοντος έγινε εκτεταμένη χρήση της βιβλιοθήκης java.Swing, ενώ για τον ευκολότερο σχεδιασμό του χρησιμοποιήθηκε

το Eclipse IDE με ένα ειδικευμένο στην δημιουργία παραθύρων plugin, το WindowsBuilder.

Παρακάτω βλέπουμε ένα παράδειγμα του τρόπου που γίνεται το consult ενός αρχείου Prolog μέσω του JPL.

```
import jpl.Query;

Query consultQuery = new Query("consult('rules.pl')");
consultQuery.hasSolution();
```

Σχήμα 12 "Consult αρχείου μέσω JPL"

Στο Σχήμα 13 φαίνεται ένα στιγμιότυπου της εφαρμογής σε μια τυχαία κατάσταση του παιχνιδιού.

Σχήμα 13 "Στιγμιότυπο εφαρμογής"

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό έγινε μια μικρή αναφορά στον τρόπο αναπαράστασης των φύλλων μέσα στην εφαρμογή και στους λόγους που οδήγησαν σε αυτό. Επίσης, παρουσιάστηκαν εκτενώς οι βασικότεροι παράγοντες που διαμορφώνουν την συμπεριφορά της κάθε στρατηγικής ΤΝ, δηλαδή το HandStrength και το HandPotential, καθώς και η αρχιτεκτονική της κάθε μιας. Ακόμα είδαμε τον τρόπο που συνδέθηκε το γραφικό περιβάλλον μέσω του JPL με την Prolog. Στο επόμενο κεφάλαιο θα εξεταστεί η αποτελεσματικότητα του RezAI και των στρατηγικών του.

ΚΕΦΑΛΑΙΟ 4 - ΠΕΙΡΑΜΑΤΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

ΕΙΣΑΓΩΓΗ

Στο προηγούμενο κεφάλαιο έγινε η περιγραφή των βημάτων ανάπτυξης του παίκτη TN, τα χαρακτηριστικά του και οι στρατηγικές που ακολουθεί. Στο κεφάλαιο αυτό θα παρουσιαστούν οι δοκιμές που έγιναν, θα δοθούν διαγράμματα από τα πειράματα που εκτελέστηκαν και θα γίνει μια προσπάθεια επεξήγησης των μετρήσεων.

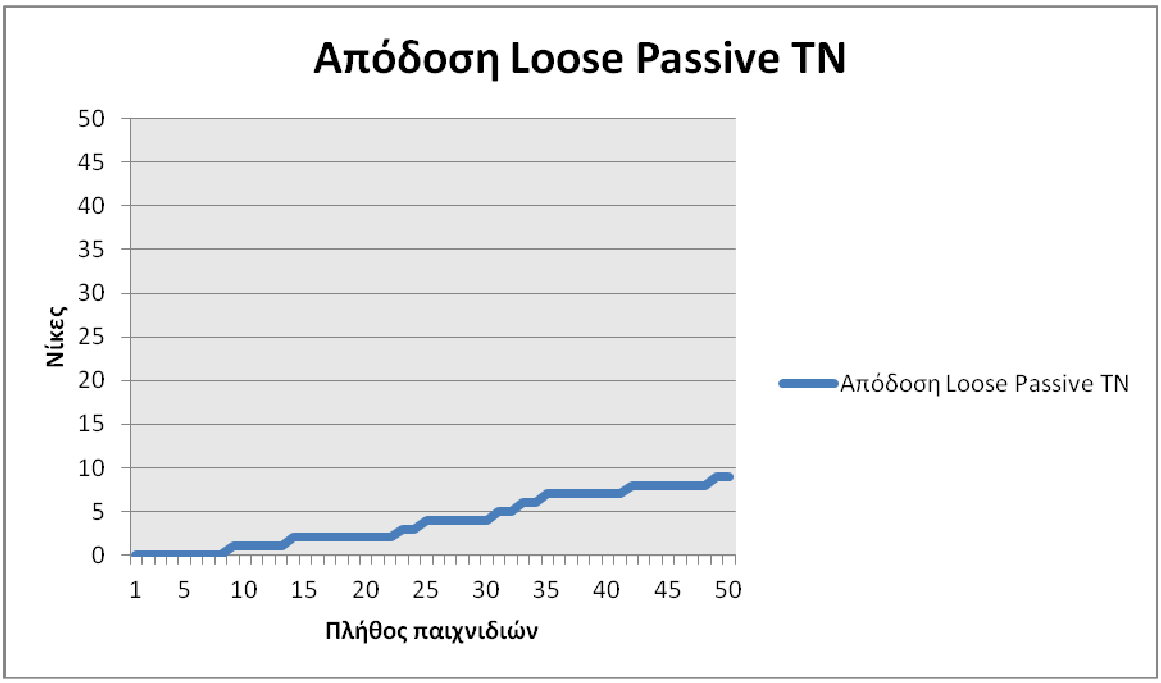
4.1 ΑΠΟΔΟΣΗ ΣΤΡΑΤΗΓΙΚΩΝ TN

Η κάθε στρατηγική TN δοκιμάστηκε ενάντια σε ένα σύνολο τεσσάρων ανθρώπινων παικτών μεσαίου και χαμηλού επιπέδου. Το μέγεθος του πειράματος δε μας επιτρέπει να εξάγουμε ασφαλή συμπεράσματα, καθώς πρόκειται μόνο για πενήντα παιχνίδια που καταγράφηκαν για κάθε μία στρατηγική TN, όμως είναι μια βάση για να μπορέσουμε να αναδείξουμε τις διαφορές τους και την απόδοση τους.

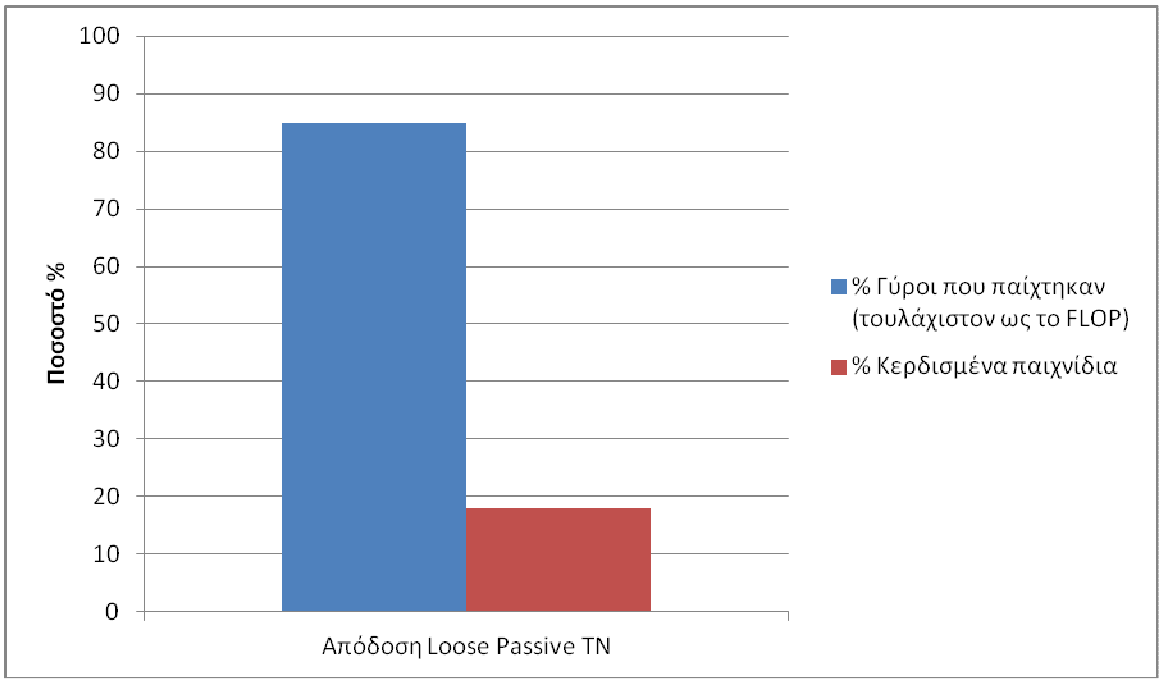
Προτιμήθηκε να παρουσιαστούν πρώτα τα διαγράμματα απόδοσης της κάθε στρατηγικής TN, ώστε να έχουμε μια καλή εικόνα για όλες, προτού να εξετασθεί η απόδοση του RezAI.

Για κάθε στρατηγική θα δούμε δύο διαγράμματα. Το πρώτο θα απεικονίζει τις νίκες που υπήρξαν ακολουθώντας αυτήν την στρατηγική σε συνάρτηση με το πλήθος των παιχνιδιών που παίχθηκαν. Να ξεκαθαρίσουμε πως με την λέξη “παιχνίδι” εννοούμε το σύνολο των γύρων που έπαιξε, μέχρι ένας από τους δύο αντιπάλους να χάσει όλα τα chips του. Το δεύτερο διάγραμμα παρουσιάζει το ποσοστό των γύρων που επέλεξε η στρατηγική να παίξει κατά τη διάρκεια των πενήντα παιχνιδιών, αλλά και το ποσοστό των παιχνιδιών που κέρδισε. Ένας γύρος παιχνιδιού θεωρείται ότι έχει παιχθεί από την στρατηγική, εάν η κατάσταση παιχνιδιού έχει φθάσει τουλάχιστον μέχρι το flop, δηλαδή υπάρχουν τουλάχιστον τρία φύλλα στο τραπέζι, ή στην περίπτωση που η στρατηγική TN ανάγκασε τον αντίπαλο να κάνει fold προτού το μοιραστεί το flop.

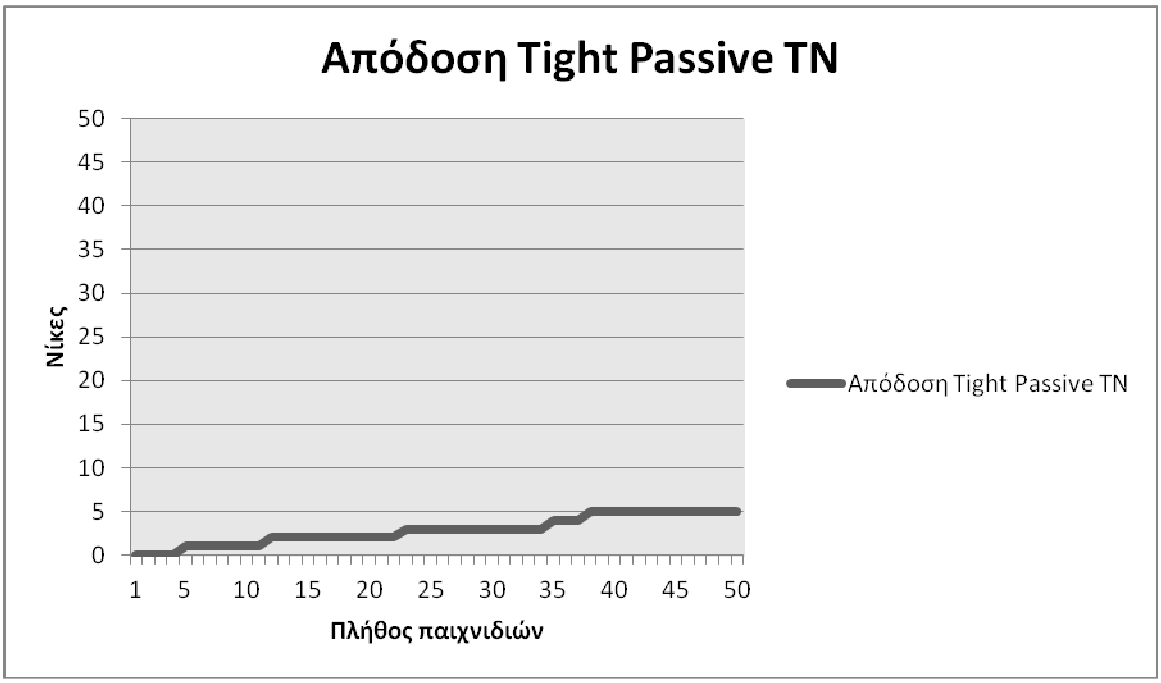
Με τα διαγράμματα αυτά, θα μπορέσουμε να βγάλουμε κάποια βασικά συμπεράσματα, όχι μόνο για το πόσο καλά τα πήγαν οι στρατηγικές TN, αλλά και την συμπεριφορά που είχαν στο σύνολο των πενήντα αυτών παιχνιδιών. Με αυτόν τον τρόπο θα μπορέσουμε να εξετάσουμε αν όντως η κάθε στρατηγική TN λειτουργεί με τον τρόπο για τον οποίο δημιουργήθηκε. Πιο συγκεκριμένα, θέλουμε να διαπιστώσουμε αν οι Loose στρατηγικές TN επαληθεύσουν την συνήθεια τους να παίζουν πολλά αρχικά χέρια και αντιθέτως οι Tight να παίζουν λίγα.



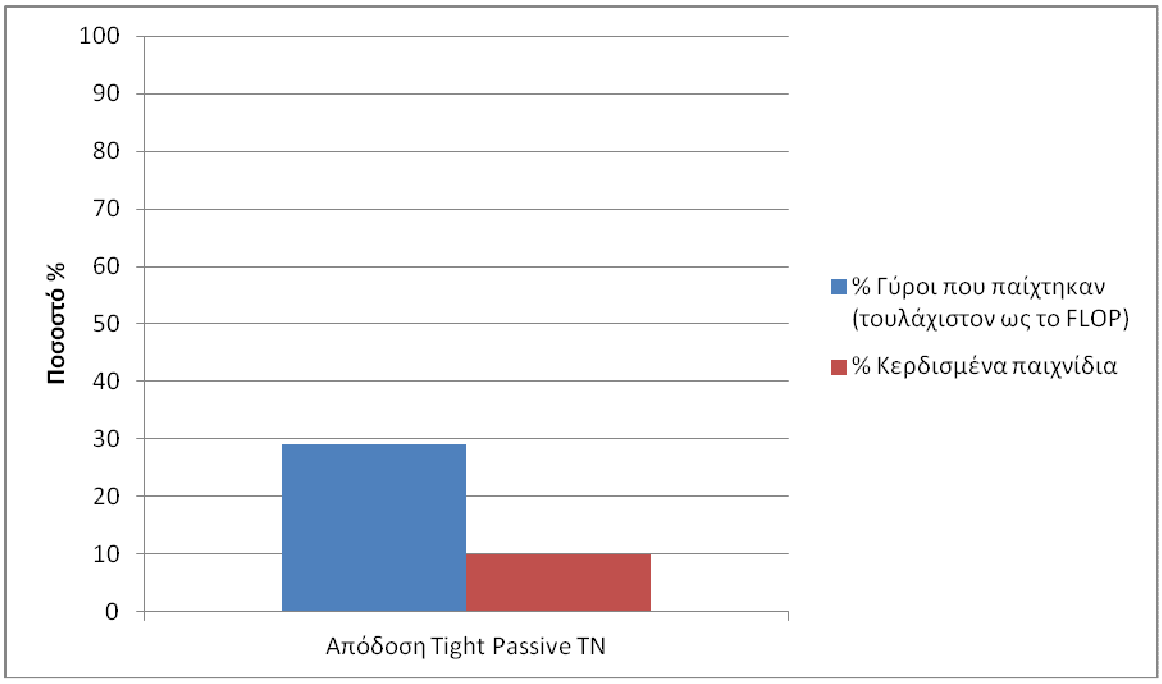
Σχήμα 14 "Απόδοση Loose Passive TN"



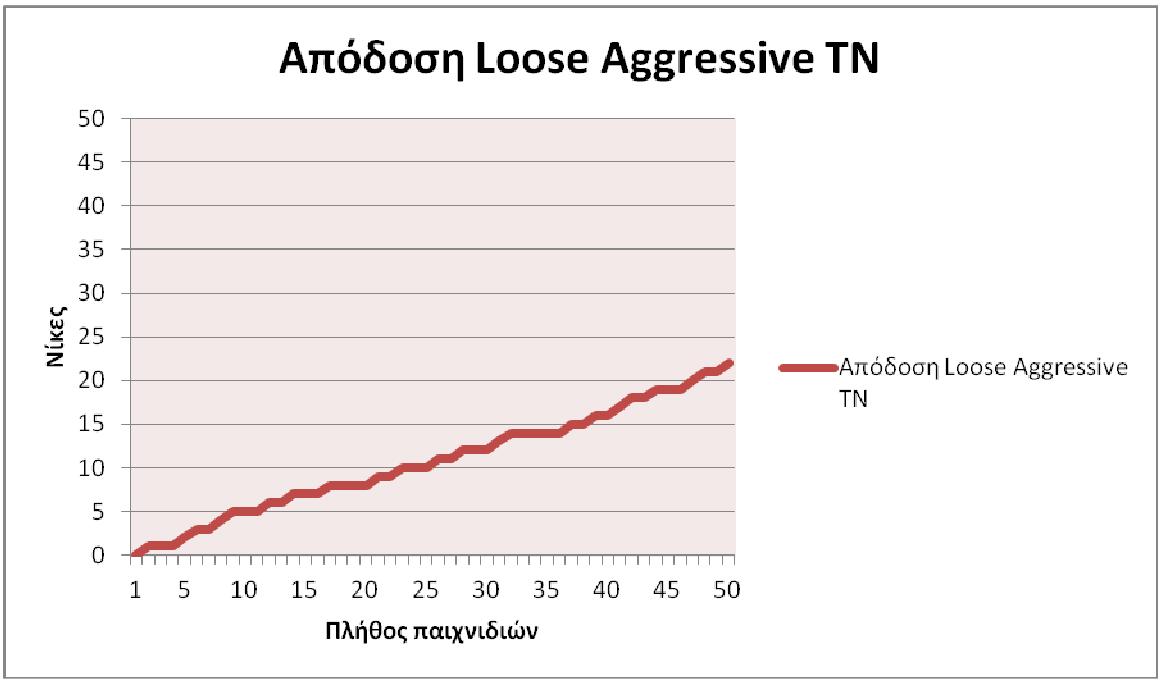
Σχήμα 15 "Ποσοστιαία απόδοση και γύροι που παίχτηκαν στην Loose Passive TN"



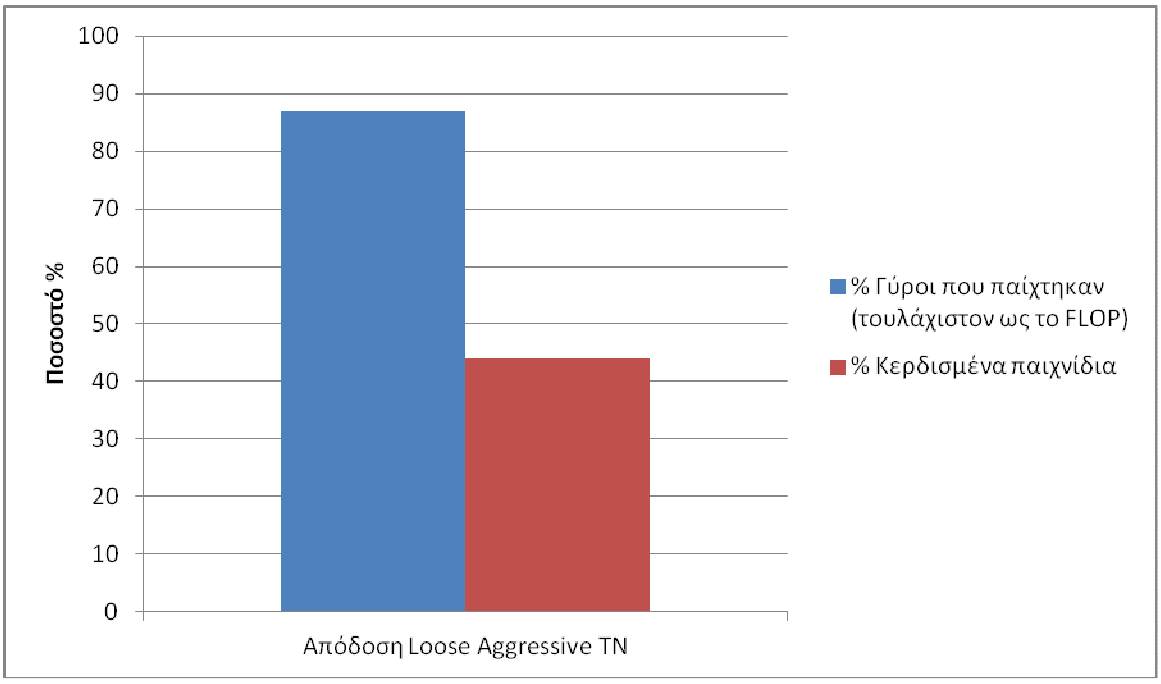
Σχήμα 16 "Απόδοση Tight Passive TN"



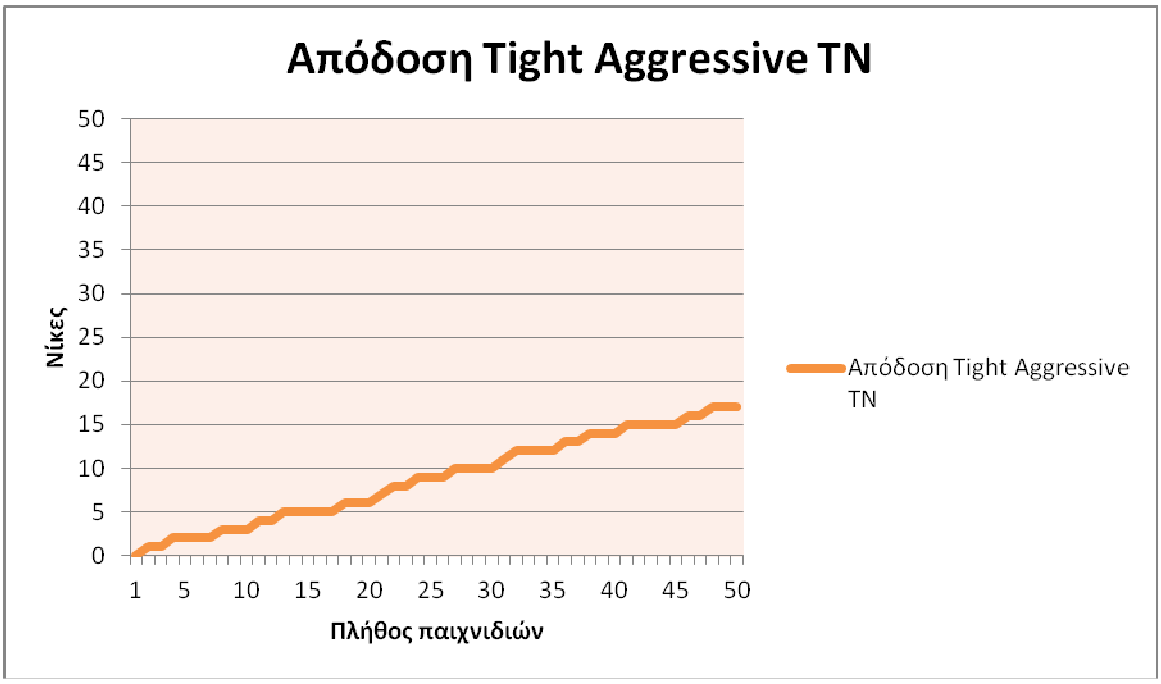
Σχήμα 17 "Ποσοστιαία απόδοση και γύροι που παίχτηκαν στην Tight Passive TN"



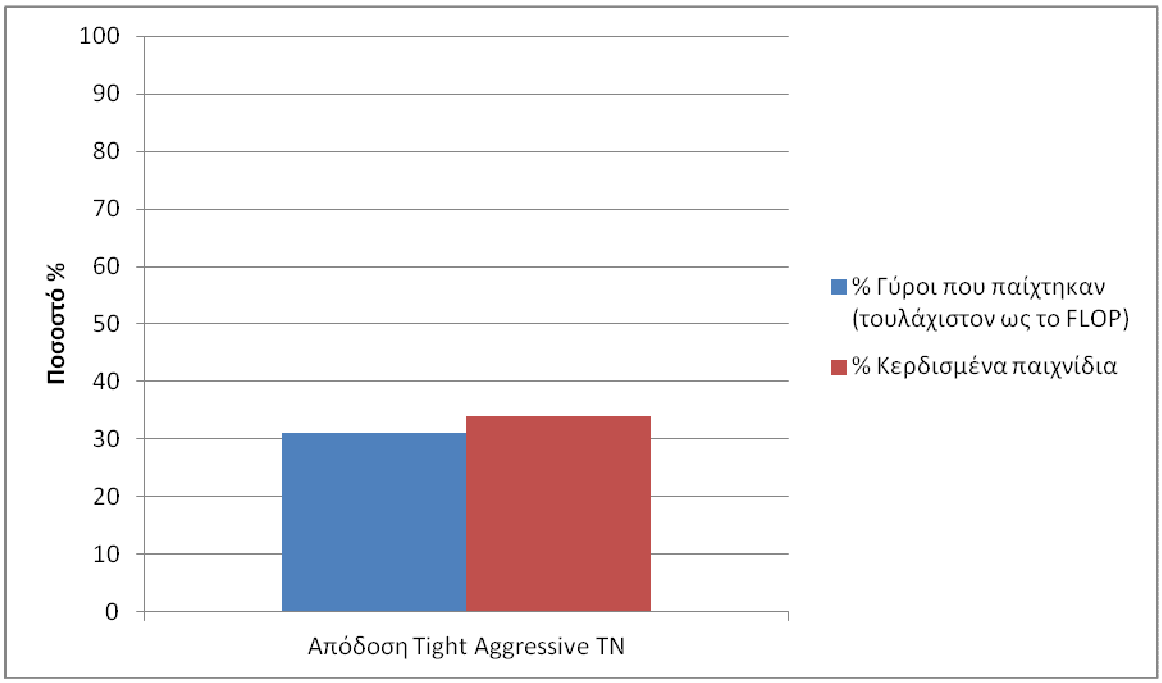
Σχήμα 18 "Απόδοση Loose Aggressive TN"



Σχήμα 19 "Ποσοστιαία απόδοση και γύροι που παίχτηκαν στην Loose Aggressive TN"



Σχήμα 20 "Απόδοση Tight Aggressive TN"

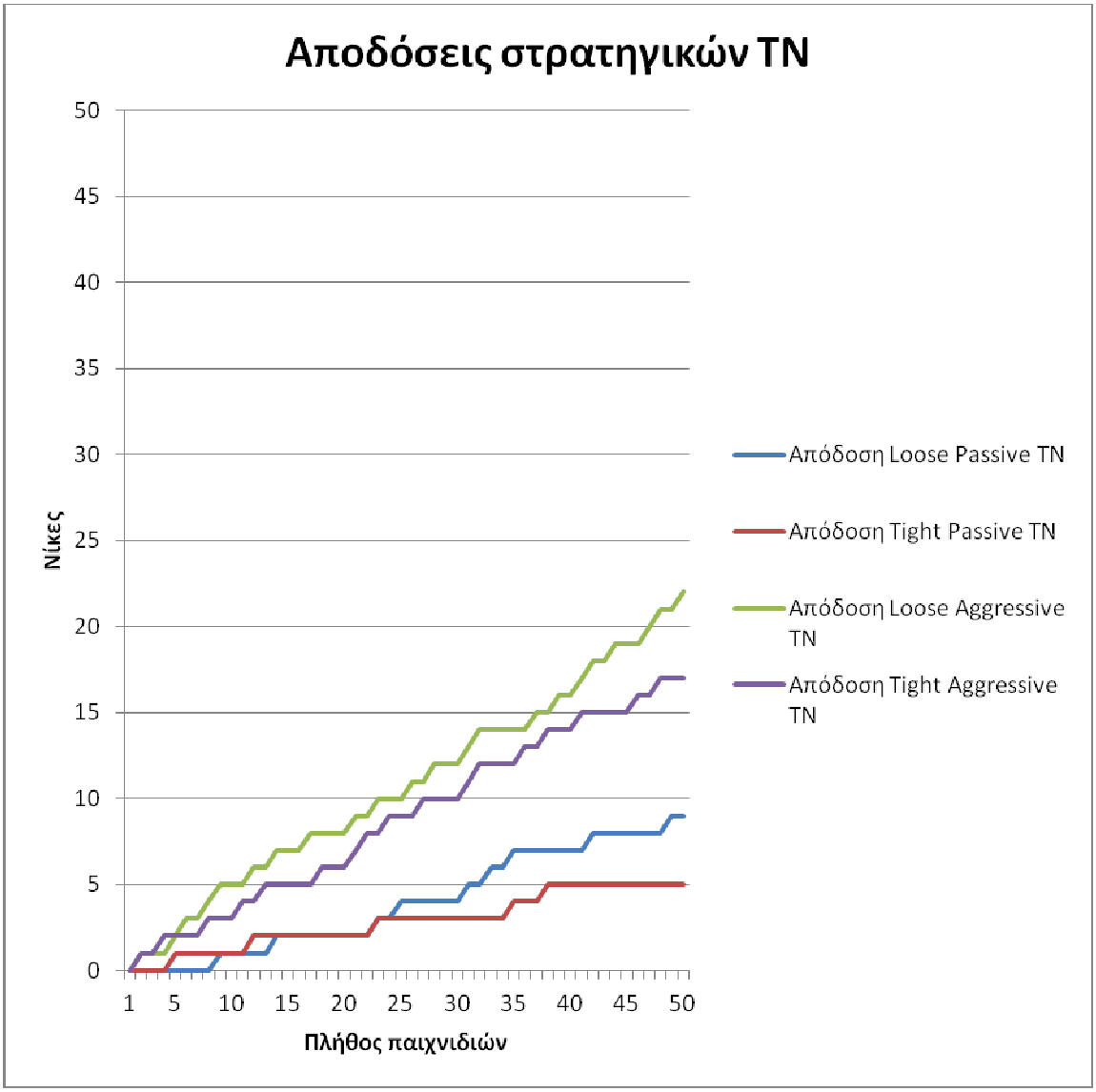


Σχήμα 21 "Ποσοστιαία απόδοση και γύροι που παίχτηκαν στην Tight Aggressive TN"

Στα παραπάνω σχήματα βλέπουμε την τραγικά χαμηλή απόδοση των Passive στρατηγικών TN ενάντια σε ανθρώπινους παίκτες χαμηλού και μεσαίου επιπέδου. Παρατηρούμε πως η Tight Passive TN έχει την μικρότερη απόδοση

(10% νίκη) λόγω του ότι η στρατηγική γίνεται εύκολα αντιληπτή από τον αντίπαλο, παίζοντας μόνο αρχικά χέρια μεγάλης αξίας. Σε λίγο καλύτερα επίπεδα κινείται η Loose Passive TN (18% νίκη).

Αντίθετα, οι στρατηγικές Aggressive έχουνε πολύ καλύτερη απόδοση. Συγκεκριμένα, η Loose Aggressive TN αποδίδει πολύ καλά (44% νίκη), παίζοντας ταυτόχρονα και ένα πολύ μεγάλο ποσοστό των αρχικών χεριών που μοιράζονται (87%). Η Tight Aggressive TN έχει και αυτή μια σχετικά ικανοποιητική απόδοση (34% νίκη), παίζοντας όμως ένα πολύ μικρότερο ποσοστό των αρχικών χεριών (31%). Συνολικά οι αποδόσεις των τεσσάρων στρατηγικών TN φαίνονται στο Σχήμα 22.



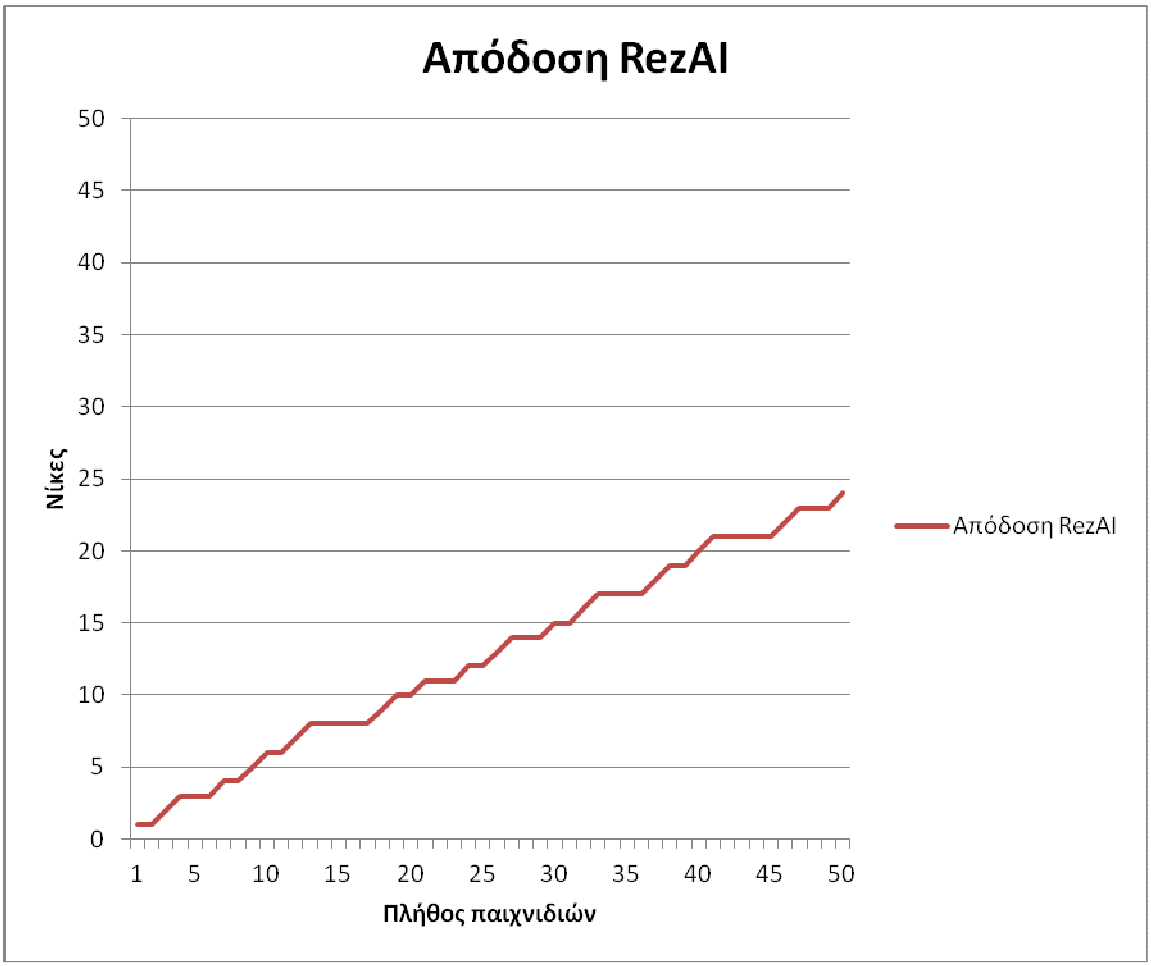
Σχήμα 22 "Αποδόσεις στρατηγικών TN"

4.2 ΑΠΟΔΟΣΗ ΤΟΥ ΠΑΙΚΤΗ ΤΝ (RezAI)

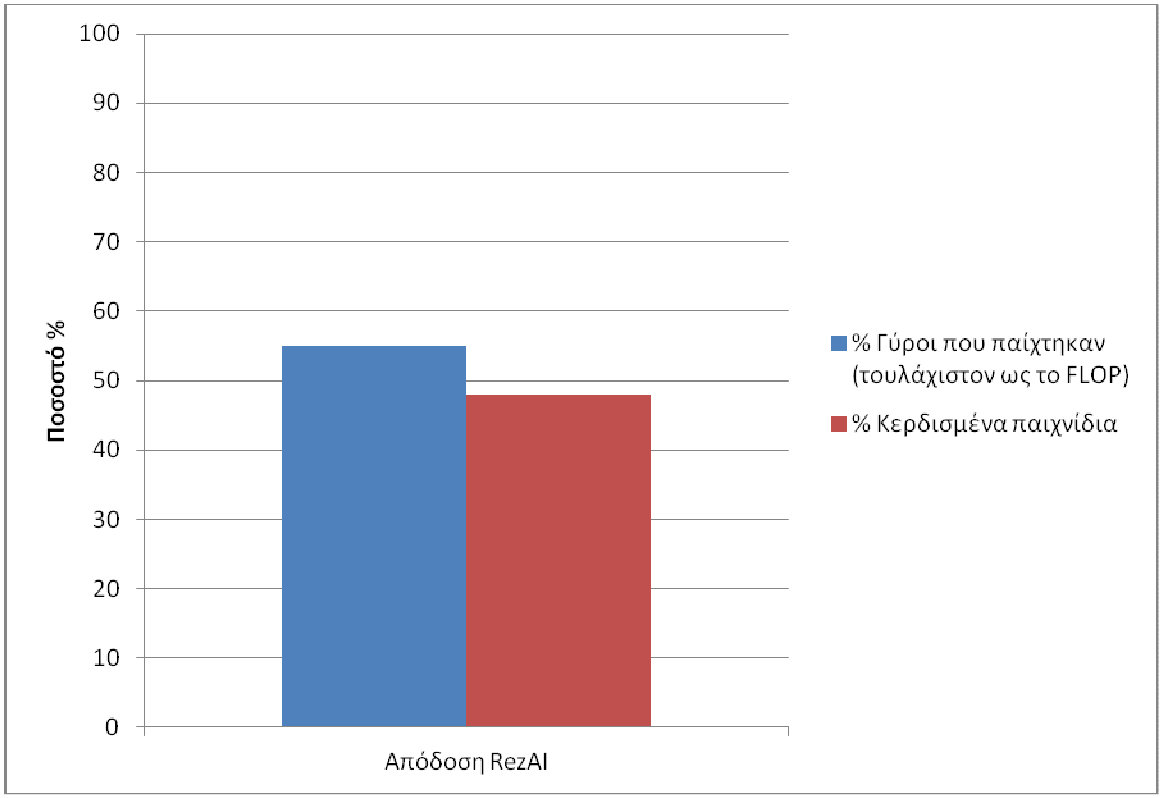
Στο Κεφάλαιο 3, είδαμε τον τρόπο με τον οποίο ο RezAI επιλέγει την στρατηγική ΤΝ που θα ακολουθήσει και υπό ποιές προϋποθέσεις την αλλάζει. Στο σημείο αυτό θα εξετάσουμε την απόδοση που είχε ο RezAI ενάντια σε ανθρώπινους αντιπάλους, όπως αυτή παρουσιάζεται στα σχήματα αυτού του υποκεφαλαίου.

Είναι ένα πολύ σημαντικό μέρος της εργασίας, διότι θα διαπιστώσουμε αν το πρόγραμμα πόκερ που αναπτύχθηκε είχε τα αναμενόμενα αποτελέσματα, δηλαδή αν τα επίπεδα της απόδοσης του είναι μέσα στα επιθυμητά όρια.

Λόγω του ότι ο RezAI είναι στην ουσία μια μείξη των τεσσάρων στρατηγικών ΤΝ, πρέπει να εξετάσουμε και κάποια επιπλέον διαγράμματα σε σχέση με πριν, για να μπορέσουμε να εξάγουμε πιο σαφή συμπεράσματα, όσον αφορά την συμπεριφορά που είχε.



Σχήμα 23 "Απόδοση RezAI"

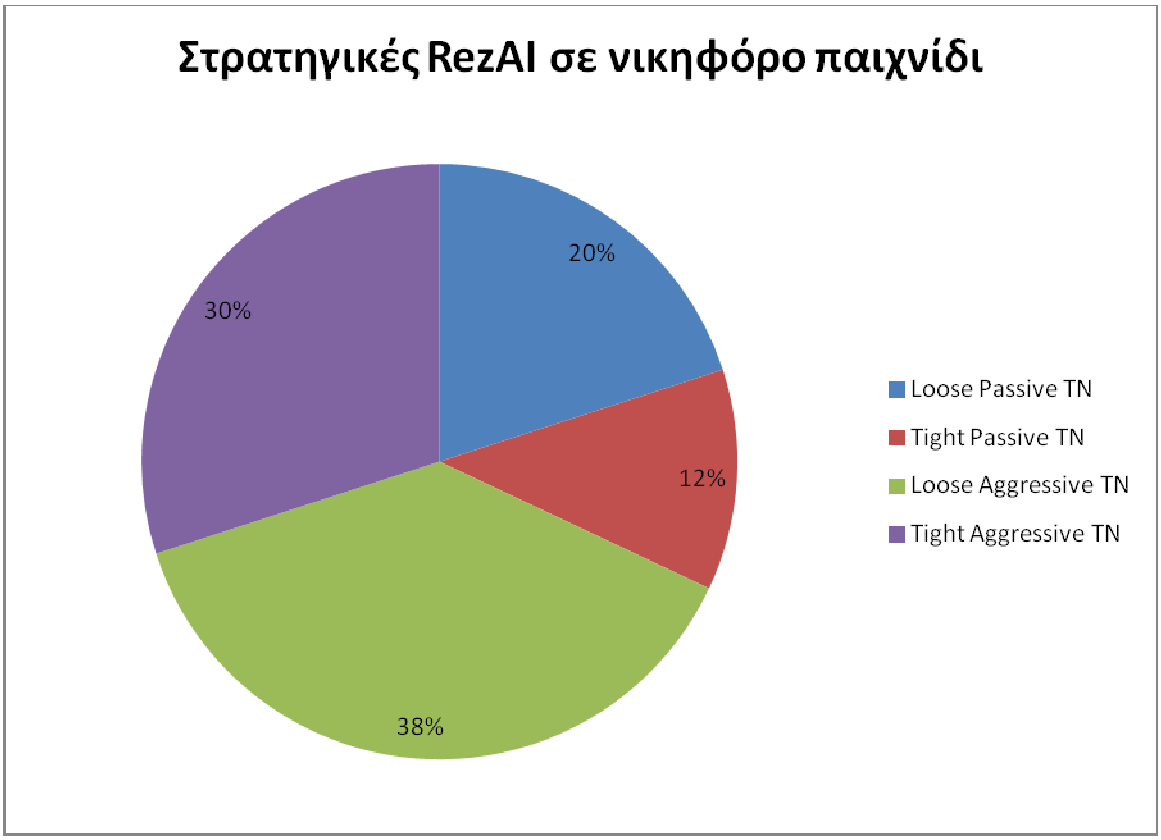


Σχήμα 24 "Ποσοστιαία απόδοση και γύροι που παίχτηκαν από τον RezAI"

Παρατηρούμε πως η απόδοση του RezAI (48% νίκη) είναι σχεδόν ίση με αυτήν της στρατηγικής Loose Aggressive TN. Αυτό δεν σημαίνει όμως πως έχουν το ίδιο στυλ παιξίματος, καθώς ο RezAI διασφαλίζει ως ένα βαθμό την μη-προβλεψιμότητα στον τρόπου παιξίματος του, ενώ η χρήση μιας μεμονωμένης στρατηγικής θα τον πρόδιδε σε βάθος χρόνου. Αυτή η πλούσια σε διαφορετικές συμπεριφορές στρατηγική του RezAI, ήταν και ένας από τους βασικούς στόχους από την αρχή της ανάπτυξης αυτής της εργασίας.

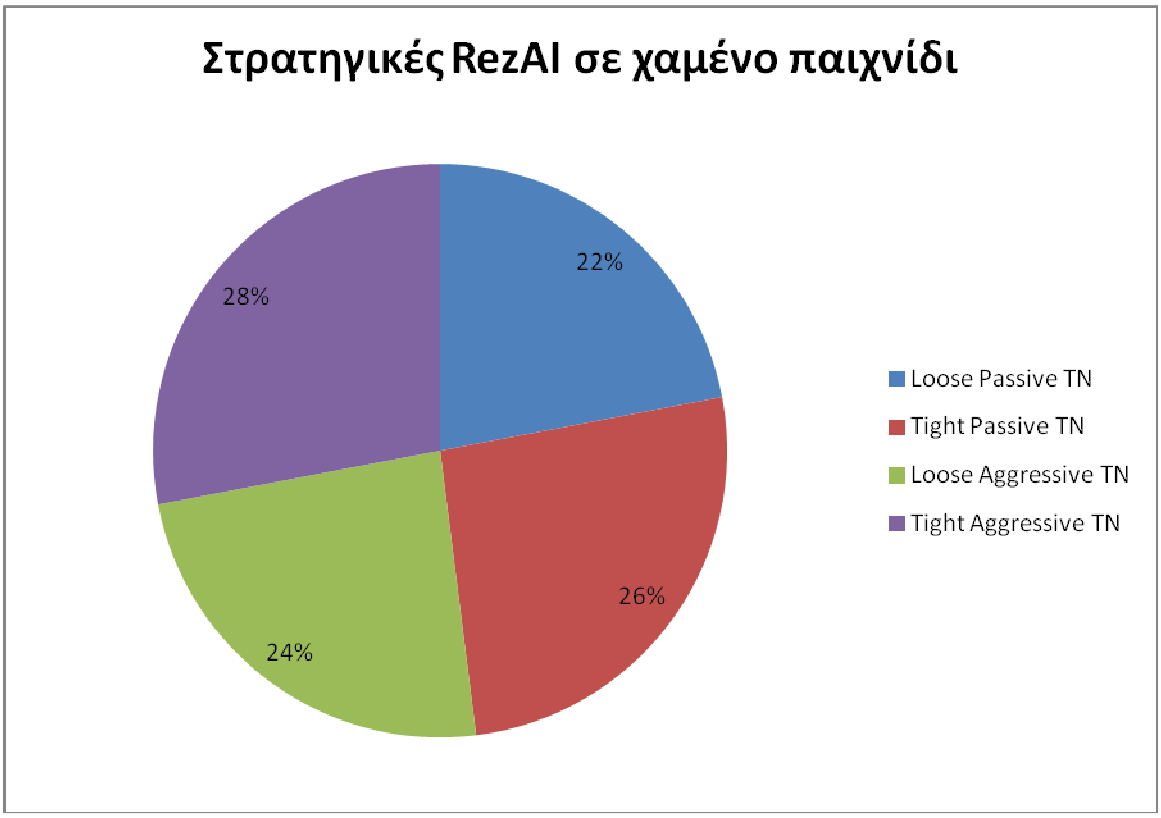
Επίσης βλέπουμε πως το ποσοστό των αρχικών χεριών που παίχτηκαν (55%), είναι κοντά στο μέσο όρο του συνόλου των τεσσάρων στρατηγικών TN, το οποίο είναι αναμενόμενο, αφού ο RezAI κάνει εναλλαγές των στρατηγικών αυτών κατά την διάρκεια του παιχνιδιού.

Στα επόμενα δύο σχήματα παρουσιάζεται το ποσοστό των γύρων που έπαιξε ο RezAI με την κάθε στρατηγική σε δύο επιλεγμένα παιχνίδια. Στο Σχήμα 25, ο RezAI νίκησε, ενώ στο Σχήμα 26 έχασε.



Σχήμα 25 "Ποσοστά χρήσης της κάθε στρατηγικής από τον RezAI σε ένα νικηφόρο παιχνίδι"

Στο νικηφόρο για τον RezAI παιχνίδι του σχήματος 25, παρατηρούμε πως οι Aggressive στρατηγικές χρησιμοποιήθηκαν στο 68% των γύρων που παίχθηκαν. Από το γεγονός αυτό, επαληθεύεται η μεγαλύτερη αποτελεσματικότητα που έχουν οι Aggressive στρατηγικές έναντι των Loose. Το μεγάλο ποσοστό χρήσης τους μπορεί να οφείλεται σε δύο αίτια, είτε ο RezAI κατάφερε να κερδίσει πολλούς γύρους του παιχνιδιού, όσο ακολουθούσε τις Aggressive στρατηγικές, με σχετικά μικρές απώλειες κι έτσι δεν αναγκάστηκε να αλλάξει την στρατηγική του, είτε ο μηχανισμός τυχαίας επιλογής στρατηγικής του RezAI κατέληγε συχνότερα στις Aggressive. Να σημειώσουμε πως το ποσοστό χρήσης της Tight Passive TN, το οποίο είναι 12%, θεωρείται ιδανικό, καθώς πρόκειται για μια στρατηγική όχι ιδιαίτερα κερδοφόρα και η ύπαρξη της βοηθάει περισσότερο στην διαμόρφωση της απρόβλεπτης συμπεριφοράς του RezAI. Αν συμβουλευτούμε τα σχήματα που είδαμε στο υποκεφάλαιο 4.1, μπορούμε να θεωρήσουμε πως όσο αυξάνεται το ποσοστό χρήσης της, τόσο θα μειώνεται και η πιθανότητα νίκης για τον RezAI.

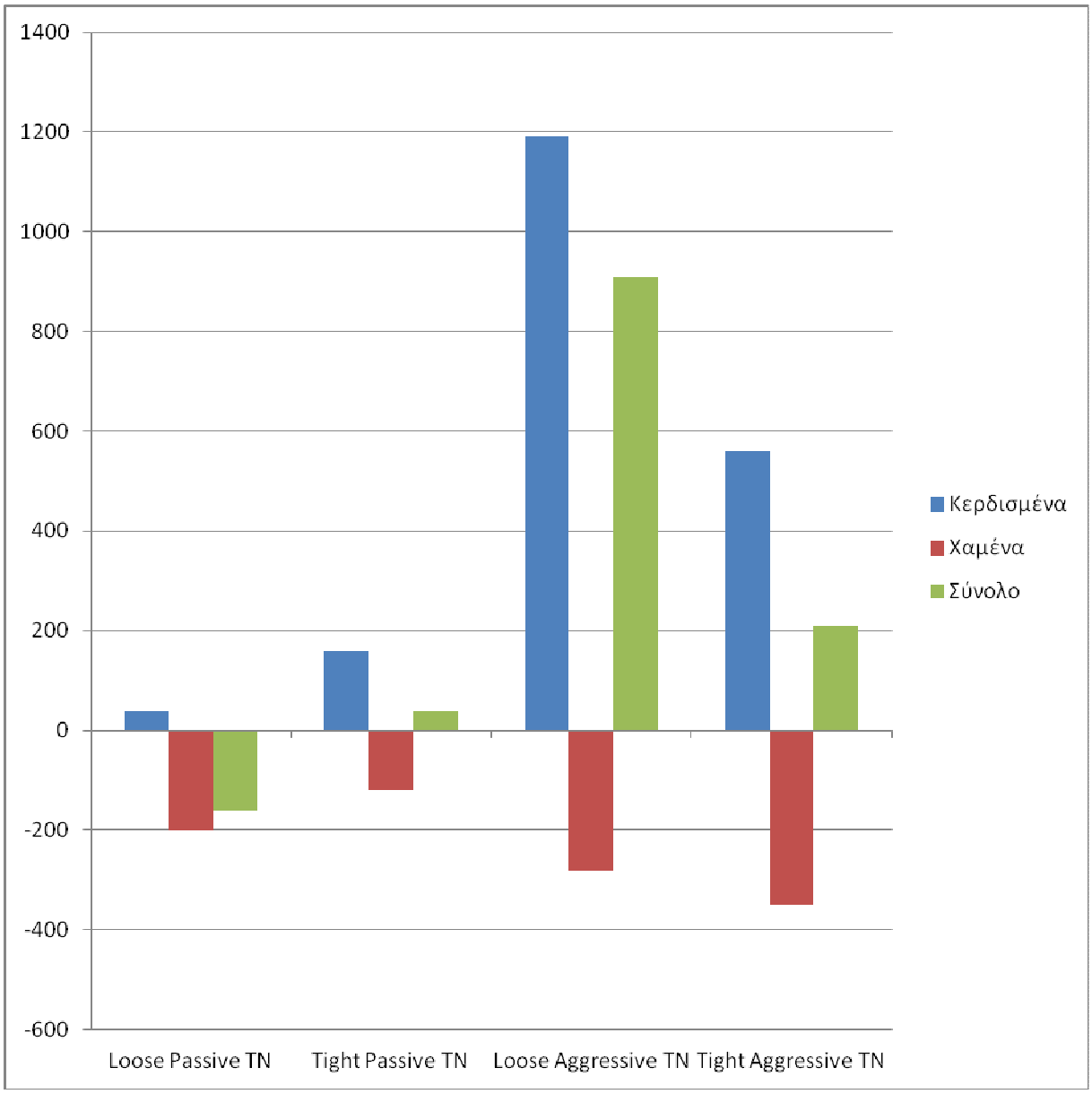


Σχήμα 26 "Ποσοστά χρήσης της κάθε στρατηγικής από τον RezAI σε ένα χαμένο παιχνίδι"

Στο Σχήμα 26 βλέπουμε το ποσοστό των στρατηγικών που χρησιμοποίησε σε κάθε γύρο ο RezAI, σε ένα παιχνίδι το οποίο τελικά έχασε, παρατηρούμε πως και για τις τέσσερις στρατηγικές τα ποσοστά είναι σχεδόν ίσα. Μια εξήγηση της κατάστασης αυτής, θα μπορούσε να είναι η περίπτωση κατά την οποία ο RezAI χάνει τον έλεγχο της παρτίδας από την αρχή, μένοντας με λιγότερα chips από τον αντίπαλο και αναγκάζεται να αλλάζει πολύ συχνά στρατηγική, μη μπορώντας να ανακάμψει, χάνοντας τελικά το παιχνίδι.

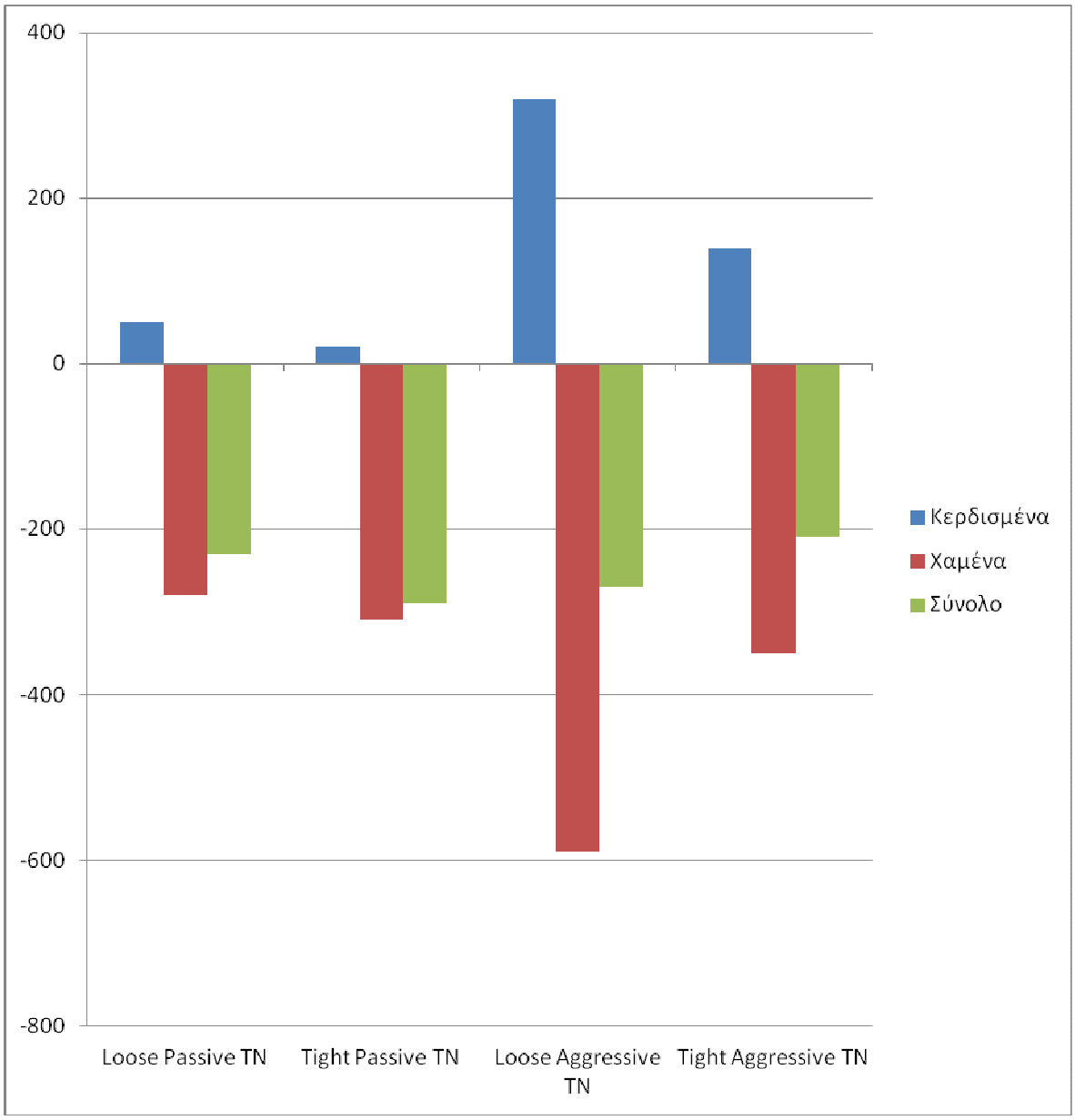
Τα δύο αυτά παιχνίδια που είδαμε στα σχήματα επιλέχθηκαν ενδεικτικά, ώστε να γίνει μια βασική ανάλυση της συμπεριφοράς του RezAI και πως αυτή επιδρά στην απόδοση του, χωρίς να σημαίνει ότι σε ένα άλλο παιχνίδι, τα ποσοστά χρήσης των στρατηγικών δεν μπορούν να είναι κατά μεγάλο βαθμό διαφορετικά.

Η ανάλυση των ποσοστών χρήσης των στρατηγικών λαμβάνοντας υπόψιν μόνο το τελικό αποτέλεσμα (νίκη ή ήττα), μπορεί κάποιες φορές να οδηγήσει σε λάθος συμπεράσματα, αφού δε γνωρίζουμε το κέρδος που είχε ο παίκτης TN με την κάθε στρατηγική. Έτσι, μια ακόμα καλύτερη προσέγγιση για την κατανόηση της απόδοσης του RezAI, θα ήταν η εξέταση των chips (POT) που κέρδισε με την κάθε στρατηγική στο σύνολο του παιχνιδιού, όπως φαίνεται παρακάτω.



Σχήμα 27 "Chips που κέρδισε ή έχασε η κάθε στρατηγική του RezAI σε ένα νικηφόρο παιχνίδι"

Το Σχήμα 27 μας δίνει μια λεπτομερή εικόνα για την απόδοση των στρατηγικών TN που χρησιμοποίησε ο RezAI σε ένα νικηφόρο για αυτόν παιχνίδι. Παρατηρούμε πως οι Aggressive στρατηγικές έχουν το μεγαλύτερο κέρδος, αλλά επίσης και περισσότερα χαμένα chips σε σύγκριση με τις αντίστοιχες Passive. Στο Σχήμα 28 βλέπουμε το αντίστοιχο διάγραμμα για ένα χαμένο παιχνίδι για τον RezAI.



Σχήμα 28 "Chips που κέρδισε ή έχασε η κάθε στρατηγική του RezAI σε ένα χαμένο παιχνίδι"

Παρατηρούμε πως και με τις τέσσερις στρατηγικές η διαφορά των chips που κέρδισε μείον αυτά που έχασε (Σύνολο) είναι περίπου η ίδια, γεγονός το οποίο μας δείχνει πως υπήρξαν συνεχείς εναλλαγές στρατηγικών χωρίς να καταφέρει να έχει καλή απόδοση, συμπεριλαμβανομένης και της Loose Aggressive TN με την οποία είχε το μεγαλύτερο κέρδος αλλά έχασε και με αυτήν τα περισσότερα chips.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό εξετάσαμε τις μετρήσεις που έγιναν για τις στρατηγικές TN και τον RezAI, μέσα από τα διαγράμματα που παρουσιάστηκαν. Βγάλαμε αρκετά συμπεράσματα όσον αφορά τη συμπεριφορά και την απόδοση τους. Επίσης, αναλύθηκαν ενδεικτικά συγκεκριμένα παιχνίδια και έγινε μια προσπάθεια εξήγησης της συμπεριφοράς του RezAI σε αυτά. Καταλήξαμε πως η απόδοση του RezAI βρίσκεται σε ικανοποιητικά επίπεδα ενάντια σε ανθρώπινους παίκτες χαμηλού και μεσαίου επιπέδου, έχοντας σε γενικές γραμμές μια καλή λειτουργία, με αρκετές εναλλαγές στρατηγικής σε κάθε παιχνίδι που διασφαλίζει την απρόβλεπτη συμπεριφορά του. Το ίδιο ικανοποιητική, στα πλαίσια αυτής της εργασίας, είναι η ικανότητα του RezAI να αξιολογήσει την παρτίδα, όπου γίνεται αντιληπτή από το πλήθος των αρχικών χεριών που έπαιξε αλλά και από το σύνολο των chips που κερδίζει σε ένα παιχνίδι.

Στην παρακάτω ενότητα παρουσιάζονται κάποια γενικά συμπεράσματα για την εργασία αυτή αλλά και κάποιες προτάσεις για μελλοντικές επεκτάσεις.

ΚΕΦΑΛΑΙΟ 5 - ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Ο RezAI είναι ένας μέτριος παίκτης πόκερ τεχνητής νοημοσύνης. Παίζει καλύτερα από αδύναμους ανθρώπινους παίκτες και πιθανότατα θα μπορούσε να τους νικήσει, ενώ ενάντια σε έμπειρους δυνατούς παίκτες έχει ελάχιστες πιθανότητες επιτυχίας. Αν και ο RezAI είναι φτιαγμένος να παίζει μόνο ενάντια σε ανθρώπους, δεν θα ήταν υπερβολή να πούμε πως δεν θα είχε καμία ελπίδα αν αγωνιζόταν κόντρα στα bots του CPRG (Computer Poker Research Group), όπως το Polaris και το Hyperborean. Χρησιμοποιεί ένα στατικό σύνολο κανόνων που με την πάροδο του χρόνου, οι μπλόφες του και οι στρατηγικές εξαπάτησης, δεν θα είναι αρκετές για να αποτρέψουν ένα CPRG bot από το να μοντελοποιήσει το παίξιμο του.

Ο RezAI δεν έχει κάποιο σύστημα μοντελοποίησης του αντιπάλου και είναι το μεγαλύτερο μειονέκτημα του απέναντι σε αντίστοιχους παίκτες πόκερ TN. Μελλοντικά το πρόγραμμα της εργασίας αυτής θα μπορούσε να αναθεωρηθεί, ώστε να έχει τη δυνατότητα να παίζει και ενάντια σε άλλα προγράμματα πόκερ με παράλληλη ανάπτυξη ενός συστήματος μοντελοποίησης του αντιπάλου. Με αυτόν τον τρόπο θα μπορούσαμε να εξάγουμε πιο ασφαλή συμπεράσματα για την απόδοση του RezAI, έχοντας την δυνατότητα να εκτελέσουμε έναν τεράστιο αριθμό πειραμάτων.

Καμία στρατηγική πόκερ δεν είναι ολοκληρωμένη χωρίς ένα καλό σύστημα μοντελοποίησης. Ένας ισχυρός παίκτης πόκερ πρέπει να είναι σε θέση να αναπτύξει ένα δυναμικά προσαρμόσιμο μοντέλο για κάθε αντίπαλο, ώστε να αναγνωρίσει πιθανές αδυναμίες.

Στα παραδοσιακά παιχνίδια, όπως το σκάκι, αυτό το κομμάτι της στρατηγικής δεν είναι απαραίτητο για να επιτύχει ένα παίξιμο ύψιστου επιπέδου. Στα παιχνίδια πλήρους πληροφορίας, είναι επαρκές να ακολουθείται η βέλτιστη ενέργεια χωρίς να λαμβάνεται υπόψιν ο αντίπαλος. Εάν ο αντίπαλος παίζει χειρότερα, τότε συνεχίζοντας να ακολουθείται η βέλτιστη ενέργεια θα εκμεταλλευτεί τα λάθη του αντιπάλου. Η μοντελοποίηση του αντιπάλου έχει μελετηθεί στο πλαίσιο των παιχνιδιών με δύο παίκτες, χωρίς να υπάρχει ιδιαίτερη βελτίωση στην απόδοση.

Στο πόκερ, η κατάσταση είναι διαφορετική. Δύο αντίπαλοι μπορεί να κάνουν διαφορετικού τύπου λάθη, τα οποία μπορούν και τα δύο να είναι εκμεταλλεύσιμα από τον αντίπαλο, αλλά με διαφορετικές ενέργειες για τον καθένα. Για παράδειγμα, ο ένας μπορεί να μπλοφάρει πάρα πολύ, ο άλλος λίγο. Προσαρμόζουμε το παίξιμο ώστε να κάνουμε call περισσότερες φορές για τον πρώτο και λιγότερες για τον δεύτερο. Αν απλά ακολουθούσαμε την βέλτιστη ενέργεια, χωρίς να μοντελοποιήσουμε τον αντίπαλο, τότε θα χάναμε την ευκαιρία να αυξήσουμε τα κέρδη μας, το οποίο είναι και το ζητούμενο του παιχνιδιού. Ακόμα και οι πιο δυνατοί παίκτες μπορεί να ακολουθήσουν εντελώς διαφορετικά στυλ παιξίματος, γι

αυτό και είναι σημαντικό να προσπαθήσουμε να συμπεράνουμε την βασική προσέγγιση του παιχνιδιού για τον κάθε παίκτη, ανεξαρτήτως του πόσο καλά παίζει.

Η μοντελοποίηση του αντιπάλου στο πόκερ χρησιμοποιείται για τουλάχιστον δύο διαφορετικούς λόγους. Θέλουμε μια γενική μέθοδο καθορισμού της δύναμης του χεριού του αντιπάλου, βασιζόμενοι στις ενέργειες του. Επίσης, θέλουμε να προβλέψουμε την ενέργεια που θα κάνει σε μια συγκεκριμένη κατάσταση.

Γενικά, στο κέντρο ενός συστήματος μοντελοποίησης βρίσκεται ένας μηχανισμός που ονομάζεται predictor. Η δουλειά του predictor είναι να χαρτογραφεί κάθε πληροφορία του παιχνιδιού σε μια κατανομή πιθανότητας (probability) όσον αφορά τις πιθανές ενέργειες του αντιπάλου. Σε limit poker όπως αυτής της εργασίας, η κατανομή θα μπορούσε να γίνει με την μορφή μιας τριάδας πιθανοτήτων (probability triple) {Pr(fold), Pr(call), Pr(raise)}.

Ένας τρόπος να προβλεφθεί η ενέργεια του αντιπάλου θα ήταν να χρησιμοποιηθεί η δικιά μας στρατηγική πονταρίσματος, ή κάποιο άλλο σύνολο κανόνων, για να κάνουμε μια ορθολογιστική επιλογή εκ μέρους του αντιπάλου. Όταν χρησιμοποιούμε ως predictor μια τέτοιου είδους καθορισμένη στρατηγική, υποθέτουμε πως ο αντίπαλος θα παίξει με έναν σχετικά λογικό τρόπο και αναφερόμαστε σε αυτόν ως γενική μοντελοποίηση αντιπάλου (generic opponent modeling).

Άλλη μία μέθοδος για την πρόβλεψη των ενεργειών του αντιπάλου είναι να υποθέσουμε πως θα συνεχίσουν να συμπεριφέρονται όπως και πριν. Για παράδειγμα, αν παρατηρηθεί ότι ο αντίπαλος ποντάρει 4 στις 10 φορές αμέσως μετά το flop, τότε μπορούμε να θεωρήσουμε ότι θα ποντάρει με το καλύτερο 40% των χεριών του. Όταν χρησιμοποιούμε το προσωπικό ιστορικό ενεργειών ενός αντιπάλου για να κάνουμε προβλέψεις, τότε ονομάζεται συγκεκριμένη μοντελοποίηση αντιπάλου (specific opponent modeling).

Επίσης, μια πιθανώς καλή λύση για την μοντελοποίηση του αντιπάλου θα ήταν η δημιουργία ενός μηχανισμού ο οποίος θα αποθηκεύει δυναμικά την κατάσταση του παιχνιδιού, τις ενέργειες που έγιναν και την τελική έκβαση, ασχέτως με ποιον αντίπαλο παίζει. Μετά από ένα μεγάλο πλήθος πειραμάτων, ο RezAI θα είναι σε θέση να επιλέξει την ενέργεια που θα κάνει, συμβουλευόμενος και την στατιστικά βέλτιστη λύση για την συγκεκριμένη κατάσταση παιχνιδιού.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] ACPC, The annual computer poker competition. (2013). Retrieved May 6, 2013, from http://www.computerpokercompetition.org/

[2] Billings, D., Schaeffer, J., Pea, L. and Szafron, D. (1999), Learning to play strong poker, in: proceedings of the Sixteenth International Conference on Machine Learning (ICML-99), J. Stefan Institute, Slovenia, 1999.

[3] Billings, D., Davidson, A., Schaeffer, J. and Szafron, D. (2002), The challenge of poker. Artificial Intelligence: Chips challenging champions – games, computers and Artificial Intelligence, 134 (1-2), 201-240.

[4] Billings, D., Papp, D., Schaeffer, J. and Szafron, D. (1998), Opponent modeling in poker, in*:* Proceedings of AAAI-98 (15th National AAAI Conference), Sweden, 1998, pp. 493–499.

[5] Billings, D. (2006), Algorithms & assessment in computer poker, Ph.D. thesis, University of Alberta.

[6] Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T. and Szafron D. (2003), Approximating game-theoretic optimal strategies for full-scale poker, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), 2003, pp. 661–668.

[7] Follek, R. (2003), SoarBot: A rule-based system for playing poker, Master's thesis, Pace University.

[8] Gilpin, A. and Sandholm, T. (2007), Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em Poker, in: 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), 2007, pp. 192–200.

[9] Johanson, M. (2007), Robust strategies and counter-strategies: Building a champion level computer poker player, Master's thesis, University of Alberta.

[10] Koller, D. and Pfeffer, A. (1995), Generating and solving imperfect information games, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, 1995, pp. 1185–1192.

[11] Nash, J. (1950), Equilibrium points in n-person games, in: Proceedings of the National Academy of Science of the United States of America, Vol. 36, No. 1, (Jan. 15. 1950), pp. 48-49.

[12] Papp, D. (1998), Dealing with imperfect information in poker, Master's thesis, University of Alberta.

[13] Schauenberg, T. (2006), Opponent modelling and search in poker, Master's thesis, University of Alberta.

[14] Schuijtvlot, E. (2011), Application of AI in poker, Study business mathematics and informatics, VU University Amsterdam.

[15] Smith, J.J. and Nau, D. (1993), Strategic planning for imperfect-information games, Games: Planning and learning, Papers from the Fall Symposium., AAAI Press, 1993, pp. 84–91.

[16] The Second Man-Machine Poker Competition. (2008). Retrieved May 6, 2013, from http://webdocs.cs.ualberta.ca/~games/poker/man-machine/Results/

ΠΑΡΑΡΤΗΜΑ Α – ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

```
%----------------------- DECK -----------------------
% rnd_deck/1
% random permutation of the deck list, storing the first 9 cards

rnd_deck(_):-
        deck(Deck),
        rnd_permu(Deck,L),
        remove_at(Card1,L,1,L2),
        nb_setval(card1,Card1),
        remove_at(Card2,L2,1,L3),
        nb_setval(card2,Card2),
        remove_at(Card3,L3,1,L4),
        nb_setval(card3,Card3),
        remove_at(Card4,L4,1,L5),
        nb_setval(card4,Card4),
        remove_at(Card5,L5,1,L6),
        nb_setval(card5,Card5),
        remove_at(Card6,L6,1,L7),
        nb_setval(card6,Card6),
        remove_at(Card7,L7,1,L8),
        nb_setval(card7,Card7),
        remove_at(Card8,L8,1,L9),
        nb_setval(card8,Card8),
        remove_at(Card9,L9,1,_),
        nb_setval(card9,Card9).
```

```
deck( [ (a|h), (k|h), (q|h), (j|h), (10|h), (9|h), (8|h), (7|h), (6|h), (5|h), (4|h), (3|h), (2|h),
                (a|d), (k|d), (q|d), (j|d), (10|d), (9|d), (8|d), (7|d), (6|d), (5|d), (4|d), (3|d),
(2|d),
                (a|s), (k|s), (q|s), (j|s), (10|s), (9|s), (8|s), (7|s), (6|s), (5|s), (4|s), (3|s), (2|s),
                (a|c), (k|c), (q|c), (j|c), (10|c), (9|c), (8|c), (7|c), (6|c), (5|c), (4|c), (3|c), (2|c) ]
).


% Remove the K'th element from a list.
% The first element in the list is number 1.
% remove_at(X,L,K,R) :- X is the K'th element of the list L;
% R is the list that remains when the K'th element is removed from L.
%    (element,list,integer,list) (?,?,+,?)

remove_at(X,[X|Xs],1,Xs).
remove_at(X,[Y|Xs],K,[Y|Ys]) :- K > 1,
  K1 is K - 1, remove_at(X,Xs,K1,Ys).

% Extract a given number of randomly selected elements from a list.
% rnd_select(L,N,R) :- the list R contains N randomly selected
%    items taken from the list L.
%    (list,integer,list) (+,+,-)

rnd_select(_,0,[]).
rnd_select(Xs,N,[X|Zs]) :- N > 0,
   length(Xs,L),
   I is random(L) + 1,
   remove_at(X,Xs,I,Ys),
   N1 is N - 1,
   rnd_select(Ys,N1,Zs).
% Generate a random permutation of the elements of a list
% rnd_permu(L1,L2) :- the list L2 is a random permutation of the
%    elements of the list L1.
%    (list,list) (+,-)
```

```
rnd_permu(L1,L2) :- length(L1,N), rnd_select(L1,N,L2).

%-------------------------------- RezAI Strategy Handling --------------------------------
% updateChipsLost/1
% Amount represents the chips lost incase of fold or loose at showdown

updateChipsLost(Amount) :-
        Amount > 0,
        nb_getval(chipsLost, X),
        TotalChipsLost is Amount + X,
        nb_setval(chipsLost, TotalChipsLost).

% changeStrategy/3
% if 100 or more chips were lost and AI has less or equal chips with player, change the strategy
changeStrategy(MyChips, OpponentChips, NewStrategy) :-
        MyChips =< OpponentChips,
        nb_getval(chipsLost, ChipsLost),
        ChipsLost >= 100,
        randomStrategy(NS),
        nb_setval(chipsLost, 0),
        NewStrategy is NS.

% randomStrategy/1
randomStrategy(X) :-
        random(1, 5, X).

%-------------------------------- HAND STRENGTH --------------------------------
% USED ONLY in pre-Flop
% handPairStrength(Card1, Card2, Strength)
% handPairStrength/3
```

```
handPairStrength(C1, C2, S):-
        handPair(C1, C2, S1),
        suited(C1, C2, S1, S2),
        S is S2.

% suited(Card1, Card2, Strength, NewStrength)
% suited/4
suited(_|X, _|Y, S1, S2):-
        X == Y,
        S2 is S1 + 10.

suited(_|X, _|Y, S1, S2):-
        X \== Y,
        S2 is S1.

% USED ONLY in pre-Turn
% handStrength/2

handStrength(L, Strength):-
        isStraightFlush(L, _, Highest) ->
                checkMinorStrength(Highest, Str),
                Strength is 1020+Str;
        isFourOfAKind(L, Rank) ->
                checkMinorStrength(Rank, Str),
                Strength is 1000+Str;
        isFullHouse(L) ->
                minorStrength(3, MS),
                minorStrength(2, MS2),
                Strength is 815+(MS*13)+MS2;
        isFlush(L, _, Highest) ->
                checkMinorStrength(Highest, Str),
                Strength is 800+Str;
```

```
isStraight(L, _, Highest) ->
        checkMinorStrength(Highest, Str),
        Strength is 785+Str;
isThreeOfAKind(L) ->
        remove_at(C1, L, 1, L2),
        remove_at(C2, L2, 1, _),
        sort_two_strength([C1, C2], 3, Str),
        Strength is 600+Str;
isTwoPairs(L) ->
        minorStrength(2, MS),
        minorStrength(2, MS2),
        sort_strength([MS, MS2], SortedStr),
        remove_at(Lower, SortedStr, 1, SortedStr2),
        remove_at(Higher, SortedStr2, 1, _),
        Strength is 400+(Higher*13)+Lower;
isOnePair(L) ->
        remove_at(C1, L, 1, L2),
        remove_at(C2, L2, 1, _),
        sort_two_strength([C1, C2], 2, Str),
        Strength is 200+Str;
isHighCard(L) ->
        remove_at(C1, L, 1, L2),
        remove_at(C2, L2, 1, _),
        sort_cards_Ahigh([C1, C2], SortedList),
        remove_at(HighCard, SortedList, 2, SortedList2),
        remove_at(HighCard2, SortedList2, 1, _),
        getHeader(HighCard, HC),
        getHeader(HighCard2, HC2),
        checkMinorStrength(HC, Str),
        checkMinorStrength(HC2, Str2),
        Strength is Str2+(Str*13);
Strength is 0.
```

```
% potentialStrength/6
potentialStrength(C1, C2, BC1, BC2, BC3, Strength):-
        isStraight(C1, C2, BC1, BC2, BC3, _, _) ->
                Strength is 1;
        isFlush(C1, C2, BC1, BC2, BC3, _, _) ->
                Strength is 1;
        Strength = 0.

% minorStrength/2
minorStrength(Occurrences, Strength):-
        occurrences(X, Occurrences),
        checkMinorStrength(X, S),
        retract(occurrences(X,Occurrences)),
        Strength is S.

% checkMinorStrength/2
checkMinorStrength(X, S):-
        isAce(X) -> S is 13;
        isKing(X) -> S is 12;
        isQueen(X) -> S is 11;
        isJack(X) -> S is 10;
        isTen(X) -> S is 9;
        isNine(X) -> S is 8;
        isEight(X) -> S is 7;
        isSeven(X) -> S is 6;
        isSix(X) -> S is 5;
        isFive(X) -> S is 4;
        isFour(X) -> S is 3;
        isThree(X) -> S is 2;
        isTwo(X) -> S is 1;
        S is 0.
```

```
isAce(a).
isKing(k).
isQueen(q).
isJack(j).
isTen(10).
isNine(9).
isEight(8).
isSeven(7).
isSix(6).
isFive(5).
isFour(4).
isThree(3).
isTwo(2).

sort_strength(L, R):-
        predsort(compare_occurrences_strength, L, R).

compare_occurrences_strength(D, X, Y):-
        compare(D, X, Y).

getHeader(H|_, H).

sort_two_strength([C1, C2], X, Strength) :-
        get_two_strength(C1, C2, X, S),
        Strength is S.

get_two_strength(C1, C2, X, S) :-
        getHeader(C1, HC1),
        getHeader(C2, HC2),
        HC1 == HC2,
        minorStrength(X, MS),
        S is MS+(13*MS).
```

```
get_two_strength(C1, C2, X, S) :-
        getHeader(C1, HC1),
        getHeader(C2, HC2),
        HC1 \= HC2,
        sort_cards_Ahigh([C1, C2], SortedList),
        remove_at(HighCard, SortedList, 2, _),
        getHeader(HighCard, HC),
        checkMinorStrength(HC, Str),
        minorStrength(X, MS),
        S is Str+(13*MS).

%------------------------- HAND COMBINATIONS -------------------------
countall([],_,_).
countall(L, E|T, Count):-
        findall(E|T, member(E|T, L), List),
        length(List, Count),
        assert(occurrences(E,Count)).

insert_sort(List,Sorted):-
        i_sort(List,[],Sorted).

i_sort([],Acc,Acc).
i_sort([H|T],Acc,Sorted):-
        insert(H,Acc,NAcc),
        i_sort(T,NAcc,Sorted).

insert(X,[],[X]).
insert(X,[Y|T],[Y|NT]):-
        X>Y,insert(X,T,NT).
insert(X,[Y|T],[X,Y|T]):-
        X=<Y.
```

```
% Pre-Turn, 2 cards at hand and 3 on the board
% isHighCard/1
% isHighCard(List)

isHighCard(1, 1).
isHighCard(L):-
        retractall(occurrences(_,_)),
        countall(L, a|_, Oca),
        countall(L, 2|_, Oc2),
        countall(L, 3|_, Oc3),
        countall(L, 4|_, Oc4),
        countall(L, 5|_, Oc5),
        countall(L, 6|_, Oc6),
        countall(L, 7|_, Oc7),
        countall(L, 8|_, Oc8),
        countall(L, 9|_, Oc9),
        countall(L, 10|_, Oc10),
        countall(L, j|_, Ocj),
        countall(L, q|_, Ocq),
        countall(L, k|_, Ock),
        insert_sort([Oca, Oc2, Oc3, Oc4, Oc5, Oc6, Oc7, Oc8, Oc9, Oc10, Ocj, Ocq,
        Ock],SortedList),
        remove_at(C1,SortedList,13,SortedList2),
        remove_at(C2,SortedList2,12,_),
        isHighCard(C1, C2).

% isOnePair/1
% isOnePair(List)
isOnePair(2, 1).
isOnePair(L):-
        retractall(occurrences(_,_)),
        countall(L, a|_, Oca),
```

```
        countall(L, 2|_, Oc2),
        countall(L, 3|_, Oc3),
        countall(L, 4|_, Oc4),
        countall(L, 5|_, Oc5),
        countall(L, 6|_, Oc6),
        countall(L, 7|_, Oc7),
        countall(L, 8|_, Oc8),
        countall(L, 9|_, Oc9),
        countall(L, 10|_, Oc10),
        countall(L, j|_, Ocj),
        countall(L, q|_, Ocq),
        countall(L, k|_, Ock),
        insert_sort([Oca, Oc2, Oc3, Oc4, Oc5, Oc6, Oc7, Oc8, Oc9, Oc10, Ocj, Ocq,
        Ock],SortedList),
        remove_at(C1,SortedList,13,SortedList2),
        remove_at(C2,SortedList2,12,_),
        isOnePair(C1, C2).

% isTwoPairs/1
% isTwoPairs(List)
isTwoPairs(2, 2).
isTwoPairs(L):-
        retractall(occurrences(_,_)),
        countall(L, a|_, Oca),
        countall(L, 2|_, Oc2),
        countall(L, 3|_, Oc3),
        countall(L, 4|_, Oc4),
        countall(L, 5|_, Oc5),
        countall(L, 6|_, Oc6),
        countall(L, 7|_, Oc7),
        countall(L, 8|_, Oc8),
        countall(L, 9|_, Oc9),
```

```
        countall(L, 10|_, Oc10),
        countall(L, j|_, Ocj),
        countall(L, q|_, Ocq),
        countall(L, k|_, Ock),
        insert_sort([Oca, Oc2, Oc3, Oc4, Oc5, Oc6, Oc7, Oc8, Oc9, Oc10, Ocj, Ocq,
Ock],SortedList),
        remove_at(C1,SortedList,13,SortedList2),
        remove_at(C2,SortedList2,12,_),
        isTwoPairs(C1, C2).


% isThreeOfAKind/1
% isThreeOfAKind(List)
isThreeOfAKind(3, 1).
isThreeOfAKind(L):-
        retractall(occurrences(_,_)),
        countall(L, a|_, Oca),
        countall(L, 2|_, Oc2),
        countall(L, 3|_, Oc3),
        countall(L, 4|_, Oc4),
        countall(L, 5|_, Oc5),
        countall(L, 6|_, Oc6),
        countall(L, 7|_, Oc7),
        countall(L, 8|_, Oc8),
        countall(L, 9|_, Oc9),
        countall(L, 10|_, Oc10),
        countall(L, j|_, Ocj),
        countall(L, q|_, Ocq),
        countall(L, k|_, Ock),
        insert_sort([Oca, Oc2, Oc3, Oc4, Oc5, Oc6, Oc7, Oc8, Oc9, Oc10, Ocj, Ocq,
Ock],SortedList),
        remove_at(C1,SortedList,13,SortedList2),
        remove_at(C2,SortedList2,12,_),
        isThreeOfAKind(C1, C2).
```

```
% straight - card list gets sorted
% isStraight/3
% isStraight(List, ResultList, HighestCard)
isStraight(L, R, Highest):-
        isStraight_Alow(L, R, Highest) ; isStraight_Ahigh(L, R, Highest).

isStraight_Alow(L, R, Highest):-
        sort_cards_Alow(L, R),
        isStraight(R, Highest).

isStraight_Ahigh(L, R, Highest):-
        sort_cards_Ahigh(L, R),
        isStraight(R, Highest).

isStraight([(a|_),(2|_),(3|_),(4|_),(5|_)], 5).
isStraight([(2|_),(3|_),(4|_),(5|_),(6|_)], 6).
isStraight([(3|_),(4|_),(5|_),(6|_),(7|_)], 7).
isStraight([(4|_),(5|_),(6|_),(7|_),(8|_)], 8).
isStraight([(5|_),(6|_),(7|_),(8|_),(9|_)], 9).
isStraight([(6|_),(7|_),(8|_),(9|_),(10|_)], 10).
isStraight([(7|_),(8|_),(9|_),(10|_),(j|_)], j).
isStraight([(8|_),(9|_),(10|_),(j|_),(q|_)], q).
isStraight([(9|_),(10|_),(j|_),(q|_),(k|_)], k).
isStraight([(10|_),(j|_),(q|_),(k|_),(a|_)], a).

sort_cards_Alow(L, R) :-
    predsort(compare_values_Alow, L, R).

compare_values_Alow(D, (A|_), (B|_)) :-
    nth0(X, [a, 2, 3, 4, 5, 6, 7, 8, 9, 10, j, q, k], A),
    nth0(Y, [a, 2, 3, 4, 5, 6, 7, 8, 9, 10, j, q, k], B),
```

```
    compare(D, X, Y).

sort_cards_Ahigh(L, R) :-
    predsort(compare_values_Ahigh, L, R).

compare_values_Ahigh(D, (A|_), (B|_)) :-
    nth0(X, [2, 3, 4, 5, 6, 7, 8, 9, 10, j, q, k, a], A),
    nth0(Y, [2, 3, 4, 5, 6, 7, 8, 9, 10, j, q, k, a], B),
    compare(D, X, Y).

% isFlush/3
% isFlush(List)
isFlush(L, R, Highest):-
        sort_cards_flush(L, R),
        isFlush(R, Highest).

isFlush([(High|h),(_|h),(_|h),(_|h),(_|h)], High).
isFlush([(High|d),(_|d),(_|d),(_|d),(_|d)], High).
isFlush([(High|s),(_|s),(_|s),(_|s),(_|s)], High).
isFlush([(High|c),(_|c),(_|c),(_|c),(_|c)], High).

sort_cards_flush(L, R) :-
    predsort(compare_values_flush, L, R).

compare_values_flush(D, (A|Suit), (B|Suit)) :-
    nth0(X, [a, k, q, j, 10, 9, 8, 7, 6, 5, 4, 3, 2], A),
    nth0(Y, [a, k, q, j, 10, 9, 8, 7, 6, 5, 4, 3, 2], B),
    compare(D, X, Y).

% isFullHouse/1
% isFullHouse(List)
isFullHouse(3, 2).
```

```
isFullHouse(3, 3).

isFullHouse(L):-

        retractall(occurrences(_,_)),

        countall(L, a|_, Oca),

        countall(L, 2|_, Oc2),

        countall(L, 3|_, Oc3),

        countall(L, 4|_, Oc4),

        countall(L, 5|_, Oc5),

        countall(L, 6|_, Oc6),

        countall(L, 7|_, Oc7),

        countall(L, 8|_, Oc8),

        countall(L, 9|_, Oc9),

        countall(L, 10|_, Oc10),

        countall(L, j|_, Ocj),

        countall(L, q|_, Ocq),

        countall(L, k|_, Ock),

        insert_sort([Oca, Oc2, Oc3, Oc4, Oc5, Oc6, Oc7, Oc8, Oc9, Oc10, Ocj, Ocq,
        Ock],SortedList),

        remove_at(C1,SortedList,13,SortedList2),

        remove_at(C2,SortedList2,12,_),

        isFullHouse(C1, C2).


% isFourOfAKind/2

% isFourOfAKind(List, Rank)

isFourOfAKind([(X|_),(X|_),(X|_),(X|_),(_|_)], X).

isFourOfAKind([(X|_),(X|_),(X|_),(_|_),(X|_)], X).

isFourOfAKind([(X|_),(X|_),(_|_),(X|_),(X|_)], X).

isFourOfAKind([(X|_),(_|_),(X|_),(X|_),(X|_)], X).

isFourOfAKind([(_|_),(X|_),(X|_),(X|_),(X|_)], X).

% isStraightFlush/3

% isStraightFlush(List, ResultList)

isStraightFlush(L, R, Highest):-
```

```
        isStraight_AlowFlush(L, R, Highest) ; isStraight_AhighFlush(L, R, Highest).

isStraight_AlowFlush(L, R, Highest):-
        sort_cards_Alow(L, R),
        isStraightFlush(R, Highest).

isStraight_AhighFlush(L, R, Highest):-
        sort_cards_Ahigh(L, R),
        isStraightFlush(R, Highest).

isStraightFlush([(a|X),(2|X),(3|X),(4|X),(5|X)], 5).
isStraightFlush([(2|X),(3|X),(4|X),(5|X),(6|X)], 6).
isStraightFlush([(3|X),(4|X),(5|X),(6|X),(7|X)], 7).
isStraightFlush([(4|X),(5|X),(6|X),(7|X),(8|X)], 8).
isStraightFlush([(5|X),(6|X),(7|X),(8|X),(9|X)], 9).
isStraightFlush([(6|X),(7|X),(8|X),(9|X),(10|X)], 10).
isStraightFlush([(7|X),(8|X),(9|X),(10|X),(j|X)], j).
isStraightFlush([(8|X),(9|X),(10|X),(j|X),(q|X)], q).
isStraightFlush([(9|X),(10|X),(j|X),(q|X),(k|X)], k).
isStraightFlush([(10|X),(j|X),(q|X),(k|X),(a|X)], a).

% handPair(Card1, Card2, handPairStrength)
% handPair Value for unsuited pair of cards, USED ONLY in pre-flop
handPair(a|_, a|_, 100).
handPair(k|_, k|_, 90).
handPair(q|_, q|_, 85).
handPair(j|_, j|_, 70).
handPair(10|_, 10|_, 55).
handPair(9|_, 9|_, 50).
handPair(8|_, 8|_, 45).
handPair(7|_, 7|_, 40).
handPair(6|_, 6|_, 35).
```

```
handPair(5|_, 5|_, 30).
handPair(4|_, 4|_, 25).
handPair(3|_, 3|_, 20).
handPair(2|_, 2|_, 15).

handPair(a|_, k|_, 87).
handPair(a|_, q|_, 80).
handPair(a|_, j|_, 65).
handPair(a|_, 10|_, 50).
handPair(a|_, 9|_, 45).
handPair(a|_, 8|_, 40).
handPair(a|_, 7|_, 35).
handPair(a|_, 6|_, 30).
handPair(a|_, 5|_, 25).
handPair(a|_, 4|_, 20).
handPair(a|_, 3|_, 15).
handPair(a|_, 2|_, 10).
handPair(k|_, a|_, 87).
handPair(q|_, a|_, 80).
handPair(j|_, a|_, 65).
handPair(10|_, a|_, 50).
handPair(9|_, a|_, 45).
handPair(8|_, a|_, 40).
handPair(7|_, a|_, 35).
handPair(6|_, a|_, 30).
handPair(5|_, a|_, 25).
handPair(4|_, a|_, 20).
handPair(3|_, a|_, 15).
handPair(2|_, a|_, 10).
```

```
% ---------------------------- STRATEGIES ----------------------------

% Action Value Range

% 0 = Fold, 1 = Check, 2 = Call, 3 = Bet, 4 = Raise.

%

% Strategy Value Range

% 0 = Random, 1 = Loose Passive, 2 = Loose Aggressive, 3 = Tight Passive, 4 = Tight
Aggressive.

%

% GameState Value Range

% 0 = pre-flop, 1 = pre-turn, 2 = pre-river, 3 = after-river

%

% PotentialStrength Value Range

% 0 = No potential, 1 = need one card to straight or flush

%

% HandStrength is calculated using the terms above

%

% OpponentAction Value Range

% 0 = No Action, 1 = Check, 2 = Call, 3 = Bet, 4 = Raise

%

% MyChipStatus = MyChips//Pot ,integer division

% Shows the status of AI's remaining chips compared to the current pot chips, lower numbers
have greater significance

%

% OpponentChipStatus = OpponentChips//Pot ,integer division

% Shows the status of opponent's remaining chips compared to the current pot chips, lower
numbers have greater significance

%

% ActionsLeft Value Range

% 1 to 6 . An integer value showing how many action are left for both players combined in the
current game state (Max 6).

%

% no_style_check(discontiguous).

%
```

```
% checkAction/8


checkAction(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
MyChipStatus, OpponentChipStatus, ActionsLeft, Action):-

        isFold(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
        MyChipStatus, OpponentChipStatus, ActionsLeft) -> Action is 0;

        isCheck(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
        MyChipStatus, OpponentChipStatus, ActionsLeft) -> Action is 1;

        isCall(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
        MyChipStatus, OpponentChipStatus, ActionsLeft) -> Action is 2;

        isBet(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
        MyChipStatus, OpponentChipStatus, ActionsLeft) -> Action is 3;

        isRaise(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
        MyChipStatus, OpponentChipStatus, ActionsLeft) -> Action is 4;

        Action is 0. %safety net


% <ACTION>(Strategy, GameState, HandStrength, PotentialStrength, OpponentAction,
MyChipStatus, OpponentChipStatus, ActionsLeft).

% ------------Loose Aggressive Strategy-------------

% -----Pre Flop-----


isFold(2, 0, HandStrength, _, OpponentAction, _, _, ActionsLeft):-

        HandStrength =< 5,

        OpponentAction > 2,

        ActionsLeft > 1.


isCheck(2, 0, HandStrength, _, OpponentAction, _, _, ActionsLeft):-

        HandStrength < 24,

        OpponentAction < 3,

        ActionsLeft > 1.


isCall(2, 0, HandStrength, _, OpponentAction, _, _, ActionsLeft):-

        HandStrength > 5,

        HandStrength =< 24,

        OpponentAction > 2,
```

```
        ActionsLeft > 1.

isBet(2, 0, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 24,
        OpponentAction < 3,
        ActionsLeft > 1.

isRaise(2, 0, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength > 24,
        OpponentAction > 2,
        ActionsLeft > 1.


isFold(2, 0, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength =< 5,
        OpponentAction > 2.

isCall(2, 0, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength > 5,
        OpponentAction > 2.

% -----Pre Turn-----

isFold(2, 1, HandStrength, 0, OpponentAction, MyChipStatus, OpponentChipStatus, ActionsLeft):-
        HandStrength < 130,
        OpponentAction > 2,
        MyChipStatus < OpponentChipStatus,
        ActionsLeft > 1.

isCheck(2, 1, HandStrength, 0, OpponentAction, _, _, ActionsLeft):-
        HandStrength < 130,
        OpponentAction < 3,
```

```
        ActionsLeft > 1.

isCall(2, 1, HandStrength, _, OpponentAction, MyChipStatus, OpponentChipStatus, ActionsLeft):-
        HandStrength < 130,
        OpponentAction > 2,
        MyChipStatus >= OpponentChipStatus,
        ActionsLeft > 1.

isCall(2, 1, HandStrength, 1, OpponentAction, MyChipStatus, OpponentChipStatus, ActionsLeft):-
        HandStrength < 130,
        OpponentAction > 2,
        MyChipStatus < OpponentChipStatus,
        ActionsLeft > 1.

isBet(2, 1, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 130,
        OpponentAction < 3,
        ActionsLeft > 1.

isBet(2, 1, HandStrength, 1, OpponentAction, _, _, ActionsLeft):-
        HandStrength < 130,
        OpponentAction < 3,
        ActionsLeft > 1.

isRaise(2, 1, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 130,
        OpponentAction > 2,
        ActionsLeft > 1.

isFold(2, 1, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength < 130,
        OpponentAction > 2.
```

```
isCall(2, 1, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength >= 130,
        OpponentAction > 2.

% -----Pre River-----

isFold(2, 2, HandStrength, 0, OpponentAction, MyChipStatus, OpponentChipStatus, ActionsLeft):-
        HandStrength < 200,
        OpponentAction > 2,
        MyChipStatus < OpponentChipStatus,
        ActionsLeft > 1.

isCheck(2, 2, HandStrength, 0, OpponentAction, _, _, ActionsLeft):-
        HandStrength < 200,
        OpponentAction < 3,
        ActionsLeft > 1.

isCall(2, 2, HandStrength, _, OpponentAction, MyChipStatus, OpponentChipStatus, ActionsLeft):-
        HandStrength < 200,
        OpponentAction > 2,
        MyChipStatus >= OpponentChipStatus,
        ActionsLeft > 1.

isCall(2, 2, HandStrength, 1, OpponentAction, MyChipStatus, OpponentChipStatus, ActionsLeft):-
        HandStrength < 200,
        OpponentAction > 2,
        MyChipStatus < OpponentChipStatus,
        ActionsLeft > 1.

isBet(2, 2, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 200,
```

```
        OpponentAction < 3,
        ActionsLeft > 1.

isBet(2, 2, HandStrength, 1, OpponentAction, _, _, ActionsLeft):-
        HandStrength < 200,
        OpponentAction < 3,
        ActionsLeft > 1.

isRaise(2, 2, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 200,
        OpponentAction > 2,
        ActionsLeft > 1.

isFold(2, 2, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength < 200,
        OpponentAction > 2.

isCall(2, 2, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength >= 200,
        OpponentAction > 2.

 % -----After River-----

isFold(2, 3, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength < 291,
        OpponentAction > 2,
        ActionsLeft > 1.

isCheck(2, 3, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength < 291,
        OpponentAction < 3,
        ActionsLeft > 1.
```

```
isCall(2, 3, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 291,
        HandStrength < 400,
        OpponentAction > 2,
        ActionsLeft > 1.

isBet(2, 3, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 291,
        OpponentAction < 3,
        ActionsLeft > 1.

isRaise(2, 3, HandStrength, _, OpponentAction, _, _, ActionsLeft):-
        HandStrength >= 400,
        OpponentAction > 2,
        ActionsLeft > 1.

isFold(2, 3, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength < 291,
        OpponentAction > 2.

isCall(2, 3, HandStrength, _, OpponentAction, _, _, 1):-
        HandStrength >= 291,
        OpponentAction > 2.
```

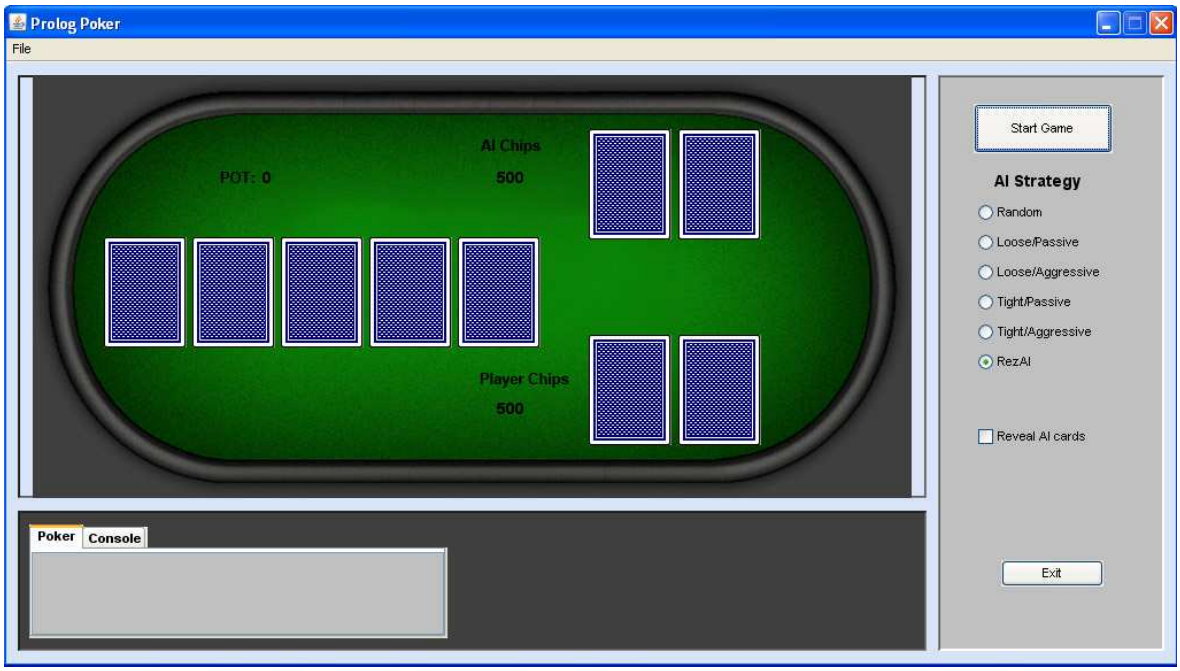
ΠΑΡΑΡΤΗΜΑ Β - ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Για να μπορέσουμε να εκτελέσουμε την εφαρμογή που δημιουργήθηκε στην εργασία αυτή, είναι απαραίτητο να είναι εγκατεστημένο στο σύστημα μας το JRE (Java Runtime Environment) της Java, όπως επίσης και η SWI-Prolog. Ακόμα, για να λειτουργήσει σωστά η βιβλιοθήκη JPL, θα πρέπει να έχουμε ενημερώσει την μεταβλητή συστήματος PATH, με την διαδρομή του φακέλου bin, που βρίσκεται μέσα στον φάκελο όπου έχουμε εγκαταστήσει την SWI-Prolog.

Για να ξεκινήσουμε την εφαρμογή, εφόσον τηρούνται οι παραπάνω προϋποθέσεις, αρκεί να εκτελέσουμε το αρχείο “poker.jar” από το γραφικό περιβάλλον του λειτουργικού μας συστήματος, ή εκτελώντας την παρακάτω εντολή από τη γραμμή εντολών:

java –jar poker.jar

Στη συνέχεια θα παρουσιάσουμε και θα σχολιάσουμε μερικά στιγμιότυπα της εφαρμογής.



Σχήμα 29 “Στιγμιότυπο εφαρμογής – Αρχική κατάσταση”

Το σχήμα 29 απεικονίζει την αρχική κατάσταση της εφαρμογής. Σε αυτό το σημείο έχουμε την επιλογή να ξεκινήσουμε το παιχνίδι, πατώντας το κουμπί “Start Game”, που βρίσκεται στο πάνω μέρος του δεξιού πλαισίου. Ακόμα, ακριβώς από κάτω, μπορούμε να αλλάξουμε την στρατηγική που θα ακολουθεί ο παίκτης ΤΝ και να επιλέξουμε το αν θα βλέπουμε τα φύλλα του, πατώντας το “Reveal AI cards”. Με το πάτημα του κουμπιού “Exit” η εφαρμογή τερματίζεται.

Σχήμα 30 "Στιγμιότυπο εφαρμογής – Μέση κατάσταση"

Στο σχήμα 30 ο χρήστης έχει ήδη πατήσει το κουμπί "Start Game" και το παιχνίδι έχει μόλις ξεκινήσει. Να αναφέρουμε πως πλέον το ίδιο κουμπί μετατράπηκε σε "Reload Game", όπου πατώντας το, θα επαναφέρουμε την εφαρμογή στο αρχικό στάδιο (σχήμα 29). Όσον αφορά καθαρά την παρτίδα πόκερ, οι ενέργειες που μπορεί να επιλέξει ο χρήστης βρίσκονται στο κάτω πλαίσιο, όπου υπάρχουν τα τρία κουμπιά. Επίσης, κάτω αριστερά υπάρχουν οι καρτέλες Poker και Console, που πληροφορούν τον χρήστη σχετικά με το παιχνίδι, μέσω μηνυμάτων στο πλαίσιο κειμένου τους.

Τέλος, το σχήμα 31, απεικονίζει την κατάσταση του παιχνιδιού όπου το παιχνίδι έχει ολοκληρωθεί, αφήνοντας στον χρήστη την επιλογή να ξεκινήσει από την αρχή ένα νέο, πατώντας το κουμπί "Reload Game", είτε να τερματίσει την εφαρμογή, πατώντας το κουμπί "Exit".

Σχήμα 31 "Στιγμιότυπο εφαρμογής – Τελική κατάσταση"