Πολυτεχνειο Κρητης
Σχολη Ηλεκτρολογων Μηχανικων Και Μηχανικων Υπολογιστων

Συστηματική Αναζήτηση και Ενισχυτική Μάθηση για το Επιτραπέζιο Παιχνίδι "Turning Points"

Διπλωματικη Εργασια

του

ΣΩΤΗΡΙΟΥ Σ. ΒΑΓΕΝΑ

Εξεταστική Επιτροπή

Αναπληρωτής Καθηγητής Μιχαήλ Γ. Λαγουδάκης (Επιβλέπων)
Αναπληρωτής Καθηγητής Γεώργιος Χαλκιαδάκης
Αναπληρωτής Καθηγητής Αντώνιος Δεληγιαννάκης

Χανιά, Νοέμβριος 2016

Technical University of Crete
School of Electrical and Computer Engineering

Systematic Search and Reinforcement Learning for the Board Game "Turning Points"

Diploma Thesis

of

Sotirios S. Vagenas

Thesis Committee

Associate Professor Michail G. Lagoudakis (Supervisor)
Associate Professor Georgios Chalkiadakis
Associate Professor Antonios Deligiannakis

Chania, November 2016

Περίληψη

Τα παιχνίδια ήταν ανέκαθεν ένα από τα κυριότερα μέσα ψυχαγωγίας για τους ανθρώπους. Συνδυάζουν τη διασκέδαση, την πνευματική ανάπτυξη και την ανάπτυξη υψηλών δεξιοτήτων. Για τους λόγους αυτούς, τα παιχνίδια αποτελούν έναν από τους σημαντικότερους τομείς έρευνας και εφαρμογής της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης. Η παρούσα διπλωματική εργασία εστιάζει στο επιτραπέζιο παιχνίδι "Turning Points". Πρόκειται για ένα περίπλοκο παιχνίδι τετραγωνικού board 4×4 δύο παικτών, στο οποίο κάθε κίνηση αντιστοιχεί στην τοποθέτηση ενός κατευθυνόμενου βέλους σε μία άδεια θέση, πιθανότατα προκαλώντας περιστροφές στα υπάρχοντα βέλη. Στόχος της εργασίας είναι η σχεδίαση ενός πράκτορα για το παιχνίδι Turning Points, ο οποίος θα μαθαίνει να παίζει και θα βελτιώνεται ενάντια σε διάφορους αντιπάλους. Η υλοποίηση του πράκτορα συνδυάζει την τεχνική αναζήτησης Minimax με α–β Pruning, μία συνάρτηση αξιολόγησης και τον αλγόριθμο ενισχυτικής μάθησης TD-Learning. Ο συνδυασμός των παραπάνω στοιχείων οδηγεί στην εξαγωγή πολλών αποδοτικών παικτών, όπως προκύπτει από τη πειραματική διαδικασία αξιολόγησης. Οι παίκτες αυτοί αντιμετωπίζουν τον άνθρωπο με πολύ μεγάλη αποτελεσματικότητα. Ενδιαφέρον, επίσης, παρουσιάζει το γεγονός ότι παραμένουν ανταγωνιστικοί σε boards μεγαλύτερων διαστάσεων από το βασικό στο οποίο έχουν εκπαιδευτεί. Τέλος, υλοποιήθηκε κατάλληλη γραφική διεπαφή για τη διεξαγωγή παιχνιδιών με χρήστες.

Abstract

Games have always been one of the main forms of entertainment for people. They combine fun, spiritual growth and skill development. For these reasons, games constitute one of the most important areas of research and application of Artificial Intelligence and Machine Learning. This thesis focuses on the board game "Turning Points". This is a challenging two-player 4×4 square board game, in which each move corresponds to the placement of a directed arrow to an empty position, possibly causing some rotations of the existing arrows. The goal of the thesis is the design of an agent for the game Turning Points, who will learn to play and improve against various opponents. The implementation of the agent combines the Minimax with alpha-beta Pruning search technique, an evaluation function and the reinforcement learning algorithm TD-Learning. The combination of these elements results in the extraction of several competitive players, as evidenced by the experimental evaluation procedure. These players faced human players competitively and efficiently. Interestingly, they also remain competitive in larger boards compared to the basic one in which they have been trained. Finally, a graphical user interface was implemented to enable game playing with users.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένειά μου, τη σύντροφό μου και τους φίλους μου, οι οποίοι με στηρίζουν σε κάθε βήμα της ζωής μου. Επίσης, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Μιχαήλ Γ. Λαγουδάκη για τη βοήθεια που μου παρείχε στην εκπόνηση της διπλωματικής μου εργασίας.

Χανιά, Νοέμβριος 2016

Σωτήριος Σ. Βαγενάς

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

Κεφάλαιο 1

Εισαγωγή

Η Ενισχυτική Μάθηση (Reinforcement Learning) συγκαταλέγεται στις δημοφιλέστερες γνωστικές περιοχές του ευρύτερου πεδίου της Τεχνητής Νοημοσύνης (Artificial Intelligence) και της Μηχανικής Μάθησης (Machine Learning). Χρησιμοποιείται από ευφυείς πράκτορες που βρίσκονται σε άγνωστα ή σχεδόν άγνωστα περιβάλλοντα, με σκοπό την εκμάθηση συμπεριφορών που οδηγούν στην εκπλήρωση συγκεκριμένων στόχων. Η εκμάθηση ενός πράκτορα είναι, ουσιαστικά, η ικανότητά του να εκμεταλλεύεται τις γνώσεις που έχει, τις παρατηρήσεις από το περιβάλλον του και τις επιδράσεις του σε αυτό, με σκοπό τη βελτίωση των ικανοτήτων του. Η ανάγκη των ευφυών πρακτόρων για εκμάθηση σε διάφορα περιβάλλοντα με διαφορετικά χαρακτηριστικά οδήγησε στην επινόηση πολλών αλγορίθμων μάθησης.

Το πρόβλημα της Ενισχυτικής Μάθησης μπορεί να περιγραφεί συνοπτικά ως το πρόβλημα της εκμάθησης μιας συμπεριφοράς μέσω δοκιμής και αποτυχίας (trial and error) από έναν ευφυή πράκτορα που αλληλεπιδρά με ένα δυναμικό περιβάλλον. Ένας τρόπος προσέγγισης του προβλήματος αυτού, είναι η αναζήτηση στο χώρο των πιθανών συμπεριφορών, δηλαδή στους πιθανούς τρόπους λήψης αποφάσεων, ώστε σε κάθε πιθανή κατάσταση του περιβάλλοντος να είναι σε θέση ο πράκτορας να επιλέγει ενέργειες, οι οποίες μακροπρόθεσμα οδηγούν σε καλά αποτελέσματα.

Τα παιχνίδια αποτελούν έναν από τους σημαντικότερους τρόπους ψυχαγωγίας των ανθρώπων. Υπάρχουν παιχνίδια απλά και εύκολα, υπάρχουν, όμως, παιχνίδια περίπλοκα και δύσκολα, με τα τελευταία να είναι πιο ελκυστικά, καθώς απαιτούν υψηλές δεξιότητες για να αντιμετωπίσει κανείς την πολυπλοκότητά τους. Οι περισσότεροι άνθρωποι έχουν την τάση να εξελίσσουν τις ικανότητες και την ευφυΐα τους σε κάποιο δύσκολο και περίπλοκο παιχνίδι παίζοντας διαδοχικές παρτίδες με διαφορετικούς αντιπάλους. Στα πλαίσια, λοιπόν, της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης, τα παιχνίδια αποτελούν σημαντικό τομέα εφαρμογής Ενισχυτικής Μάθησης, διότι προσφέρουν τη δυνατότητα σε έναν ευφυή πράκτορα να βελτιώνει το επίπεδό του μέσω δοκιμής και αποτυχίας, όπως ακριβώς και οι άνθρωποι.

1.1 Περιγραφή και Συνεισφορά Εργασίας

Ένα παιχνίδι δύσκολο και περίπλοκο είναι το επιτραπέζιο παιχνίδι Turning Points, το οποίο παίζεται από δύο (ή περισσότερους παίκτες) σε ένα τετραγωνικό board, όπου τοποθετούν βέλη προς τις τέσσερις βασικές κατευθύνσεις τα οποία μπορούν κατ΄ επέκταση να περιστρέψουν τα ήδη υπάρχοντα βέλη. Στόχος του κάθε παίκτη είναι να στρέψει τα περισσότερα βέλη προς

την πλευρά του. Το παιχνίδι απαιτεί σημαντική ικανότητα αντίληψης του αποτελέσματος που επιφέρει κάθε ενέργεια βραχυπρόθεσμα και μακροπρόθεσμα, λόγος που οδήγησε στην επιλογή του για την εργασία αυτή.

Σκοπός της εργασίας είναι η δημιουργία και η εκπαίδευση ενός πράκτορα που θα παίζει το παιχνίδι Turning Points όσο το δυνατόν καλύτερα. Πρόκειται για τον πρώτο πράκτορα που υλοποιείται για το παιχνίδι αυτό. Η προσέγγισή μας για την δημιουργία του πράκτορα αυτού, είναι να συνδυάσουμε διάφορες τεχνικές και αλγορίθμους με σκοπό την εξαγωγή ενός αποτελεσματικού παίκτη. Χρησιμοποιούμε δέντρο αναζήτησης για την συστηματική ανάπτυξη του παιχνιδιού και οι κινήσεις επιλέγονται με βάση τον αλγόριθμο αναζήτησης α–β Pruning. Τα κριτήρια επιλογής της κατάλληλης κίνησης παρέχονται από τη συνάρτηση αξιολόγησης, η οποία κάνει μια εκτίμηση της μακροπρόθεσμης εξέλιξης του παιχνιδιού από οποιαδήποτε πιθανή κατάσταση. Τη διαδικασία της εκμάθησης του πράκτορα αναλαμβάνει ο αλγόριθμος TD-Learning, ο οποίος "μαθαίνει" τις τιμές των παραμέτρων της συνάρτησης αξιολόγησης ώστε να κάνει καλύτερες προβλέψεις.

Οι παίκτες που δημιουργήθηκαν ως στιγμιότυπα του παραπάνω πράκτορα είναι πολύ αποτελεσματικοί και επιδεικνύουν πολύ καλή απόδοση ενάντια στον άνθρωπο. Στα πλαίσια της εργασίας, για να υπάρχει η δυνατότητα της άμεσης αξιολόγησης οποιουδήποτε παίκτη, υλοποιήθηκε και ένα διαδραστικό γραφικό περιβάλλον απεικόνισης του παιχνιδιού, όπου ο άνθρωπος-χρήστης έχει τη δυνατότητα να παίξει παρτίδες και να αντιμετωπίσει διάφορους αντιπάλους (άνθρωπο ή υπολογιστή).

1.2 Επισκόπηση Εργασίας

Στο Κεφάλαιο 2 διατυπώνεται εν συντομία το θεωρητικό υπόβαθρο της εργασίας. Αναλύονται συνοπτικά η έννοια του πράκτορα, το δέντρο αναζήτησης, τεχνικές αναζήτησης, το δέντρο παιχνιδιού, οι αλγόριθμοι Minimax και α–β Pruning, η έννοια και η χρησιμότητα της συνάρτησης αξιολόγησης, καθώς και η διαδικασία ενισχυτικής μάθησης με τη χρήση του αλγορίθμου TD-Learning.

Στο Κεφάλαιο 3 γίνεται η πλήρης παρουσίαση του παιχνιδιού Turning Points και των κανόνων του, καθώς και η αναλυτική περιγραφή των στόχων της εργασίας. Επίσης, παρατίθενται κάποιες σχετικές εργασίες από την υπάρχουσα βιβλιογραφία.

Στο Κεφάλαιο 4 περιγράφεται η δική μας προσέγγιση στην σχεδίαση και υλοποίηση του πράκτορα για το παιχνίδι Turning Points. Αναλυτικότερα, εξετάζεται το δέντρο παιχνιδιού Minimax για το Turning Points, καθώς και η εφαρμογή του αλγορίθμου Minimax με α–β Pruning. Επίσης, αναλύεται η δομή της συνάρτηση αξιολόγησης και τα χαρακτηριστικά (features) που χρησιμοποιήθηκαν. Στη συνέχεια, εξετάζεται η εφαρμογή του αλγορίθμου TD-Learning και οι λεπτομέρειες της διαδικασίας της μάθησης.

Στο Κεφάλαιο 5 παρουσιάζεται βήμα-βήμα η υλοποίηση της εργασίας. Δίνονται στοιχεία για την γλώσσα προγραμματισμού που χρησιμοποιήθηκε, για τις βασικές συναρτήσεις και κλάσεις κάθε λειτουργίας, και, τέλος, για τη δημιουργία του γραφικού περιβάλλοντος.

Στο Κεφάλαιο 6 περιγράφεται η πειραματική διαδικασία που πραγματοποιήθηκε για να αξιολογηθεί η αποτελεσματικότητα της σχεδίασης του πράκτορα. Στη συνέχεια, παρουσιάζονται τα αποτελέσματα των πειραμάτων, οι διαφορετικοί παίκτες που προέκυψαν και η αξιολόγησή

τους μέσω της διεξαγωγής διαφόρων τουρνουά μεταξύ τους.

Στο Κεφάλαιο 7 γίνεται μία αποτίμηση της εργασίας και των αποτελεσμάτων που εξήχθησαν από αυτήν. Τέλος, αναφέρονται πιθανές μελλοντικές επεκτάσεις της εργασίας.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Πράκτορας

Πράκτορας [1, 2] είναι μία οντότητα η οποία παρατηρεί ένα περιβάλλον, λαμβάνει ερεθίσματα από αυτό και επενεργεί πάνω του, με σκοπό την επίτευξη κάποιων στόχων. Ένας πράκτορας πρέπει να λειτουργεί αυτόνομα και να είναι ευέλικτος, έτσι ώστε να μπορεί να προσαρμόζεται σε οποιαδήποτε αλλαγή. Επίσης, ένας πράκτορας μαθαίνει και διευρύνει τις γνώσεις του μέσα από τις πληροφορίες που συλλέγει.

Ορθολογικός ονομάζεται ο πράκτορας που ενεργεί με στόχο την επίτευξη του καλύτερου δυνατού αποτελέσματος. Ουσιαστικά, ορθολογικός πράκτορας είναι αυτός που ενεργεί αντικειμενικά σωστά (με βάση τη λογική).

2.2 Αναζήτηση

2.2.1 Πρόβλημα Αναζήτησης

Υπάρχουν δύο κατηγορίες πρακτόρων. Οι ανακλαστικοί και οι πράκτορες βασισμένοι στους στόχους. Οι πρώτοι λαμβάνουν ερεθίσματα και βάσει αυτών εκτελούν κάποιες ενέργειες. Οι δεύτεροι διατηρούν μία εσωτερική αναπαράσταση της κατάστασης του κόσμου και προσπαθούν να βρουν ενέργειες που θα τους οδηγήσουν σε κάποια κατάσταση όπου επιτυγχάνεται ο στόχος. Οι βασισμένοι στους στόχους πράκτορες, προσπαθούν να επιλύσουν ένα πρόβλημα με τον καλύτερο δυνατό τρόπο. Για να γίνει αυτό, διατυπώνουν το πρόβλημα με τα εξής στοιχεία [1, 2]:

- Αρχική κατάσταση προβλήματος αναζήτησης.
- Συνάρτηση διαδόχων καταστάσεων. Για οποιαδήποτε κατάσταση παράγει όλες τις πιθανές διάδοχες καταστάσεις με βάση όλες τις επιτρεπτές ενέργειες του πράκτορα στη συγκεκριμένη κατάσταση.
- Έλεγχος επίτευξης στόχου. Εξετάζει αν μία κατάσταση αποτελεί κατάσταση στόχου.
- Κόστος διαδρομής. Αντικατοπτρίζει την απόδοση του πράκτορα κατά μήκος μίας διαδρομής στο χώρο καταστάσεων.

2.2.2 Δέντρο Αναζήτησης

Η πιο διαδεδομένη μέθοδος επίλυσης προβλημάτων αναζήτησης είναι η συστηματική ανάπτυξη ενός δέντρου αναζήτησης [1, 2]. Το δέντρο αναζήτησης δημιουργείται από την αρχική κατάσταση και τη συνάρτηση διαδόχων. Η διαδικασία δημιουργίας του περιγράφεται παρακάτω.

Ξεκινώντας, ο αρχικός κόμβος (αρχική κατάσταση) εξετάζεται αν αποτελεί κατάσταση στόχου. Αν αποτελεί στόχο, τότε επιλύεται το πρόβλημα και η αναζήτηση τερματίζεται. Διαφορετικά, εφαρμόζεται η συνάρτηση διαδόχων και παράγεται ένα σύνολο νέων καταστάσεων, που αποτελούν την επέκταση του αρχικού κόμβου, βρίσκονται ένα επίπεδο βαθύτερα στο δέντρο και θεωρούνται παιδιά του. Αμέσως μετά, εξετάζεται ένας εκ των νέων κόμβων και η διαδικασία αυτή συνεχίζεται αναδρομικά επεκτείνοντας κόμβους και αναπτύσσοντας το δέντρο σε διαφορετικά σημεία. Το μέσο πλήθος των παιδιών ενός κόμβου ονομάζεται παράγοντας διακλάδωσης (branching factor) και καθορίζει το πόσο γρήγορα πυκνώνει το δέντρο ως προς το βάθος του. Πριν επεκταθεί κάποιος κόμβος, εξετάζεται αν αποτελεί κατάσταση στόχου ώστε να τερματιστεί εκεί η αναζήτηση. Οι επεκτάσεις συνεχίζονται μέχρι να βρεθεί κατάσταση στόχου ή να μην υπάρχει άλλος κόμβος προς επέκταση, που σημαίνει ότι όλο το δέντρο έχει αναπτυχθεί και η αναζήτηση ολοκληρώνεται ανεπιτυχώς.

Σημαντικό ρόλο στην παραπάνω διαδικασία παίζει η στρατηγική αναζήτησης, η οποία καθορίζει ποιος κόμβος από τα φύλλα του τρέχοντος δέντρου θα επεκταθεί σε κάθε βήμα.

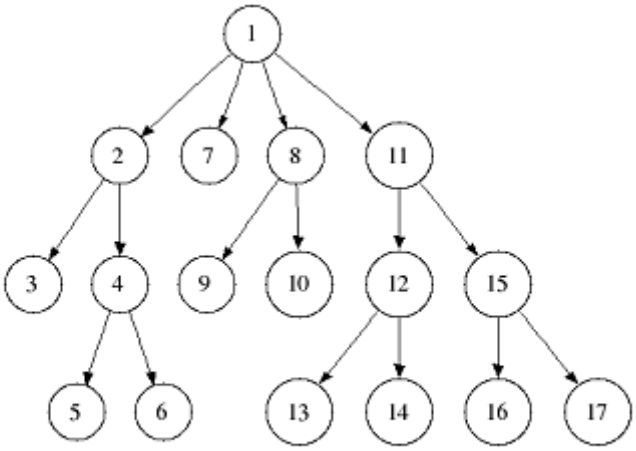
2.2.3 Αναζήτηση Πρώτα σε Βάθος

Η αναζήτηση πρώτα σε βάθος (Depth-First Search, DFS) [1, 2], είναι μία στρατηγική αναζήτησης κατά την οποία σε κάθε βήμα επεκτείνεται ο κόμβος που βρίσκεται βαθύτερα στο τρέχον δέντρο αναζήτησης.

Η στρατηγική αυτή αρχικά επιλέγει τον αρχικό κόμβο. Εφόσον επεκταθεί, δημιουργούνται όλα τα παιδιά του. Στη συνέχεια, η στρατηγική αυτή θα προχωρήσει στο πρώτο παιδί του. Εφόσον επεκταθεί και αυτό, θα δημιουργήθουν όλα τα παιδιά του και στη συνέχεια θα επιλεγεί το πρώτο από αυτά, εφόσον βρίσκεται στο μεγαλύτερο βάθος. Η στρατηγική συνεχίζει να επιλέγει κόμβους με τον ίδιο τρόπο σε ολοένα αυξανόμενο βάθος, μέχρι να βρεθεί κατάσταση στόχου ή να φτάσει σε τερματικό κόμβο του δέντρου (φύλλο), όπου επιστρέφει στο προηγούμενο επίπεδο και επεκτείνει τον επόμενο βαθύτερο ανεξερεύνητο κόμβο κ.ο.κ.

Η αναζήτηση πρώτα σε βάθος είναι παρόμοια με τη λειτουργία μίας στοίβας (Last In First Out, LIFO). Οι απαιτήσεις μνήμης για αυτήν την στρατηγική δεν είναι ιδιαίτερα μεγάλες, εφόσον χρειάζεται να αποθηκεύει τους ανεξερεύνητους κόμβους που γειτνιάζουν με την τρέχουσα διαδρομή από τον αρχικό κόμβο μέχρι κάποιο φύλλο του δέντρου.

Ένα παράδειγμα αναζήτησης πρώτα σε βάθος φαίνεται στο Σχήμα 2.1, όπου κάθε κόμβος φέρει τον αριθμό που υποδεικνύει τη σειρά επέκτασής του. Πρώτος θα επεκταθεί ο κόμβος 1 (αρχικός κόμβος). Τα παιδιά του είναι οι κόμβοι 2, 7, 8 και 11. Ο επόμενος κόμβος που θα επεκταθεί είναι ο 2 (το πρώτο παιδί). Αυτός με τη σειρά του επεκτείνεται και παράγει τους κόμβους 3 και 4. Σειρά να επεκταθεί παίρνει ο κόμβος 3, ο οποίος, όμως, αποτελεί φύλλο του δέντρου με συνέπεια να μην μπορεί να επεκταθεί περαιτέρω. Επιστρέφοντας, λοιπόν, στον κόμβο 2, επεκτείνεται το επόμενο παιδί του, δηλαδή ο κόμβος 4. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο μέχρι να βρεθεί κατάσταση στόχου ή να μην υπάρχει ανεξερεύνητος κόμβος.



Σχήμα 2.1: *Σειρά Επέκτασης Κόμβων στην Αναζήτηση Πρώτα σε Βάθος.*

2.2.4 Αναζήτηση με Υπαναχώρηση

Η αναζήτηση με υπαναχώρηση (Backtracking Search) [1, 2] είναι μια παραλλαγή της αναζήτησης πρώτα σε βάθος. Η διαφορά τους είναι ότι κατά την επέκταση ενός κόμβου, η συνάρτηση διαδόχων αρχικά δημιουργεί μόνο το πρώτο παιδί του. Όταν η εκτέλεση της αναζήτησης επιστρέψει ανεπιτυχώς στον κόμβο-πατέρα, ο κόμβος αυτός "θυμάται" ποιο είναι το επόμενο παιδί που πρέπει να παραχθεί, το δημιουργεί και η αναζήτηση συνεχίζεται από αυτό. Συνεπώς, τα παδιά ενός κόμβου δε δημιουργούνται μαζικά, αλλά διαδοχικά όταν έρθει η σειρά του καθενός.

Μία σημαντική λεπτομέρεια για αυτήν την στρατηγική, είναι ότι υπάρχει η δυνατότητα να παραχθεί ο κόμβος-παιδί απλά εφαρμόζοντας την αντίστοιχη ενέργεια ως άμεση τροποποίηση της κατάστασης του κόμβου-πατέρα. Αυτή η διευκόλυνση, ωστόσο, προϋποθέτει να υπάρχει και η δυνατότητα αναίρεσης μίας τέτοιας τροποποίησης, ετσί ώστε όταν έρθει η ώρα η αναζήτηση να υπαναχωρήσει από το παιδί στον πατέρα να επανέλθει η σωστή κατάσταση.

Κατά συνέπεια, το μόνο που χρειάζεται να διατηρείται στη μνήμη κατά την αναζήτηση με υπαναχώρηση είναι η κατάσταση του τρέχοντος κόμβου, η οποία συνεχώς τροποποιείται στη διάρκεια της αναζήτησης. Διαπιστώνεται, λοιπόν, ότι αυτή η στρατηγική αποφέρει σημαντική εξοικοινόμηση μνήμης (και χρόνου).

2.2.5 Αναζήτηση Περιορισμένου Βάθους

Η αναζήτηση πρώτα σε βάθος, αλλά και η αναζήτηση με υπαναχώρηση, αντιμετωπίζουν πολλές δυσκολίες όταν ένα δέντρο έχει μεγάλο έως και άπειρο βάθος, το οποίο συνήθως οφείλεται σε ατέρμονες διαδρομές στο δέντρο. Μπορεί, δηλαδή, η αναζήτηση να εγκλωβιστεί σε ένα τμήμα του δέντρου που έχει πολύ μεγάλο βάθος και να μην εξερευνήσει καθόλου το υπόλοιπο δέντρο. Οπότε, δεν μπορεί να εγγυηθεί την πληρότητα της αναζήτησης και την εύρεση κατάστασης στόχου σε λογικά χρονικά πλαίσια.

Το πρόβλημα αυτό αντιμετωπίζεται με την αναζήτηση περιορισμένου βάθους [1, 2]. Σε αυτήν την στρατηγική το δέντρο αναζήτησης επεκτείνεται μέχρι κάποιο προκαθορισμένο βάθος

και υπαναχωρεί άμεσα μόλις φτάσει σε αυτό (αποκοπή), με σκοπό την αποφυγή όσων αναφέρθηκαν παραπάνω. Ωστόσο, αυτή η στρατηγική μπορεί να αποδειχθεί προβληματική στην πράξη, αφού οι πιθανές καταστάσεις στόχου μπορεί να βρίσκονται σε βάθος μεγαλύτερο από το όριο επέκτασης. Για το λόγο αυτό, σε κάθε κόμβο όπου γίνεται αποκοπή της αναζήτησης χρειάζεται να εφαρμοστεί μία συνάρτηση αξιολόγησης, η οποία κρίνει κατά πόσο ένας τέτοιος κόμβος (που δεν αντιστοιχεί σε κατάσταση-στόχου) μπορεί να μας οδηγήσει με βαθύτερη αναζήτηση σε μία κατάσταση στόχου. Περισσότερα για τη συνάρτηση αξιολόγησης παρουσιάζονται σε σχετική ενότητα παρακάτω.

2.3 Αναζήτηση σε Παιχνίδια

Μία μεγάλη κατηγορία προβλημάτων αναζήτησης αποτελούν και τα ανταγωνιστικά παιχνίδια, για τα οποία υπάρχουν ακριβείς περιγραφές και σαφείς στόχοι. Σε τέτοια παιχνίδια ο πράκτορας καλείται να αντιμετωπίσει έναν ή περισσότερους αντιπάλους με αντιδιαμετρικούς στόχους. Τα παιχνίδια αυτά μπορούν να χαρακτηριστούν και ως προβλήματα αναζήτησης με αντιπαλότητα.

2.3.1 Δέντρο Παιχνιδιού

Ένα παιχνίδι ενός παίκτη μπορεί εύκολα να αναπαρασταθεί με ένα δέντρο αναζήτησης, το οποίο ονομάζεται δέντρο παιχνιδιού [1, 2]. Στο δέντρο αυτό κάθε κόμβος αντιστοιχεί σε μία πιθανή κατάσταση του παιχνιδιού. Η αρχική κατάσταση του παιχνιδιού είναι στη ρίζα του δέντρου. Η συνάρτηση διαδόχων παράγει τα παιδιά ενός κόμβου λαμβάνοντας υπ' όψιν όλες τις πιθανές κινήσεις που μπορεί ο παίκτης να εκτελέσει (κάθε παιδί αντιστοιχεί σε μία κίνηση του παίκτη). Κάθε διαδρομή που οδηγεί σε μία κατάσταση, αντιστοιχεί σε μία ακολουθία κινήσεων στο παιχνίδι. Τέλος, το κόστος διαδρομής είναι οι απολαβές του παίκτη βάσει των κανόνων του παιχνιδιού κατά μήκος της διαδρομής.

Έτσι, λοιπόν, δημιουργείται το δέντρο παιχνιδιού, στο οποίο μπορεί να εφαρμοστεί αναζήτηση. Σκοπός της αναζήτησης είναι να βρεθεί κάποια τερματική κατάσταση του παιχνιδιού (κατάσταση στόχου) στην οποία ο παίκτης μεγιστοποιεί τις απολαβές του.

Τι γίνεται, όμως, σε ένα παιχνίδι με δύο παίκτες;

2.3.2 Minimax

Στα παιχνίδια δύο παικτών, όπου οι παίκτες παίζουν εναλλάξ, πρέπει να λαμβάνονται υπ' όψιν και οι (πιθανές) επιλογές του αντιπάλου. Σε αντίθεση με τα παιχνίδια ενός παίκτη, στα ανταγωνιστικά παιχνίδια τα συμφέροντα ενός παίκτη κατά κανόνα συγκρούονται με τα συμφέροντα του αντιπάλου του.

Η ανταγωνιστικότητα ενός τέτοιου παιχνιδιού δεν επιτρέπει σε έναν παίκτη να ελέγξει το παιχνίδι όπως τον συμφέρει. Συνεπώς, πρέπει να επιλέγει κάθε κίνησή του με μεγάλη προσοχή και υπολογίζοντας τον αντίπαλό του. Σκοπός κάθε παίκτη δεν είναι απλά να εντοπίσει καταστάσεις όπου κερδίζει το παιχνίδι, αλλά να βεβαιωθεί ότι δεν θα παρεμποδιστεί από τον αντίπαλό του, αν αυτό είναι εφικτό. Αυτό υποδεικνύει ότι δεν ζητείται απλά μία επιθυμητή

διαδρομή, αλλά η καλύτερη δυνατή διαδρομή. Πρέπει, λοιπόν, να εξερευνηθεί κάθε κόμβος του δέντρου του παιχνιδιού.

Το πρόβλημα είναι ότι ο κάθε παίκτης δεν γνωρίζει τη στρατηγική που ακολουθεί ο αντίπαλός του. Οπότε, προσπαθεί να μελετήσει τις πιθανές κινήσεις του αντιπάλου του, με σκοπό να επιλέξει τις καλύτερες δυνατές κινήσεις για τον εαυτό του. Στην πράξη κάτι τέτοιο αποδεικνύεται δύσκολο, καθώς πρέπει να εξεταστούν ολόκληρες ακολουθίες κινήσεων και όχι μεμονωμένες κινήσεις.

Λύση στο παραπάνω πρόβλημα δίνει ο αλγόριθμος Minimax [1, 2, 3]. Σύμφωνα με τον αλγόριθμο αυτό, ο ένας παίκτης ονομάζεται Max και ο άλλος Min. Ο Max προσπαθεί να μεγιστοποιήσει τις απολαβές του, ενώ ο Min προσπαθεί να ελαχιστοποιήσει τις απολαβές του Max, ώστε ισοδύναμα να μεγιστοποιήσει τις δικές του. Με αυτόν τον τρόπο προσπαθεί ο αλγόριθμος να μοντελοποιήσει την συμπεριφορά των αντιπάλων. Δεδομένου ότι οι δύο παίκτες έχουν συγκρουόμενα συμφέροντα, η μοντελοποίηση αυτή υπονοεί ότι οποιαδήποτε απολαβή δίνεται σε κάποιον παίκτη αφαιρείται από τον άλλο. Το πλεονέκτημα μίας τέτοιας μοντελοποίησης είναι ότι το αποτέλεσμα του παιχνιδιού μπορεί να αναπαρασταθεί με μία μόνο τιμή (απολαβές του Max—οι απολαβές του Min είναι συμμετρικές). Τέτοια παιχνίδια είναι γνωστά ως παιχνίδια μηδενικού αθροίσματος (zero-sum games) και σε αυτά βρίσκει εφαρμογή ο αλγόριθμος Minimax.

Ο αλγόριθμος Minimax βασίζεται στη αναζήτηση με υπαναχώρηση. Το δέντρο του παιχνιδιού κατασκευάζεται με τον ίδιο τρόπο που αναφέρθηκε παραπάνω για τα παιχνίδια ενός παίκτη. Η μόνη διαφορά είναι ότι κάθε επίπεδο του δέντρου αντιστοιχεί εκ περιτροπής σε κίνηση του ενός ή του άλλου παίκτη, με τον Max να έχει τον πρώτο λόγο στη ρίζα του δέντρου. Κάθε φύλλο του δέντρου αντιστοιχεί σε μία τερματική κατάσταση του παιχνιδιού, η οποία προκύπτει από μία συγκεκριμένη ακολουθία κινήσεων των δύο παικτών. Σε κάθε φύλλο μπορεί να υπολογιστεί με ακρίβεια το αποτέλεσμα (τιμή) του παιχνιδιού. Η απόδοση τιμές στους ενδιάμεσους κόμβους γίνεται από κάτω προς τα πάνω (από τα φύλλα προς τη ρίζα). Κάποιες από τις τιμές στα φύλλα επιστρέφουν αναδρομικά στα υψηλότερα επίπεδα του δέντρου. Όμως, η επιλογή της τιμής που θα επιστραφεί σε κάθε κόμβο καθορίζεται από τον παίκτη που παίζει στο αντίστοιχο επίπεδο του δέντρου.

Σύμφωνα, λοιπόν, με τον αλγόριθμο Minimax, διακρίνονται δύο περιπτώσεις, μία για κάθε παίκτη. Στην περίπτωση που παίζει ο Max, επιλέγεται από τις τιμές των παιδιών του, η μέγιστη τιμή. Στην περίπτωση που παίζει ο Min, επιλέγεται από τις τιμές των παιδιών του, η ελάχιστη τιμή. Έτσι, όταν προσομοιώνεται κίνηση του Max γίνεται η καλύτερη δυνατή επιλογή για τον Max, ενώ όταν προσομοιώνεται κίνηση του Min γίνεται η καλύτερη δυνατή επιλογή για τον Min. Με αυτόν τον τρόπο ο Max προσομοιώνει το καλύτερο σενάριο ως προς τις επιλογές του, σε συνδυασμό με το χειρότερο σενάριο για τον ίδιο ως προς τις επιλογές του αντιπάλου. Αυτό αποσκοπεί στην εύρεση μίας στρατηγικής, η οποία θα είναι ασφαλής για τον ίδιο ακόμη και ενάντια σε έναν βέλτιστο αντίπαλο. Η παραπάνω διαδικασία θα διαδώσει μία μοναδική τιμή στη ρίζα του δέντρου, η οποία υποδηλώνει ότι υπάρχει βέλτιστη ακολουθία κινήσεων του Max που οδηγεί σε αυτό το αποτέλεσμα, ακόμη και εάν ο αντίπαλος απαντήσει βέλτιστα. Η βέλτιστη κίνηση του Max στη ρίζα αντιστοιχεί στο παιδί με τη μέγιστη τιμή (αυτή που προωθήθηκε στη ρίζα). Η αναζήτηση Minimax φαίνεται στον Αλγόριθμο 2.1. Κάθε φορά που ένας παίκτης καλείται να επιλέξει κίνηση, εκτελεί τον αλγόριθμο αυτόν με αρχική κατάσταση

Αλγοριθμος 2.1: *Minimax.*

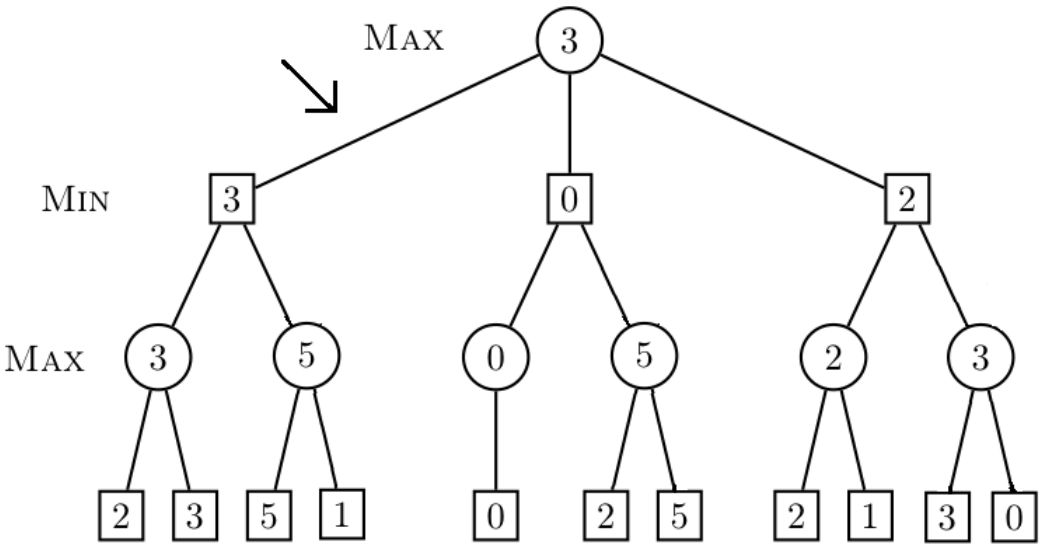
```
1: function: Max(node)
2: if node is terminal node then
3:     return Value(node)
4: else
5:     maxValue = −∞
6:     for each child of node do
7:         maxValue = max{maxValue, Min(child)}
8:     end for
9:     return maxValue
10: end if
11:
12:
13: function: Min(node)
14: if node is terminal node then
15:     return Value(node)
16: else
17:     minValue = +∞
18:     for each child of node do
19:         minValue = min{minValue, Max(child)}
20:     end for
21:     return minValue
22: end if
23:
24:
25: Initial Call: Max(root)
```

την τρέχουσα κατάσταση του παιχνιδιού. Οπότε, η επιλογή της βέλτιστης κίνησης βάσει της στρατηγικής Minimax προκύπτει από την πλήρη ανάπτυξη του δέντρου του παιχνιδιού και τον υπολογισμό της τιμής του παιχνιδιού στη ρίζα.

Ένα παράδειγμα αναζήτησης Minimax φαίνεται στο Σχήμα 2.2. Το παράδειγμα αυτό ξεκινάει με κίνηση του Max, μετά παίζει ο Min και στο τέλος ξανά ο Max, δηλαδή αυτό το ιδεατό παιχνίδι τελειώνει μετά από τρεις κινήσεις. Ο αλγόριθμος Minimax, όπως προαναφέρθηκε, αποδίδει τις κατάλληλες τιμές στους κόμβους από κάτω προς τα πάνω. Στο τελευταίο επίπεδο αποδίδονται στους κόμβους οι τιμές που προκύπτουν από τους κανόνες του παιχνιδιού, σύμφωνα με την τερματική κατάσταση του παιχνιδιού που αντικατοπτρίζει κάθε κόμβος. Στο αμέσως υψηλότερο επίπεδο, στο οποίο παίζει ο Max, κάθε κόμβος επιλέγει τη μεγαλύτερη τιμή από τις τιμές των παιδιών του. Ανεβαίνοντας ένα επίπεδο ακόμη, παίζει ο παίκτης Min και κάθε κόμβος επιλέγει τη μικρότερη τιμή από τις τιμές των παιδιών του. Τέλος, φτάνοντας στη ρίζα του δέντρου, όπου παίζει ξανά ο Max, επιλέγεται ως τελική κίνηση εκείνη που οδηγεί στο παιδί με την μεγαλύτερη τιμή. Η κίνηση αυτή σημειώνεται στο σχήμα.

2.3.3 Minimax με α-β Pruning

Η πολυπλοκότητα του αλγορίθμου Minimax αυξάνεται εκθετικά όσο περισσότερες είναι οι πιθανές κινήσεις κάθε παίκτη στη διάρκεια του παιχνιδιού. Αυτό συμβαίνει διότι δημιουργούνται



Σχήμα 2.2: *Παράδειγμα Αναζήτησης Minimax.*

εκθετικά περισσότερες πιθανές καταστάσεις, με συνέπεια το δέντρο να γίνεται ολοένα και πιο πυκνό σε μεγάλο βάθος. Έτσι, η εύρεση της βέλτιστης κίνησης είναι ιδιαίτερα χρονοβόρα διαδικασία, καθώς πρέπει να γίνει διαπέραση ολόκληρου του δέντρου.

Για να βελτιωθούν τα παραπάνω προβλήματα, χρειάζεται να μειωθούν οι κόμβοι του δέντρου. Κάποιοι από τους κόμβους, ακόμη και ολόκληρα υποδέντρα, δεν επηρεάζουν την τελική απόφαση του Minimax στη ρίζα. Η σκέψη, λοιπόν, είναι πώς τέτοιοι κόμβοι μπορεί να αποκοπούν, εξοικονομώντας πολύτιμο χρόνο.

Λύση έρχεται να δώσει η τεχνική α–β Pruning (Κλάδεμα α - β) [1, 2, 4]. Η τεχνική αυτή δίνει ακριβώς τα ίδια αποτελέσματα με τον αλγόριθμο Minimax, με μόνη διαφορά ότι χρειάζεται λιγότερο χρόνο. Συνεπώς, ο αλγόριθμος Minimax με α–β Pruning αποτελεί βελτίωση του αλγορίθμου Minimax.

Ο αλγόριθμος Minimax με α–β Pruning χρησιμοποιώντας τα όρια α και β ανακαλύπτει ποιοι κόμβοι δεν επηρεάζουν την αναζήτηση και τους αποκόπτει αντί να τους επεκτείνει. Έτσι, αποφεύγεται η άσκοπη εξερεύνηση κόμβων ή ολόκληρων υποδέντρων. Το όριο α είναι η καλύτερη τιμή για τον Max στο τρέχον σημείο του δέντρου και το όριο β η καλύτερη τιμή για τον Min στο τρέχον σημείο του δέντρου. Σε κάθε κόμβο που παίζει ο Max ανανεώνεται η τιμή του α (μόνο προς τα πάνω), ενώ σε κάθε κόμβο που παίζει ο Min ανανεώνεται η τιμή του β (μόνο προς τα κάτω). Όσον αφορά το κλάδεμα, οι κόμβοι Max κλαδεύουν με βάση το όριο β, ενώ οι κόμβοι Min με βάση το όριο α. Σε κάθε κόμβο εξετάζεται εάν το α ξεπέρασε το β. Εάν το έχει ξεπεράσει τότε ο κόμβος αποκόπτεται, όπως και κάθε υποδέντρο που θα επεκτεινόταν από αυτόν. Η αναζήτηση Minimax με α–β Pruning φαίνεται στο Αλγόριθμο 2.2.

Αναλυτικότερα, στην περίπτωση που η εκτέλεση βρίσκεται σε Max κόμβο και το α ξεπεράσει το β σημαίνει πως ο παίκτης Min σε υψηλότερο επίπεδο έχει επιλογή που του εγγυάται καλύτερο αποτέλεσμα β (αφού είναι μικρότερη τιμή), οπότε δεν χρειάζεται να συνεχιστεί η εξερεύνηση του κόμβου Max λόγω του ότι ο Min δεν θα επιτρέψει ποτέ την ακολουθία κινήσεων που οδηγεί σε αυτόν τον κόμβο, ο οποίος τελικά πρόκειται να πάρει τιμή μεγαλύτερη

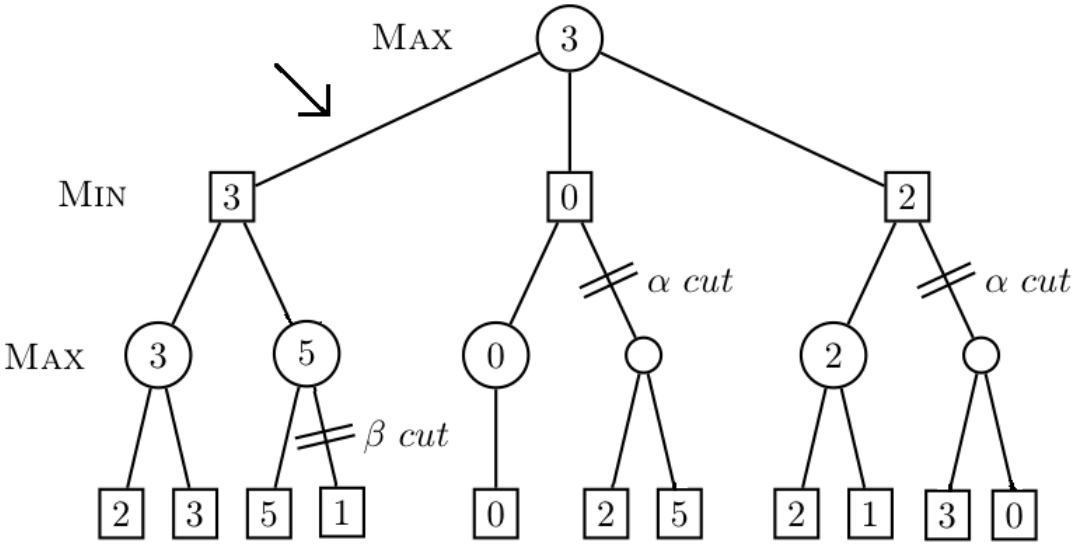
Αλγοριθμος 2.2: *Minimax με α–β Pruning.*

```
1: function: Max(node, α, β)
2: if node is terminal node then
3:     return Value(node)
4: else
5:     maxValue = −∞
6:     for each child of node do
7:         maxValue = max{maxValue, Min(child, α, β)}
8:         α = max{α, maxValue}
9:         if β ≤ α then
10:            break
11:        end if
12:    end for
13:    return maxValue
14: end if
15:
16:
17: function: Min(node, α, β)
18: if node is terminal node then
19:    return Value(node)
20: else
21:    minValue = +∞
22:    for each child of node do
23:        minValue = min{minValue, Max(child, α, β)}
24:        β = min{β, minValue}
25:        if β ≤ α then
26:            break
27:        end if
28:    end for
29:    return minValue
30: end if
31:
32:
33: Initial Call: Max(root, −∞, +∞)
```

ή ίση του α.

Αντιστρόφως, στην περίπτωση που η εκτέλεση βρίσκεται σε Min κόμβο και το β γίνει μικρότερο του α σημαίνει πως ο παίκτης Max σε υψηλότερο επίπεδο έχει επιλογή που του εγγυάται καλύτερο αποτέλεσμα α (αφού είναι μεγαλύτερη τιμή), οπότε δεν χρειάζεται να συνεχιστεί η εξερεύνηση του κόμβου Min λόγω του ότι ο Max δεν θα επιτρέψει ποτέ την ακολουθία κινήσεων που οδηγεί σε αυτόν τον κόμβο, ο οποίος τελικά πρόκειται να πάρει τιμή μικρότερη ή ίση του β.

Στο Σχήμα 2.3 φαίνεται το αποτέλεσμα μετά την εφαρμογή του Minimax με α–β Pruning στο παράδειγμα αναζήτησης με Minimax του Σχήματος 2.2. Οι επτά κόμβοι που αποκόπτονται είναι περιττοί για την εύρεση της βέλτιστης κίνησης. Επισημαίνεται ότι η τελική τιμή στη ρίζα ταυτίζεται με αυτήν του αλγορίθμου Minimax.



Σχήμα 2.3: *Παράδειγμα Αναζήτησης Minimax με α–β Pruning.*

2.4 Συνάρτηση Αξιολόγησης

Όπως αναφέρθηκε σε προηγούμενες ενότητες, είναι πολύ δύσκολο να εξερευνηθεί ένα μεγάλου μεγέθους δέντρο. Ειδικότερα σε ένα δέντρο παιχνιδιού, όπου πρέπει να εξερευνηθεί πλήρως και να φτάσει η αναζήτηση σε όλες τις τερματικές καταστάσεις, η χρονική πολυπλοκότητα είναι πολύ μεγάλη. Λύση σε αυτό το πρόβλημα δίνεται με την αναζήτηση περιορισμένου βάθους.

Στα παιχνίδια, όμως, όπου εφαρμόζεται Minimax με α–β Pruning, εάν τεθούν όρια επέκτασης μπορεί στα σημεία αποκοπής του δέντρου να μην υπάρχουν τερματικές καταστάσεις, αλλά ενδιάμεσες καταστάσεις του παιχνιδιού. Σε αυτήν την περίπτωση, λοιπόν, δεν μπορεί να αποδοθεί τιμή, καθώς δεν υπάρχει πληροφορία για το αποτέλεσμα του παιχνιδιού. Το καλύτερο που μπορεί να γίνει σε έναν τέτοιο ενδιάμεσο κόμβο είναι να εκτιμηθεί το αποτέλεσμα του παιχνιδιού από την αντίστοιχη κατάσταση. Συνεπώς, γίνεται κανονικά η αναζήτηση Minimax με α–β Pruning σε περιορισμένο βάθος, ωστόσο στις ενδιάμεσες καταστάσεις των σημείων αποκοπής εφαρμόζεται μία ευριστική συνάρτηση αξιολόγησης [1, 5], η οποία θα αποδώσει κατάλληλη τιμή στον κόμβο ανάλογη με την εκτίμηση της ποιότητας της αντίστοιχης κατάστασης.

Η συνάρτηση αξιολόγησης έχει ως στόχο να αξιολογεί τις ενδιάμεσες καταστάσεις ενός παιχνιδιού, ώστε να μπορεί να αποφανθεί ποιες από αυτές μπορούν να οδηγήσουν τον παίκτη σε επιθυμητό αποτέλεσμα και αντίστροφα ποιες μπορούν να τον οδηγήσουν σε μη επιθυμητό αποτέλεσμα. Ωστόσο, η συνάρτηση αξιολόγηγης δεν μπορεί να εγγυηθεί συγκεκριμένο αποτέλεσμα, αφού υπάρχει έλλειψη πληροφορίας για την εξέλιξη του παιχνιδιού. Για το λόγο αυτό, χρειάζεται ιδιαίτερη προσοχή στη σχεδίαση της συνάρτησης αξιολόγησης, ώστε να οδηγεί τον παίκτη, μέσω της αναζήτησης, σε όσο το δυνατόν καλύτερα μονοπάτια του δέντρου.

Ένα κλασικός τρόπος σχεδίασης συνάρτησης αξιολόγησης βασίζεται σε δύο σημαντικά στοιχεία. Αρχικά, πρέπει να γίνει επιλογή των κατάλληλων (αριθμητικών) χαρακτηριστικών

(features) βάσει των οποίων μπορεί να εκτιμηθεί η ποιότητα μίας κατάστασης. Από μόνα τους, όμως, τα χαρακτηριστικά αυτά δεν αρκούν. Πρέπει να δωθεί η κατάλληλη σημασία στο κάθε χαρακτηριστικό, ώστε η συνάρτηση αξιολόγησης να είναι αποτελεσματική στις εκτιμήσεις της. Για να επιτευχθεί αυτό, τα χαρακτηριστικά συνδυάζονται αθροιστικά με διαφορετικό βάρος το καθένα. Με τον τρόπο αυτό σταθμίζεται κάθε χαρακτηριστικό στο βαθμό που του ορίζει το βάρος του (θετικό ή αρνητικό).

Ορίζοντας, λοιπόν, τη συνάρτηση αξιολόγησης, πρόκειται για μία σταθμισμένη γραμμική (ως προς τα βάρη) συνάρτηση που έχει την εξής μορφή:

$$Value(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s),$$

όπου s η κατάσταση του παιχνιδιού που αξιολογείται,
$Value(s)$ η τιμή αξιολόγησης της κατάστασης s,
n το πλήθος των χαρακτηριστικών,
f_i η τιμή του χαρακτηριστικού i και
w_i το βάρος του χαρακτηριστικού i.

Πρέπει να σημειωθεί ότι η επιλογή των χαρακτηριστικών και των βαρών τους απαιτούν μεγάλη προσοχή, ώστε η συνάρτηση αξιολόγησης να είναι αποτελεσματική (να οδηγεί σε καλές επιλογές κινήσεων) και αποδοτική (να υπολογίζεται γρήγορα). Επίσης, εάν η αναζήτηση φτάσει σε τερματικές καταστάσεις, τότε η συνάρτηση αξιολόγησης αποδίδει τιμές στις καταστάσεις με βάση το αποτέλεσμα του παιχνιδιού σε αυτές, αγνοώντας τα χαρακτηριστικά και τα βάρη τους.

Εφόσον έχει οριστεί η συνάρτηση αξιολόγησης, η αναζήτηση Minimax με α–β Pruning σε περιορισμένο βάθος παίρνει τη μορφή που φαίνεται στον Αλγόριθμο 2.3.

2.5 Μάθηση

Όταν ένας πράκτορας σχεδιάζεται με σκοπό την επίλυση κάποιων προβλημάτων, δεν είναι δεδομένη εξαρχής η ικανότητα και η αποτελεσματικότητά του. Σε κάποια απλά περιβάλλοντα είναι εύκολο να σχεδιαστεί αποτελεσματικός πράκτορας, όμως, σε δυσκολότερα περιβάλλοντα κάτι τέτοιο δεν είναι εύκολο.

Η δυνατότητα βελτίωσης, λοιπόν, ενός πράκτορα είναι ένα από τα πιο σημαντικά στοιχεία του. Για να επιτευχθεί η βελτίωση αυτή, είναι απαραίτητη κάποια μέθοδος μάθησης. Μάθηση, σύμφωνα με τις επιταγές της μηχανικής μάθησης, είναι η ικανότητα ενός πράκτορα να εκμεταλλεύεται τις γνώσεις που έχει, τις παρατηρήσεις από το περιβάλλον του και τις επιδράσεις του σε αυτό, με σκοπό τη βελτίωση των ικανοτήτων του. Το ερευνητικό πειδίο της μηχανικής μάθησης διακρίνει πολλές μορφές μάθησης, όπως επιβλεπόμενη, μη επιβλεπόμενη και ενισχυτική μάθηση. Η πλέον κατάλληλη για τους σκοπούς της εργασίας είναι η ενισχυτική μάθηση.

Αλγοριθμος 2.3: *Minimax με α–β Pruning σε Περιορισμένο Βάθος.*

```
1: function: Max(node, α, β, depth)
2: if node is terminal node or depth == 0 then
3:     return Value(node)
4: else
5:     maxValue = −∞
6:     for each child of node do
7:         maxValue = max{maxValue, Min(child, α, β, depth − 1)}
8:         α = max{α, maxValue}
9:         if β ≤ α then
10:            break
11:        end if
12:    end for
13:    return maxValue
14: end if
15:
16:
17: function: Min(node, α, β, depth)
18: if node is terminal node or depth == 0 then
19:     return Value(node)
20: else
21:     minValue = +∞
22:     for each child of node do
23:         minValue = min{minValue, Max(child, α, β, depth − 1)}
24:         β = min{β, minValue}
25:         if β ≤ α then
26:             break
27:         end if
28:     end for
29:     return minValue
30: end if
31:
32:
33: Initial Call: Max(root, −∞, +∞, cut-off-depth)
```

2.5.1 Ενισχυτική Μάθηση

Ενισχυτική μάθηση [6, 5, 1] είναι μία μορφή μάθησης, κατά την οποία ένας πράκτορας προσπαθεί να βελτιωθεί μέσα από τις αλληλεπιδράσεις του με το περιβάλλον μέσω δοκιμής και αποτυχίας (trial and error). Χρησιμοποιείται σε πολλές εφαρμογές, όπως στον έλεγχο κίνησης ρομπότ, στη βελτιστοποίηση εργασιών σε γραμμές παραγωγής, στην εκμάθηση παιχνιδιών κ.α.

Πιο συγκεκριμένα, στα πλαίσια των παιχνιδιών που μας ενδιαφέρουν, η ενισχυτική μάθηση έχει ως σκοπό την εκμάθηση της καλύτερης δυνατής συνάρτησης αξιολόγησης. Όπως αναφέρθηκε σε προηγούμενη ενότητα, η συνάρτηση αξιολόγησης για να είναι αποδοτική απαιτεί κατάλληλη επιλογή χαρακτηριστικών και των βαρών τους. Παρόλο που η επιλογή των χαρακτηριστικών συνήθως γίνεται από τον ίδιο το σχεδιαστή, από τη διαδικασία της μάθησης προκύπτουν τα σημαντικά χαρακτηριστικά και οι τιμές των βαρών τους. Όσα χαρακτηριστικά

δεν είναι σημαντικά θα καταλήξουν με βάρη μηδενικά και θα απαλειφθούν. Έτσι, μέσω της μάθησης βελτιστοποιείται η συνάρτηση αξιολόγησης και κατ΄ επέκταση ο πράκτορας.

Η διαδικασία της ενισχυτικής μάθησης βασίζεται στην επαναληπτική αντιμετώπιση ενός προβλήματος (π.χ παιχνίδι), όπου ο πράκτορας μαθαίνει τι πρέπει να κάνει, καθώς ζημειώνεται από τις λανθασμένες επιλογές του και επιβραβεύεται για τις σωστές επιλογές του. Όσο περισσότερες οι επαναλήψεις, τόσο ο πράκτορας σταθεροποιείται και εφαρμόζει όλα όσα έχει μάθει με μεγαλύτερη αποτελεσματικότητα. Ανάλογα με τα χαρακτηριστικά του προβλήματος ενισχυτικής μάθησης που αντιμετωπίζει ο πράκτορας επιλέγεται ο κατάλληλος αλγόριθμος από την πληθώρα των αλγορίθμων που υπάρχουν στη βιβλιογραφία για τέτοιου είδους προβλήματα. Η πιο δημοφιλής επιλογή στα πλαίσια των παιχνιδιών είναι ο αλγόριθμος TD-Learning.

2.5.2 TD-Learning

Η μάθηση χρονικών διαφορών (Temporal Difference Learning, TD-Learning) [5, 6] είναι μία από τις κυριότερες μεθόδους ενισχυτικής μάθησης. Σκοπός της είναι να προσπαθήσει να ελαχιστοποιήσει τη διαφορά αξιολόγησης μεταξύ διαδοχικών καταστάσεων.

Πιο συγκεκριμένα, σε κάθε βήμα ενός παιχνιδιού προσπαθεί να προσαρμόσει την τιμή αξιολόγησης της τρέχουσας κατάστασης s, έτσι ώστε να προσεγγίσει την τιμή αξιολόγησης της κατάστασης s' που θα προκύψει αφού παίξουν και οι δύο παίκτες. Αυτό είναι επιθυμητό, διότι οι μελλοντικές καταστάσεις βρίσκονται βαθύτερα στο δέντρο, πιο κοντά στις τερματικές καταστάσεις, και δίνουν πιο αξιόπιστη πληροφορία για την έκβαση και το αποτέλεσμα του παιχνιδιού. Είναι επίσης συμβατό με τη λογική της πλήρους ανάπτυξης του δέντρου βάσει Minimax, όπου όλες οι τιμές των ενδιάμεσων κόμβων έχουν προέλθει από τιμές των φύλλων με αναδρομική διάδοση προς τη ρίζα. Έτσι, μέσα από τη διαδικασία αυτή ανανεώνονται συνεχώς τα βάρη των χαρακτηριστικών της συνάρτησης αξιολόγησης, μέχρι ο πράκτορας να μπορέσει να κάνει καλές αξιολογήσεις των ενδιάμεσων καταστάσεων και κατά συνέπεια να βελτιώσει και να σταθεροποιήσει την απόδοσή του.

Στα πλαίσια ενός παιχνιδιού, γίνεται μία επαναληπτική διεξαγωγή παρτίδων του πράκτορα που μαθαίνει ενάντια σε κάποιον αντίπαλο, όπου μετά από κάθε ζεύγος κινήσεων γίνεται ανανέωση των βαρών του πράκτορα. Ο κανόνας εκμάθησης TD-Learning που πραγματοποιεί την ανανέωση αυτή παρουσιάζεται και επεξηγείται παρακάτω:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \alpha f_i(s)\Big(Value(s') - Value(s)\Big),$$

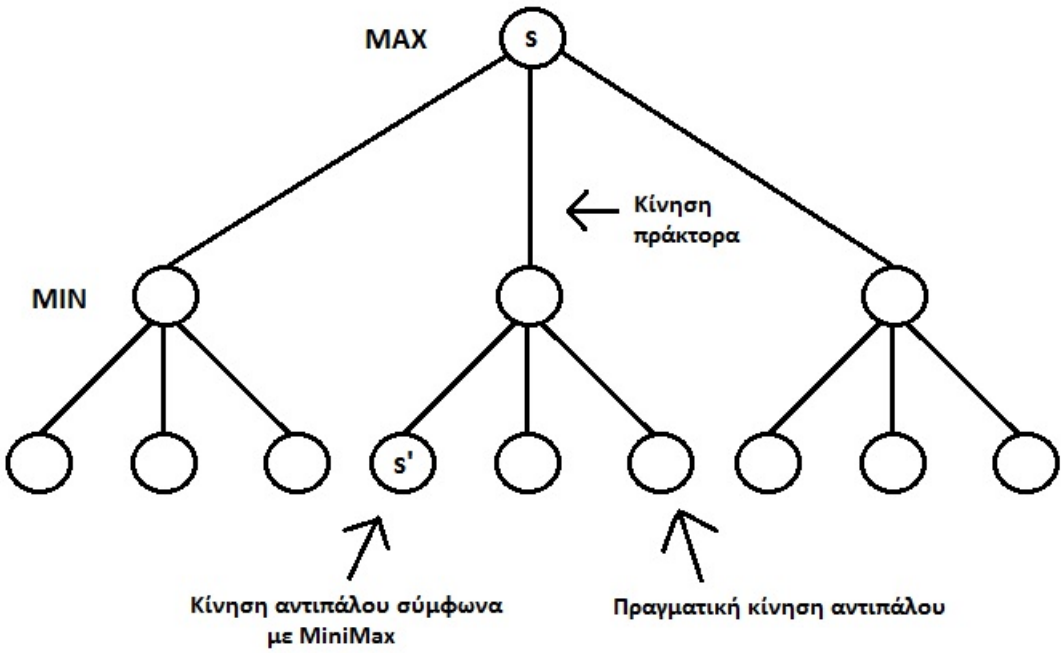
όπου s η τρέχουσα κατάσταση του παιχνιδιού,
s' η κατάσταση που προκύπτει μετά από ένα επιλεγμένο ζεύγος κινήσεων,
$Value(s)$ η τιμή αξιολόγησης της κατάστασης s,
$Value(s')$ η τιμή αξιολόγησης της κατάστασης s',
$f_i(s)$ η τιμή του χαρακτηριστικού i στην κατάσταση s,
$w_i^{(t)}$ το βάρος του χαρακτηριστικού i τη χρονική στιγμή t και
α ο ρυθμός μάθησης (learning rate) στο διάστημα $(0, 1]$.

Στον παραπάνω κανόνα φαίνεται ότι όσο μεγαλύτερη είναι η διαφορά αξιολόγησης των διαδοχικών καταστάσεων s και s', τόσο μεγαλύτερη είναι η αλλαγή στην τιμή του εκάστοτε

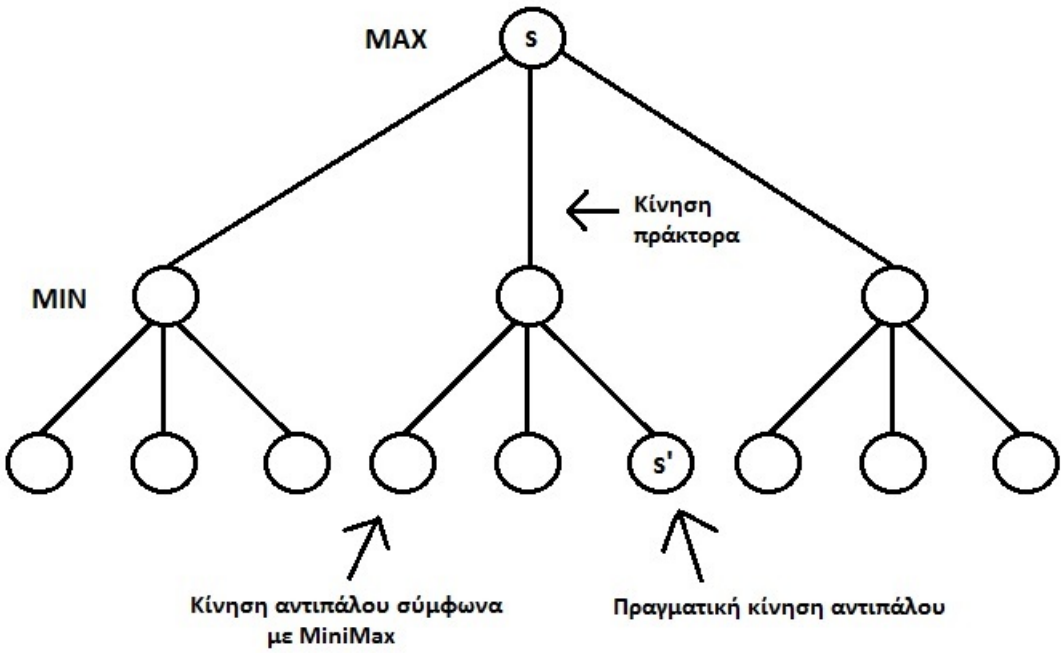
βάρους. Αυτό σημαίνει ότι ο πράκτορας μαθαίνει πως να προβλέπει με αποτελεσματικότερο τρόπο την επίδραση κάθε πιθανής επιλογής του. Αντιθέτως, όσο μικρότερη είναι η διαφορά, τόσο μικρότερη είναι η αλλαγή στην τιμή του εκάστοτε βάρους. Σε αυτήν την περίπτωση επιτυγχάνεται ο σκοπός της μάθησης και ο πράκτορας μπορεί να προβλέψει με αποτελεσματικότερο τρόπο την επίδραση κάθε πιθανής επιλογής του.

Όσον αφορά την κατάσταση s', αυτή μπορεί να επιλεχθεί με δύο διαφορετικούς τρόπους. Στην πρώτη περίπτωση που φαίνεται στο Σχήμα 2.4, επιλέγεται να είναι η κατάσταση η οποία προβλέπεται ότι θα προκύψει μετά την βέλτιστη κίνηση του πράκτορα και την βέλτιστη κίνηση του αντιπάλου του, σύμφωνα με το δέντρο παιχνιδιού και την αναζήτηση Minimax με α–β Pruning που έκανε ο πράκτορας. Στη δεύτερη περίπτωση που φαίνεται στο Σχήμα 2.5, επιλέγεται να είναι η πραγματική κατάσταση που προέκυψε μετά την βέλτιστη κίνηση του πράκτορα και την πραγματική κίνηση του αντιπάλου του. Ουσιαστικά, η διαφορά ανάμεσα στους δύο τρόπους έγκειται στην επιλογή της κίνησης του αντιπάλου. Κάθε περίπτωση έχει το μειονέκτημά της. Στην πρώτη, ο πράκτορας μαθαίνει να παίζει ενάντια στον πιο δυνατό αντίπαλο, καθώς εξετάζει μόνο βέλτιστες επιλογές του αντιπάλου (κατά την κρίση του), αλλά δεν μπορεί να ανακαλύψει και να εκμεταλλευτεί τις αδυναμίες του πραγματικού αντιπάλου του. Στη δεύτερη, ο πράκτορας μαθαίνει να παίζει ενάντια στον πραγματικό αντίπαλό του και να εκμεταλλεύεται τις αδυναμίες του, όμως ενδέχεται να μην μπορεί να αντιμετωπίσει άλλον αντίπαλο αποτελεσματικά. Ο πρώτος τρόπος οδηγεί σε πράκτορες που παίζουν πιο συντηρητικά, αλλά μπορούν να αντιμετωπίσουν μεγαλύτερο εύρος αντιπάλων, ενώ ο δεύτερος τρόπος οδηγεί σε πράκτορες που εξειδικεύονται στην αντιμετώπιση συγκεκριμένων αντιπάλων.

Τέλος, ο ρυθμός μάθησης α αντικατοπτρίζει το πόσο επηρεάζει η νέα πληροφορία την παλιά. Όσο μεγαλύτερος είναι τόσο αγνοείται η παλιά πληροφορία, με αποτέλεσμα ο πράκτορας να ρέπει συνεχώς προς καθετί νεότερο. Όσο μικρότερος είναι τόσο αγνοείται η νέα πληροφορία, με αποτέλεσμα ο πράκτορας να δίνει έμφαση στην εμπειρία που έχει συσσωρεύσει και να μαθαίνει με πολύ αργό ρυθμό. Στη διάρκεια, λοιπόν, της επαναληπτικής διαδικασίας, στα πρώτα στάδια πρέπει να δωθεί μεγάλη σημασία στην νέα πληροφορία που λαμβάνει ο πράκτορας, καθώς δεν έχει αποκτήσει εμπειρία, και σταδιακά να μεταφέρει το ενδιαφέρον του στη συσσωρευμένη πληροφορία. Έτσι, ο πράκτορας, αρχικά, λαμβάνει νέα πληροφορία και προσαρμόζεται βάσει αυτής και με το πέρας των επαναλήψεων σταθεροποιείται στην εμπειρία που συνέλλεξε. Για να επιτευχθεί αυτή η σταδιακή μετάβαση στην πράξη, ο ρυθμός μάθησης πρέπει να ξεκινάει με μία σχετικά μεγάλη τιμή και να μειώνεται όσο εκτελείται η διαδικασία της μάθησης, με στόχο να προσεγγίσει την τιμή 0 στο τέλος.



Σχήμα 2.4: *Πρώτη Περίπτωση Επιλογής s'.*



Σχήμα 2.5: *Δεύτερη Περίπτωση Επιλογής s'.*

Κεφάλαιο 3

Περιγραφή Προβλήματος

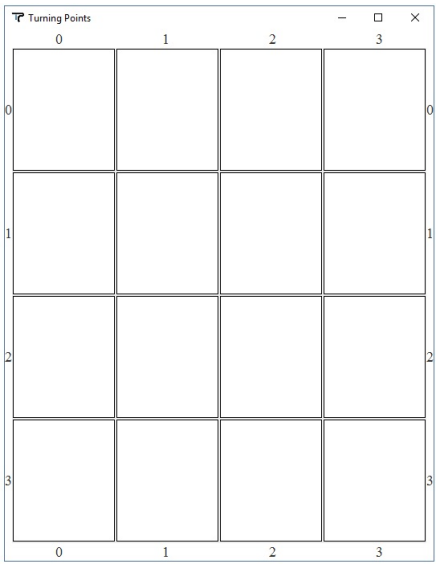
Ένας σημαντικός τομέας της Τεχνητής Νοημοσύνης, από τις απαρχές της, είναι τα παιχνίδια. Τα παιχνίδια διακρίνονται σε διαφορετικές κατηγορίες ανάλογα με το πόσοι παίχτες συμμετέχουν. Η κατηγορία που έχει μελετηθεί περισσότερο είναι τα παιχνίδια μηδενικού αθροίσματος δύο παικτών (two-player, zero-sum games). Σε τέτοιου είδους παιχνίδια σκοπός είναι να υλοποιηθεί ένας ευφυής πράκτορας που θα παίζει το παιχνίδι όσο το δυνατόν καλύτερα ενάντια σε οποιοδήποτε αντίπαλο. Ένα τέτοιο παιχνίδι μηδενικού αθροίσματος δύο παικτών είναι και το επιτραπέζιο παιχνίδι Turning Points.

3.1 Turning Points

Το επιτραπέζιο παιχνίδι Turning Points [7] επινοήθηκε από τον Joseph Kisenwether το 2004 και ανήκει στη συλλογή παιχνιδιών *Icehouse Games*. Πρόκειται για ένα παιχνίδι στο οποίο συμμετέχουν δύο παίκτες, ο *Βόρειος (North)* και ο *Νότιος (South)*. Το περιβάλλον στο οποίο ανταγωνίζονται οι δύο παίκτες είναι ένα τετραγωνικό board 3×3. Στην εργασία αυτή, όμως, το board μπορεί να έχει οποιοδήποτε μέγεθος. Το παιχνίδι από μόνο του είναι δύσκολο στη λήψη απόφασης για την καταλληλότερη κίνηση, λόγω του ότι είναι πολύ εύκολο με μία κίνηση να έχουμε δραματικές αλλαγές στην κατάσταση του παιχνιδιού, πράγμα που το κάνει ακόμα πιο ενδιαφέρον. Έχει υψηλό παράγοντα διακλάδωσης (branching factor) ο οποίος αυξάνεται τετραγωνικά όσο μεγαλύτερο γίνεται το board. Το παιχνίδι έχει 5^{16} πιθανές καταστάσεις (κάθε κουτάκι μπορεί να έχει βέλος σε μία από τις τέσσερις κατευθύνσεις ή να είναι άδειο, δηλαδή πέντε επιλογές για 16 κουτάκια), 4^{16} πιθανές τερματικές καταστάσεις (στις τερματικές καταστάσεις υπάρχει σε κάθε κουτάκι βέλος σε μία από τις τέσσερις κατευθύνσεις, οπότε τέσσερις επιλογές για 16 κουτάκια) οι οποίες δεν είναι όλες απαραίτητα προσβάσιμες από την αρχική κατάσταση και 10^{23} διαφορετικές εκβάσεις (φύλλα του δέντρου παιχνιδιού) των οποίων ο υπολογισμός αναλύεται στην Ενότητα 4.1. Συνεπώς, είναι πολύ δύσκολο για έναν άνθρωπο να είναι πλήρως αποτελεσματικός σε αυτό το παιχνίδι.

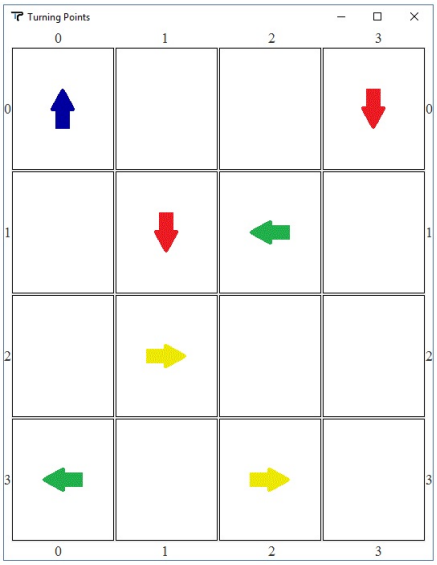
Στην παρούσα εργασία θεωρήθηκε πως το βασικό board έχει διαστάσεις 4×4, διαστάσεις οι οποίες είναι αρκετά μεγάλες για να εξαχθεί καλύτερος πράκτορας και αρκετά μικρές ώστε να μην αυξάνεται δραματικά το υπολογιστικό κόστος. Αρχικά, το board είναι άδειο και παίζει πρώτος οποιοσδήποτε από τους δύο παίκτες συμφωνηθεί (Σχήμα 3.1).

Κάθε παίκτης, όταν έχει σειρά να παίξει, μπορεί να τοποθετήσει σε ένα άδειο κουτάκι του board ένα βέλος που δείχνει σε μία από τις τέσσερις βασικές κατευθύνσεις (Βορράς / μπλε,



Σχήμα 3.1: *Αρχική Κατάσταση του Παιχνιδιού Turning Points.*

Ανατολή / κίτρινο, Νότος / κόκκινο, Δύση / πράσινο). Εάν το βέλος "κοιτάει" σε κουτάκι που είναι άδειο ή έξω από το board, τότε δε συμβαίνει κάποια περαιτέρω αλλαγή και σειρά για να παίξει παίρνει ο άλλος παίκτης. Εάν, όμως, το βέλος "κοιτάει" σε κουτάκι που περιέχει άλλο βέλος, τότε περιστρέφεται το άλλο βέλος κατά 90^o δεξιόστροφα. Αν αυτό με τη σειρά του, μετά την περιστροφή του, "κοιτάει" σε κουτάκι με βέλος τότε συμβαίνει το ίδιο στο επόμενο βέλος. Αυτό συνεχίζεται μέχρι το βέλος που μόλις περιστράφηκε να "κοιτάξει" σε κουτάκι που είναι άδειο ή έξω από το board, όπου και σταματάει η διαδικασία περιστροφής των βελών και δίνεται η σειρά στον άλλον παίκτη να παίξει. Μία ενδιάμεση κατάσταση του παιχνιδιού φαίνεται στο Σχήμα 3.2. Τονίζεται ότι σε περίπτωση που κάποιο βέλος μετά την περιστροφή του "κοιτάξει" το βέλος που τοποθέτησε ο παίκτης στην κίνησή του, τότε γίνεται κανονικά η περιστροφή και αυτού του βέλους.

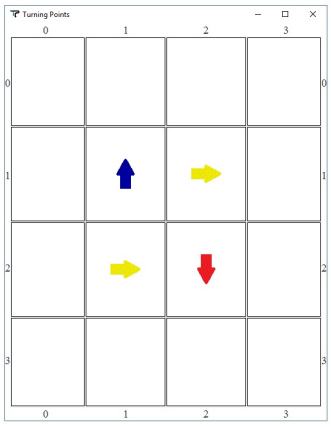


Σχήμα 3.2: *Ενδιάμεση Κατάσταση του Παιχνιδιού Turning Points.*

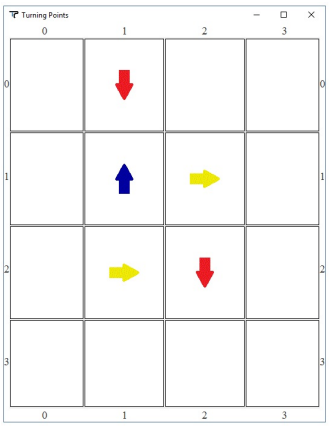
Η διαδικασία εκτέλεσης μίας κίνησης γίνεται πιο κατανοητή βλέποντας το Σχήμα 3.3.

Το παιχνίδι τελειώνει όταν γεμίσει το board, δηλαδή στο 4×4 μετά από 16 κινήσεις (οχτώ για κάθε παίκτη). Νικητής είναι ο παίκτης που έχει περισσότερα βέλη στραμμένα προς το μέρος του. Με λίγα λόγια, ο γενικότερος σκοπός του παιχνιδιού είναι κάθε παίκτης να προσπαθήσει να εξασφαλίσει τα περισσότερα βέλη στραμμένα προς αυτόν στο τέλος, με τρόπο τέτοιο που δε θα επιτρέπει στον αντίπαλο να πραγματοποιήσει κάτι αντίστοιχο. Πρέπει να σημειωθεί ότι πρόκειται για ένα παιχνίδι μηδενικού αθροίσματος, το οποίο έχει σαν πιθανό αποτέλεσμα ακόμα και την ισοπαλία.

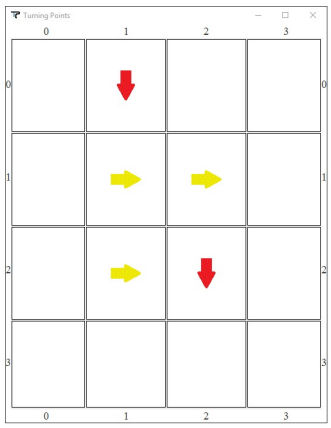
Στο Σχήμα 3.4 φαίνεται μία τερματική κατάσταση στην οποία νικητής είναι ο Βόρειος παίκτης με σκορ 7-3. Ενώ στο Σχήμα 3.5 φαίνεται μία νίκη του Νότιου παίκτη με σκορ 4-7. Τέλος, στο Σχήμα 3.6 φαίνεται μία περίπτωση ισοπαλίας με σκορ 5-5.



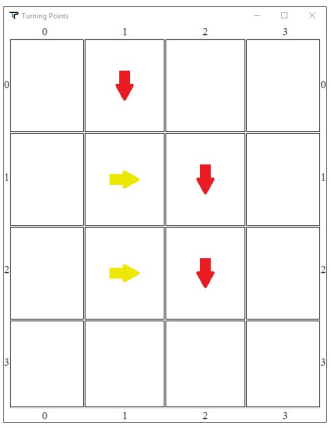
(α′) *Αρχικά.*



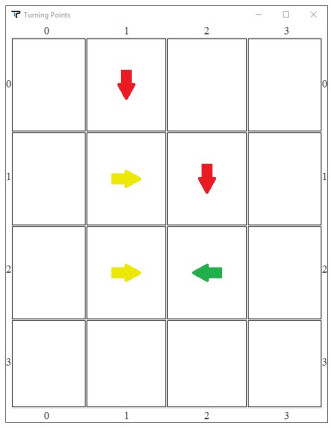
(β′) *Κίνηση: Νότιο βέλος στο (0,1).*



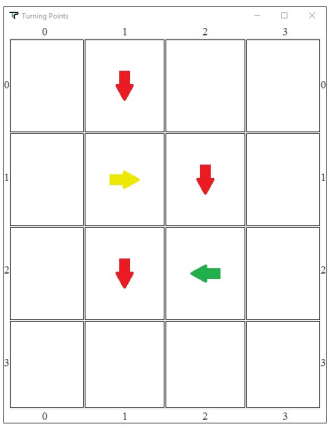
(γ′) *Περιστροφή βέλους στο (1,1).*



(δ′) *Περιστροφή βέλους στο (1,2).*

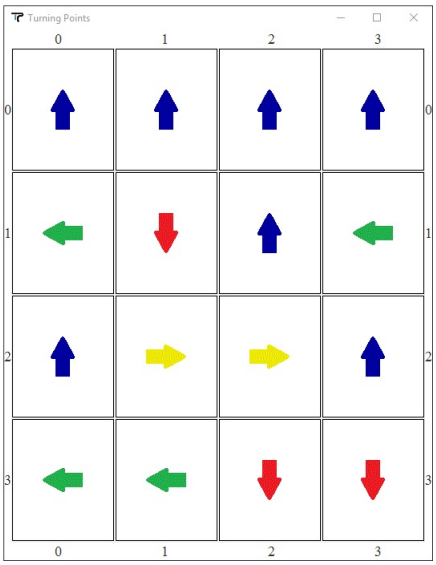


(ε′) *Περιστροφή βέλους στο (2,2).*

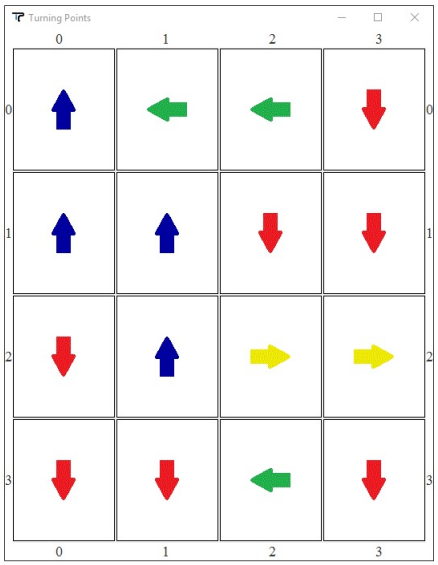


(ϛ′) *Περιστροφή βέλους στο (2,1).*

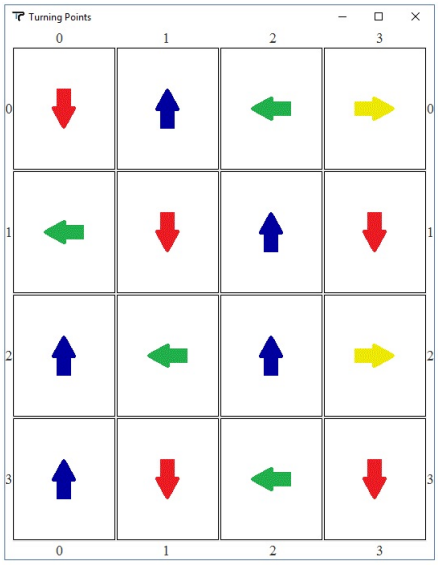
Σχήμα 3.3: *Παράδειγμα Εκτέλεσης Κίνησης στο Παιχνίδι Turning Points.*



Σχήμα 3.4: *Τερματική Κατάσταση του Παιχνιδιού Turning Points - Νίκητης Βόρειος.*



Σχήμα 3.5: *Τερματική Κατάσταση του Παιχνιδιού Turning Points - Νίκητης Νότιος.*



Σχήμα 3.6: *Τερματική Κατάσταση του Παιχνιδιού Turning Points - Ισοπαλία.*

3.2 Στόχος Εργασίας

Στόχος της εργασίας αυτής είναι η σχεδίαση και η εκπαίδευση ενός πράκτορα που θα παίζει όσο το δυνατόν βέλτιστα το παιχνίδι Turning Points.

Ένα απαραίτητο βήμα για τον παραπάνω στόχο είναι να υλοποιηθεί το κατάλληλο περιβάλλον για το παιχνίδι. Αυτό σημαίνει ότι, αρχικά, θα πρέπει να υλοποιηθεί το παιχνίδι υποστηριζόμενο από το σύνολο των κανόνων του και στη συνέχεια το γραφικό περιβάλλον για την απεικόνισή του.

Όσον αφορά τον πράκτορα, στόχος είναι να χρησιμοποιηθούν, αρχικά, τεχνικές αναζήτησης, όπως Minimax και η βελτίωσή της α - β Pruning, και στη συνέχεια, να σχεδιαστεί και να ενσωματωθεί κάποια κατάλληλη συνάρτηση αξιολόγησης. Έπειτα, θα εφαρμοστεί η διαδικασία της ενισχυτικής μάθησης με χρήση της μεθόδου χρονικών διαφορών (Temporal Difference (TD) Learning), κατά την οποία ο πράκτορας μαθαίνει να βελτιώνει την απόδοσή του στο παιχνίδι. Τέλος, στόχος είναι να συγκριθούν διάφοροι παίκτες, που θα κατασκευαστούν ευριστικά ή θα προκύψουν ως στιγμιότυπα του πράκτορα, μέσα από τη διεξαγωγή τουρνουά και την εξέταση των αποτελεσμάτων.

3.3 Σχετικές Εργασίες

Η επιλογή του παιχνιδιού Turning Points έγινε μετά από μεγάλη αναζήτηση για ένα παιχνίδι μη τετριμμένο, το οποίο δεν έχει μελετηθεί στο παρελθόν. Συνεπώς, δεν υπάρχουν σχετικές εργασίες γι΄ αυτό το παιχνίδι, όσον αφορά το κομμάτι της σχεδίασης και υλοποίησης ενός πράκτορα που προκύπτει από κάποια διαδικασία μάθησης και παίζει το παιχνίδι βάσει κάποιας μεθόδου αναζήτησης. Ωστόσο, στο παρελθόν έχει υλοποιηθεί ένα γραφικό περιβάλλον για το παιχνίδι αυτό, έτσι ώστε όσοι ενδιαφέρονται για το παιχνίδι να μπορούν να παίξουν αντίπαλοι με ένα άλλο χρήστη (όχι με τον υπολογιστή). Πρόκειται για ένα demo [8] που κατασκευάστηκε από τον Tom Mason, ο οποίος είχε τη βοήθεια των Geoff Hanna και Zoe Miller.

Μπορεί, λοιπόν, να μην υπάρχει κάτι σχετικό με το εν λόγω παιχνίδι στη βιβλιογραφία, αλλά γενικά υπάρχουν πολλές σημαντικές και αξιοπρόσεκτες εργασίες στον τομέα της Τεχνητής Νοημοσύνης και της Ενισχυτικής Μάθησης.

Ο Claude Shannon το 1950 σχεδίασε τον πρώτο πράκτορα που μπορούσε να παίξει σκάκι [9]. Το 1959 ο Arthur L. Samuel σχεδίασε έναν ικανοποιητικό πράκτορα για το παιχνίδι της ντάμας [10], ο οποίος μάθαινε να παίζει αντιμετωπίζοντας τον εαυτό του.

Ένα παιχνίδι-σταθμός στην ιστορία των πρακτόρων είναι το παιχνίδι Othello. Οι πράκτορες *IAGO* [11] και *The Moor* που υλοποιήθηκαν μέχρι και το 1980 μπορούσαν να νικήσουν τους καλύτερους παίκτες. Σημαντική πρόοδο, όμως, έδειξε ο πράκτορας *Logistello* [12] του Michael Buro, ο οποίος το 1997 κέρδισε τον παγκόσμιο πρωταθλητή Takeshi Murakami με σκορ 6-0 [13]. Είναι πολύ σημαντικά τα όσα επιτεύχθηκαν στο παιχνίδι αυτό, αν αναλογιστεί κανείς ότι το παιχνίδι έχει περίπου 10^{28} πιθανές καταστάσεις και 10^{58} διαφορετικές εκβάσεις (φύλλα του δέντρου παιχνιδιού).

Άλλη μία σημαντική εργασία εκπονήθηκε το 1992 από τον Gerald Tesauro . Δημιούργησε τον πράκτορα *TD-Gammon* [14] για το παιχνίδι πόρτες (τάβλι). Ο πράκτορας αυτός δεν κέρδισε τους καλύτερους παίκτες, όμως έφτασε πολύ κοντά. Αξιοσημείωτο είναι το γεγονός

ότι παρά τις ήττες του, οι αντίπαλοί του βελτιώθηκαν ανακαλύπτοντας νέες τεχνικές και στρατηγικές οι οποίες ανέβασαν το συνολικό επίπεδο του παιχνιδιού.

Τα παραπάνω είναι ένα μικρό δείγμα για το πως εξελίσσονται οι τομείς που προαναφέρθηκαν. Είναι προφανές, λοιπόν, ότι υπάρχουν και θα υπάρξουν ακόμα μεγαλύτερα επιτεύγματα στους τομείς αυτούς τα προσεχή χρόνια.

Κεφάλαιο 4

Η Δική μας Προσέγγιση

Η υλοποίηση ενός πράκτορα που παίζει ένα παιχνίδι απαιτεί μεθοδικότητα, οργάνωση και σωστή μετάβαση από ένα στάδιο υλοποίησης στο επόμενο. Αυτά, λοιπόν, είναι τα σημαντικότερα στοιχεία που συνέβαλλαν στην υλοποίηση του πράκτορα για το παιχνίδι Turning Points. Υπενθυμίζεται ότι ο πράκτορας υλοποιήθηκε με βάση το board με διαστάσεις 4×4.

4.1 Δέντρο Παιχνιδιού MiniMax

Το δέντρο του παιχνιδιού Minimax [1, 2, 3] (Ενότητα 2.3.2) είναι η δομή που επιτρέπει στον πράκτορα να υπολογίσει όλους τους πιθανούς συνδυασμούς κινήσεων των δύο παικτών που τίθενται αντιμέτωποι, του Max (ο πράκτοράς μας) και του Min. Υπολογίζοντας όλους αυτούς τους συνδυασμούς, ο πράκτοράς μας είναι σε θέση να επιλέξει την κίνηση που θεωρεί πιο προσοδοφόρα.

Το δέντρο κατασκευάζεται κάθε φορά που ο πρακτοράς μας έχει σειρά να παίξει. Η επέκταση των κόμβων γίνεται με βάση τον αλγόριθμο της αναζήτησης με υπαναχώρηση. Στο πρώτο επίπεδο του δέντρου βρίσκεται ο κόμβος-ρίζα που περιέχει την κατάσταση που επικρατεί στο παιχνίδι εκείνη τη στιγμή. Στο επόμενο επίπεδο, δημιουργούνται τα παιδιά του κόμβου-ρίζα που αντιστοιχούν σε όλες τις πιθανές ενέργειες που μπορεί να κάνει ο πράκτοράς μας, δηλαδή τοποθέτηση βέλους σε κάθε άδειο κουτάκι με κάθε πιθανή κατεύθυνση. Στο αμέσως επόμενο επίπεδο συμβαίνει το ίδιο ακριβώς για τον παίκτη Min. Αυτό συνεχίζεται εναλλάξ για κινήσεις των δύο παικτών, μία δική μας (Max) και μία του αντιπάλου (Min), μέχρι το όριο βάθους επέκτασης (αποκοπή) D. Συνεπώς, επιλέγεται κίνηση προβλέποντας συνδυασμούς μετά από D κινήσεις. Κανονικά το δέντρο πρέπει να επεκταθεί μέχρι τις τερματικές καταστάσεις του παιχνιδιού, όμως, το υπολογιστικό κόστος για κάτι τέτοιο είναι τεράστιο. Για το λόγο αυτό γίνεται αποκοπή της αναζήτησης σε βάθος D και χρήση της συνάρτησης αξιολόγησης. Ενδεικτικά, στην παρούσα εργασία χρησιμοποιήθηκε όριο βάθους επέκτασης $D = 4$, το οποίο εξισορροπεί ικανοποιητικά την ποιότητα της τελικής απόφασης σε σχέση με το χρόνο αναζήτησης.

Για να γίνει κατανοητό το όριο επιτρεπτού βάθους, πρέπει να αναφέρουμε ότι ο παράγοντας διακλάδωσης (branching factor) είναι 36 παιδιά ανά κόμβο. Αυτό δείχνει ότι σε κάθε επίπεδο υπάρχουν κατά μέσο όρο 36 παιδιά, που το καθένα έχει 36 παιδιά κ.ο.κ. Δεν είναι, όμως, ακριβώς έτσι διότι, αρχικά, είναι περισσότερες οι επιτρεπτές κινήσεις και για κάθε επίπεδο που κατεβαίνουμε λιγοστεύουν. Αυτό φαίνεται στο παράδειγμα που ακολουθεί. Εάν ο παίκτης μας

παίζει πρώτος, έχει 16 κουτάκια άδεια να κινηθεί με τέσσερις πιθανές επιλογές για το καθένα, άρα 64 πιθανές κινήσεις (παιδιά). Ο αντίπαλος θα έχει 15 κουτάκια, οπότε 60 πιθανές κινήσεις, κ.ο.κ. Με αυτά τα δεδομένα υπολογίζεται ότι οι πιθανές εκβάσεις του παιχνιδιού είναι περίπου $64 \times 60 \times 56 \times ... \times 8 \times 4 \approx 10^{23}$, όπως αναφέρεται στην Ενότητα 3.1.

Για να κατασκευάσουμε, λοιπόν, το δέντρο χρησιμοποιούμε αναδρομικές κλήσεις που μας οδηγούν ένα βάθος πιο κάτω. Στις τερματικές καταστάσεις ή σε καταστάσεις που φτάνουμε στο μέγιστο επιτρεπτό βάθος D, η αρχική μας προσέγγιση ήταν να κάνουμε αξιολόγηση με βάση το score του παιχνιδιού στην εκάστοτε κατάσταση. Εφαρμόζοντας τον αλγόριθμο MiniMax επιλέγεται η καταλληλότερη κίνηση για τον πράκτορά μας. Πρέπει να σημειωθεί στην περίπτωση ισοδύναμων κινήσεων βάσει αξιολόγησης, οι κινήσεις αυτές εισάγονται σε μία λίστα από την οποία επιλέγεται η τελική κίνηση τυχαία.

4.2 Βελτίωση α-β Pruning

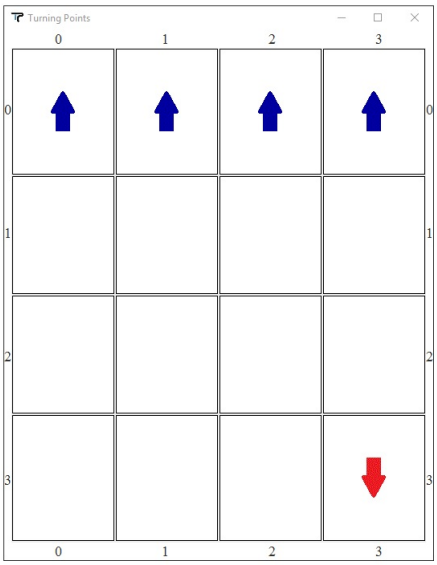
Όπως προαναφέρθηκε, η αναζήτηση τερματίζεται σε όριο βάθους επέκτασης D. Εφόσον, όμως, η αναζήτηση γίνεται με βάση μόνο τον αλγόριθμο Minimax , ο χρόνος επιλογής κίνησης στην πράξη, ακόμα και για $D = 4$ δεν ήταν τόσο ικανοποιητικός (μεγαλύτερος των τριών δευτερολέπτων). Το μειονέκτημα, του Minimax είναι ότι επεκτείνει όλους τους κόμβους. Όμως, όπως αναφέρεται στην Ενότητα 2.3.3, υπάρχει βελτίωση του αλγορίθμου αυτού.

Ο αλγόριθμος Minimax με α–β Pruning [1, 2, 4] βοήθησε στο να αποκόπτονται κόμβοι από το δέντρο που δεν έχουν καμία χρησιμότητα. Έτσι, μειώνονται οι συνολικοί κόμβοι που προσπελαύνονται αφού οι αποκομμένοι κόμβοι δεν επεκτείνονται.

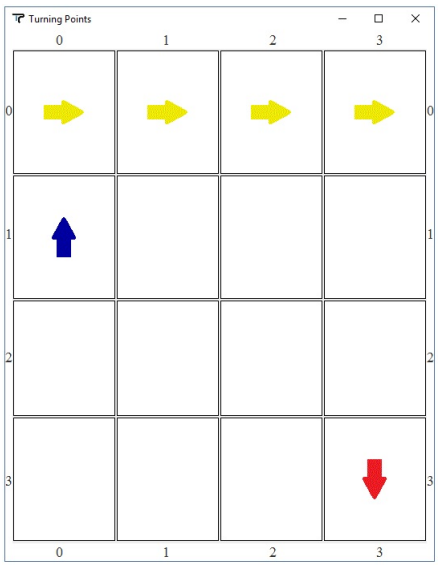
Με αυτόν τον τρόπο καταφέραμε να είναι πιο γρήγορος ο πράκτοράς μας με όριο βάθους $D = 4$, ακόμα και με όριο βάθους $D = 5$. Δε μπορέσαμε, όμως, να ξεπεράσουμε αυτά τα όρια δεδομένου του υλικού που διαθέτουμε. Συνεπώς, καταλήξαμε στη χρήση ορίου βάθους $D = 4$, λόγω του ότι μετά την επέκταση των κόμβων σε άρτιο βάθος ο πράκτοράς μας καταλήγει σε τερματικές καταστάσεις που είναι η σειρά του να παίξει. Η αποκοπή σε άρτιο βάθος παίζει σημαντικό ρόλο αφού επιλέγεται, τελικά, κίνηση με πρόβλεψη η οποία σταματάει σε γύρο όπου παίζει και έχει τον έλεγχο του παιχνιδιού ο πράκτοράς μας και έτσι δεν αφήνει την πρόβλεψή του στα χέρια του αντιπάλου. Αυτό αντιστοιχεί στις ήρεμες καταστάσεις (quiescent states) του παιχνιδιού, που πρώτος περιέγραψε ο Claude Shannon [9].

4.3 Συνάρτηση Αξιολόγησης

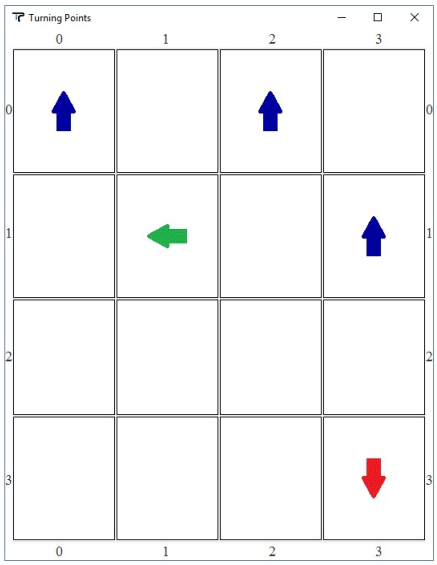
Αφού καταλήξαμε στον αλγόριθμο Minimax με α–β Pruning και στο κατάλληλο όριο βάθους, συνεχίσαμε με τη συνάρτηση αξιολόγησης (evaluation function) [1, 5] (Ενότητα 2.4). Στα αρχικά στάδια χρησιμοποιήθηκε ως συνάρτηση αξιολόγησης το score της εκάστοτε κατάστασης, κάτι που δεν ήταν τόσο αποτελεσματικό. Στο Σχήμα 4.1 φαίνεται μία περίπτωση που η πρόβλεψη με βάση μόνο το score παραπλανεί τον πράκτορά μας. Στο συγκεκριμένο παράδειγμα ο πράκτοράς μας αγωνίζεται ως Βόρειος. Στην πρώτη περίπτωση, η κατάσταση s_1 που φαίνεται στο (α′) αξιολογείται με τιμή $Score(s_1) = 4 - 1 = 3$. Όμως, αν ο αντίπαλος τοποθετήσει βόρειο βέλος στο (1,0), προκύπτει η κατάσταση s'_1 που φαίνεται στο (β′), η οποία αξιολογείται με τιμή $Score(s'_1) = 1 - 1 = 0$. Στη δεύτερη περίπτωση, η κατάσταση s_2 που



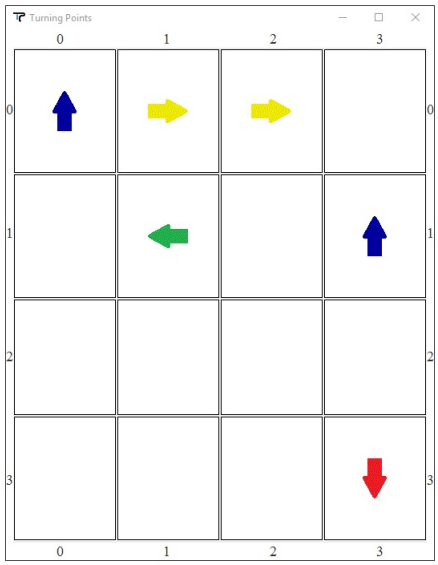
(α′) $Score(s_1) = 4 - 1 = 3$.



(β′) $Score(s'_1) = 1 - 1 = 0$.



(γ′) $Score(s_2) = 3 - 1 = 2$.



(δ′) $Score(s'_2) = 2 - 1 = 1$.

Σχήμα 4.1: *Παράδειγμα Αξιολόγησης Καταστάσεων με Βάση το Score.*

φαίνεται στο (γ′) αξιολογείται με τιμή $Score(s_2) = 3 - 1 = 2$. Όμως, αν ο αντίπαλος τοποθετήσει ανατολικό βέλος στο (0,1), προκύπτει η κατάσταση s'_2 που φαίνεται στο (δ′), η οποία αξιολογείται με τιμή $Score(s'_2) = 2 - 1 = 1$. Παρόλο που η αξιολόγηση με βάση το score υποδεικνύει ότι η κατάσταση s_1 είναι προτιμότερη από την s_2, είναι φανερό από τα παραπάνω ότι η s_1 είναι πιο ευάλωτη από την s_2, οπότε μακροπρόθεσμα ίσως πρέπει να προτιμηθεί η s_2. Το παραπάνω παράδειγμα, επίσης, δείχνει για ποιο λόγο η αξιολόγηση ενδιάμεσων καταστάσεων είναι καλύτερο να γίνεται σε κόμβους όπου παίζει ο πράκτοράς μας.

Εξετάζοντας, λοιπόν, το παιχνίδι αναζητήσαμε κάποια χαρακτηριστικά (features) που προσδιορίζουν την κατάσταση του παιχνιδιού, ώστε η αξιολόγηση να βασιστεί σε αυτά και έτσι να παίξουν σημαντικό ρόλο στην έκβαση του παιχνιδιού και στην λήψη απόφασης για την καταλληλότερη κίνηση. Καταλήξαμε σε ένα σύνολο 16 χαρακτηριστικών τα οποία φαίνονται στον Πίνακα 4.1. Γενικά, θεωρούμε ότι ένα τοποθετημένο βέλος απειλείται άμεσα όταν έχει

$\mathbf{f_i}$	**Χαρακτηριστικό**	**Περιγραφή**
f_1	`score`	Το score του παιχνιδιού.
f_2	`eastArrows`	Πλήθος ανατολικών βελών.
f_3	`westArrows`	Πλήθος δυτικών βελών.
f_4	`northSafeArrows`	Πλήθος βορείων βελών που δεν απειλούνται άμεσα.
f_5	`southSafeArrows`	Πλήθος νοτίων βελών που δεν απειλούνται άμεσα.
f_6	`northCornerArrows`	Πλήθος βορείων βελών που βρίσκονται σε γωνία του board.
f_7	`southCornerArrows`	Πλήθος νοτίων βελών που βρίσκονται σε γωνία του board.
f_8	`northOpenArrows`	Πλήθος βορείων βελών που έχουν γειτονικό κενό κουτάκι.
f_9	`southOpenArrows`	Πλήθος νοτίων βελών που έχουν γειτονικό κενό κουτάκι.
f_{10}	`northFragileArrows`	Πλήθος βορείων βελών που απειλούνται άμεσα.
f_{11}	`southFragileArrows`	Πλήθος νοτίων βελών που απειλούνται άμεσα.
f_{12}	`eastArrPerSeq`	Μέσο πλήθος ανατολικών βελών ανά κάθετη ακολουθία ανατολικών βελών.
f_{13}	`eastArrInSeq`	Συνολικό πλήθος ανατολικών βελών που βρίσκονται σε ακολουθίες ανατολικών βελών.
f_{14}	`westArrPerSeq`	Μέσο πλήθος δυτικών βελών ανά κάθετη ακολουθία δυτικών βελών.
f_{15}	`westArrInSeq`	Συνολικό πλήθος δυτικών βελών που βρίσκονται σε ακολουθίες δυτικών βελών.
f_{16}	`movesLeft`	Κινήσεις που απομένουν.

Πίνακας 4.1: *Χαρακτηριστικά*

τουλάχιστον ένα γειτονικό (προς τις τέσσερις βασικές κατευθύνσεις) κενό κουτάκι, που σημαίνει ότι μπορεί να περιστραφεί άμεσα αν γίνει η κατάλληλη κίνηση στη γειτονιά του. Επίσης, ένα τοποθετημένο βέλος απειλείται άμεσα όταν μία περιστροφή κάποιου υπάρχοντος γειτονικού του βέλους θα προκαλέσει και περιστροφή του ιδίου. Κάθετη ακολουθία όμοιων βελών θεωρείται οποιαδήποτε συστάδα δύο ή περισσότερων όμοιων συνεχόμενων βελών. Για λόγους ισονομίας, όλα τα χαρακτηριστικά κανονικοποιούνται στο διάστημα [0,1] με βάση τις μέγιστες και ελάχιστες τιμές τους.

Κάποια από τα χαρακτηριστικά αυτά δεν δίνουν τόσο χρήσιμη πληροφορία από μόνα τους. Για να αποδώσουν καλύτερα συνδυάζονται με άλλα χαρακτηριστικά, όπου ο συνδυασμός τους αποτελεί ένα νέο συνδυαστικό (cross-term) χαρακτηριστικό. Πιο συγκεκριμένα, υιοθετήσαμε δύο τέτοια χαρακτηριστικά που αποτελούν συνδυασμό δύο απλών χαρακτηριστικών και φαίνονται στον Πίνακα 4.2. Διαισθητικά τα χαρακτηριστικά αυτά δίνουν ένα μέτρο της κατανομής συνεχόμενων όμοιων βελών σε διαφορετικές ακολουθίες.

Με τα χαρακτηριστικά και τα βάρη που τους αντιστοιχούν υπολογίζεται η συνάρτηση αξιολόγησης. Τα βάρη αντιπροσωπεύουν τη συμβολή του κάθε χαρακτηριστικού στην αξιολόγηση

$\mathbf{f_i}$	**Χαρακτηριστικό**	**Περιγραφή**
f_{17}	`eastArrSeq`	Είναι ο συνδυασμός (`eastArrPerSeq` × `eastArrInSeq`) και αντιστοιχεί στο πλήθος και στο μέγεθος ακολουθιών ανατολικών βελών.
f_{18}	`westArrSeq`	Είναι ο συνδυασμός (`westArrPerSeq` × `westArrInSeq`) και αντιστοιχεί στο πλήθος και στο μέγεθος ακολουθιών δυτικών βελών.

Πίνακας 4.2: *Συνδυαστικά Χαρακτηριστικά*

μίας κατάστασης. Όταν φτάσει ο πράκτοράς μας σε μία κατάσταση που πρέπει να αξιολογηθεί, τότε υπολογίζεται η συνάρτηση αξιολόγησης για την κατάσταση αυτή. Αρχικά, υπολογίζονται τα 18 χαρακτηριστικά και κανονικοποιούνται. Στη συνέχεια, κάθε χαρακτηριστικό πολλαπλασιάζεται με το βάρος που του αντιστοιχεί και, τέλος, αθροίζονται για να υπολογιστεί η τιμή της αξιολόγησης. Ο τύπος υπολογισμού της συνάρτησης αξιολόγησης φαίνεται παρακάτω:

$$Value(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_{18} f_{18}(s) = \sum_{i=1}^{18} w_i f_i(s),$$

όπου s η κατάσταση του παιχνιδιού που αξιολογείται,
$Value(s)$ η τιμή αξιολόγησης της κατάστασης s,
f_i η τιμή του χαρακτηριστικού i και
w_i το βάρος του χαρακτηριστικού i.

Σημειώνεται ότι στην περίπτωση που μία κατάσταση σε φύλλο του δέντρου είναι τερματική κατάσταση του παιχνιδιού, τότε αξιολογείται μόνο με βάση το score.

Για να μπορεί ο πράκτοράς μας να είναι αποτελεσματικός, χρειάζεται κατάλληλη και αποδοτική συνάρτηση αξιολόγησης. Όπως προαναφέρθηκε, κάθε χαρακτηριστικό της συνάρτησης αξιολόγησης έχει ένα βάρος που του αντιστοιχεί. Για να είναι λοιπόν αποδοτική η συνάρτηση αξιολόγησης χρειάζεται σωστή επιλογή βαρών. Ένα σύνολο βαρών (διάνυσμα) $\mathbf{w}$ αποτελεί μία εκδοχή του πράκτορά μας. Συνεπώς, κάθε τέτοιο διάνυσμα είναι ένας άλλος παίκτης.

Πριν προχωρήσουμε στην εκμάθηση του πράκτορά μας, δημιουργήσαμε χειροκίνητα ένα παίκτη (διάνυσμα βαρών) αρκετά αξιόλογο. Ο παίκτης αυτός ονομάστηκε MP (My Player). Αυτός ο παίκτης ήταν βαρόμετρο για τη διαδικασία της μάθησης, αφού ο πράκτοράς μας θα μπορούσε να μάθει και ενάντια σε έναν αξιόλογο παίκτη. Πρέπει να σημειωθεί ότι ο παίκτης αυτός ήταν πολύ αποτελεσματικός ενάντια στον παίκτη που αξιολογεί με βάση μόνο το score, κερδίζοντάς τον στο 90% των παιχνιδιών που ήρθαν αντιμέτωποι. Επίσης, με ποσοστό 95% κερδίζει τον παίκτη που παίζει εντελώς τυχαία.

4.4 TD-Learning

Η εκμάθηση του πράκτορά μας είναι μία επαναληπτική διαδικασία, κατά την οποία ο παίκτης μας καλείται να αντιμετωπίσει έναν αντίπαλο σε πολλά συνεχόμενα παιχνίδια με σκοπό να μάθει να παίζει καλύτερα. Μία διαδικασία μάθησης διαρκεί πολλές επαναλήψεις, δηλαδή ο πρακτοράς

μας αντιμετωπίζει τον αντίπαλο σε πάρα πολλά παιχνίδια. Ανά κάποιο αριθμό επαναλήψεων οι δύο παίκτες αλλάζουν πλευρά ώστε η μάθηση να καλύψει και τις δύο πλευρές. Τα βάρη του πράκτορά μας αρχικοποιούνται σε όποιες τιμές επιλέξουμε, πριν ξεκινήσει η διαδικασία μάθησης.

Κατά τη διάρκεια της διαδικασίας, ο παίκτης μας μετά από κάθε κίνηση που πραγματοποιεί ανανεώνει τα βάρη του. Η ανανέωση αυτή γίνεται με βάση τον αλγόριθμο TD-Learning [5, 6] (Ενότητα 2.5.2):

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \alpha f_i(s)\Big(Value(s') - Value(s)\Big),$$

όπου s η τρέχουσα κατάσταση του παιχνιδιού,
s' η κατάσταση που προκύπτει μετά από ένα επιλεγμένο ζεύγος κινήσεων,
$Value(s)$ η τιμή αξιολόγησης της κατάστασης s,
$Value(s')$ η τιμή αξιολόγησης της κατάστασης s',
$f_i(s)$ η τιμή του χαρακτηριστικού i στην κατάσταση s,
$w_i^{(t)}$ το βάρος του χαρακτηριστικού i τη χρονική στιγμή t και
α ο ρυθμός μάθησης (learning rate) στο διάστημα $(0, 1]$.

Για την επιλογή του s' χρησιμοποιήθηκε ο πρώτος τρόπος από αυτούς που παρουσιάστηκαν και αναλύθηκαν στην Ενότητα 2.5.2, διότι ο πράκτοράς μας θέλουμε να εκπαιδευτεί και να είναι ανταγωνιστικός ενάντια σε οποιοδήποτε αντίπαλο. Ουσιαστικά, η s' είναι η κατάσταση η οποία προβλέπεται ότι θα προκύψει μετά την βέλτιστη κίνηση του πράκτορα και την βέλτιστη κίνηση του αντιπάλου του, σύμφωνα με το δέντρο παιχνιδιού και την αναζήτηση που έκανε ο πράκτορας. Ανανεώνοντας, λοιπόν, τα βάρη του μετά από κάθε κίνηση (ακόμα και εάν αλλάζει επανάληψη, η ανανέωση συνεχίζεται κανονικά), ο παίκτης μας καταλήγει στο τέλος με ένα νέο διάνυσμα βαρών. Αυτό το διάνυσμα είναι το αποτέλεσμα της μάθησης. Σημειώνεται ότι ο αντίπαλος παραμένει σταθερός κατά τη διάρκεια της μάθησης.

Κεφάλαιο 5

Υλοποίηση

Η εργασία αυτή, ο πράκτορας και το γραφικό περιβάλλον, υλοποιήθηκε σε γλώσσα Java [15, 16]. Χρησιμοποιήθηκε η πλατφόρμα Eclipse IDE με εγκατεστημένο το Java JRE 1.8. Το λειτουργικό σύστημα στο οποίο έγινε η υλοποίηση είναι Windows 7 και Windows 10. Το πρόγραμμα έτρεξε σε σύστημα 64bit με 4GB RAM. Οι ελάχιστες απαιτήσεις, ωστόσο, είναι χαμηλότερες. Για την εργασία αυτή δεν προϋπήρχε σχετικός κώδικας, συνεπώς, υλοποιήθηκε όλη από την αρχή.

5.1 Περιβάλλον Παιχνιδιού

Το πρώτο κομμάτι που υλοποιήθηκε είναι αυτό της δημιουργίας του παιχνιδιού και των κανόνων του. Για το λόγο αυτό δημιουργήθηκε η κλάση `Game`, στην οποία η κατάσταση του παιχνιδιού απεικονίζεται σε ένα δισδιάστατο πίνακα. Επίσης, στην κλάση αυτή ορίζονται καθολικά οι σημαντικότεροι παράμετροι όλης της εργασίας (διαστάσεις board, όριο βάθους επέκτασης, ρυθμός μάθησης κ.α), έτσι ώστε τον όλο σύστημα να είναι πλήρως παραμετροποιήσιμο. Την προετοιμασία του παιχνιδιού αναλαμβάνει η συνάρτηση `setup`, στην οποία ορίζεται ο τύπος του κάθε παίκτη (Random, User, Minimax, Pruning), πόσα παιχνίδια θα παιχτούν, ποιος θα παίξει πρώτος κάθε παιχνίδι, τι διαδικασία θα εκτελεστεί (παιχνίδι ή μάθηση) και, τέλος, καλεί τη συνάρτηση `play` που εκτελεί τη διαδικασία που τελικά επιλέγεται.

Η συνάρτηση `play` εκτελεί ένα παιχνίδι. Δίνει σειρά σε κάθε παίκτη και εκείνος επιλέγει την κίνησή του η οποία εκτελείται καλώντας τη συνάρτηση `move`. Η συνάρτηση αυτή πραγματοποιεί όλες τις αναγκαίες μεταβολές, που προκαλεί η κίνηση ενός παίκτη, στον πίνακα του παιχνιδιού.

Για την υλοποίηση των παικτών δημιουργήθηκε η κλάση `Player`, η οποία περιέχει τη συνάρτηση `nextMove` που καλεί κάθε παίκτης όταν είναι η σειρά του να παίξει. Για να είναι πιο ξεκάθαρος ο τύπος του παίχτη, η κλάση `Player` είχε σαν υποκλάσεις τις `Random, User, MinMax` και `Pruning`, όπου η καθεμία είχε τη δική της `nextMove`.

Όταν, λοιπόν, το παιχνίδι τελειώσει εκτυπώνονται στην κονσόλα τα αποτελέσματα.

5.2 Minimax και α-β Pruning

Αφού υλοποιήθηκε το περιβάλλον του παιχνιδιού, συνέχεια δόθηκε με την εφαρμογή των αλγορίθμων Minimax και Minimax με α–β Pruning στο παιχνίδι. Στις αντίστοιχες

υποκλάσεις της κλάσης `Player`, δημιουργήθηκαν οι συναρτήσεις `MaxMove` και `MinMove`. Η `nextMove` καλεί τη `MaxMove` η οποία, αν δεν έχει φτάσει στο μέγιστο όριο βάθους επέκτασης D ή σε τερματική κατάσταση του παιχνιδιού, καλεί αναδρομικά τη `MinMove`. Κάτι αντίστοιχο συμβαίνει και στη `MinMove`. Τονίζεται ότι στην εργασία χρησιμοποιείται μόνο η υποκλάση `Pruning`, διότι αποτελεί βελτίωση της κλάσης `MinMax`.

Αυτές οι συναρτήσεις είναι υπεύθυνες για την κατασκευή του δέντρου του παιχνιδιού. Κάθε μία από αυτές εκτελεί μία κίνηση και καλεί την άλλη πηγαίνοντας ένα επίπεδο κάτω. Όταν επιστρέφεται μία κλήση από χαμηλότερο επίπεδο, κάθε συνάρτηση έχει την ευθύνη να αναιρέσει την κίνηση που έκανε και να συνεχίσει με την επόμενη, έτσι ώστε να συνεχιστεί η επέκταση άλλου κόμβου του δέντρου.

5.3 Αξιολόγηση

Όταν οι συναρτήσεις `MaxMove` και `MinMove` φτάσουν σε φύλλο του δέντρου, δηλαδή σε τερματική κατάσταση του παιχνιδιού ή στο μέγιστο όριο βάθους επέκτασης, τότε η κατάσταση στην οποία αντιστοιχεί το φύλλο αξιολογείται από τη συνάρτηση `evaluate` που βρίσκεται στην κλάση `Evaluation`.

Η συνάρτηση αυτή υπολογίζει την τιμή κάθε χαρακτηριστικού σε μία κατάσταση, κανονικοποιεί όλα τα χαρακτηριστικά και χρησιμοποιώντας το διάνυσμα βαρών που έχει οριστεί υπολογίζει την αξιολόγηση της κατάστασης. Μετά επιστρέφει την τιμή της αξιολόγησης στο φύλλο του δέντρου και έτσι συνεχίζεται η δημιουργία του δέντρου παιχνιδιού. Στο τέλος, με βάση τον αλγόριθμο του Minimax με α–β Pruning, επιλέγεται η τελική κίνηση του πράκτορά μας, η οποία και εκτελείται.

5.4 TD-Learning

Στο επόμενο στάδιο υλοποιήθηκε η διαδικασία της μάθησης. Στην κλάση `Learning` δημιουργήθηκε η συνάρτηση `update`, η οποία υλοποιεί την ανανέωση των βαρών σύμφωνα με τον αλγόριθμο TD-Learning.

Η συνάρτηση αυτή καλείται από την κλάση `Game` και τη συνάρτηση `play` μετά από κάθε κίνηση του πράκτορά μας, αν και μόνο αν έχει οριστεί από τη συνάρτηση `setup` ότι πρόκειται για διαδικασία μάθησης. Η κλάση `Game` υπολογίζει και παρέχει στην κλήση της `update` τα απαραίτητα ορίσματα για την ανανέωση.

Το διάνυσμα βαρών, ωστόσο δεν το παρέχει η κλάση `Game`. Όπως αναφέρθηκε στην προηγούμενη ενότητα, το διάνυσμα αυτό ορίζεται στην κλάση `Evaluation`. Για το λόγο αυτό δημιουργήθηκε η συνάρτηση `getWeightsNorth` (αντίστοιχα `getWeightsSouth`), η οποία μας επιστρέφει το διάνυσμα βαρών του αντίστοιχου παίκτη. Με αυτό το κομμάτι υλοποίησης ολοκληρώνεται και εκπληρώνεται ο βασικός στόχος της εργασίας.

5.5 Γραφικό Περιβάλλον

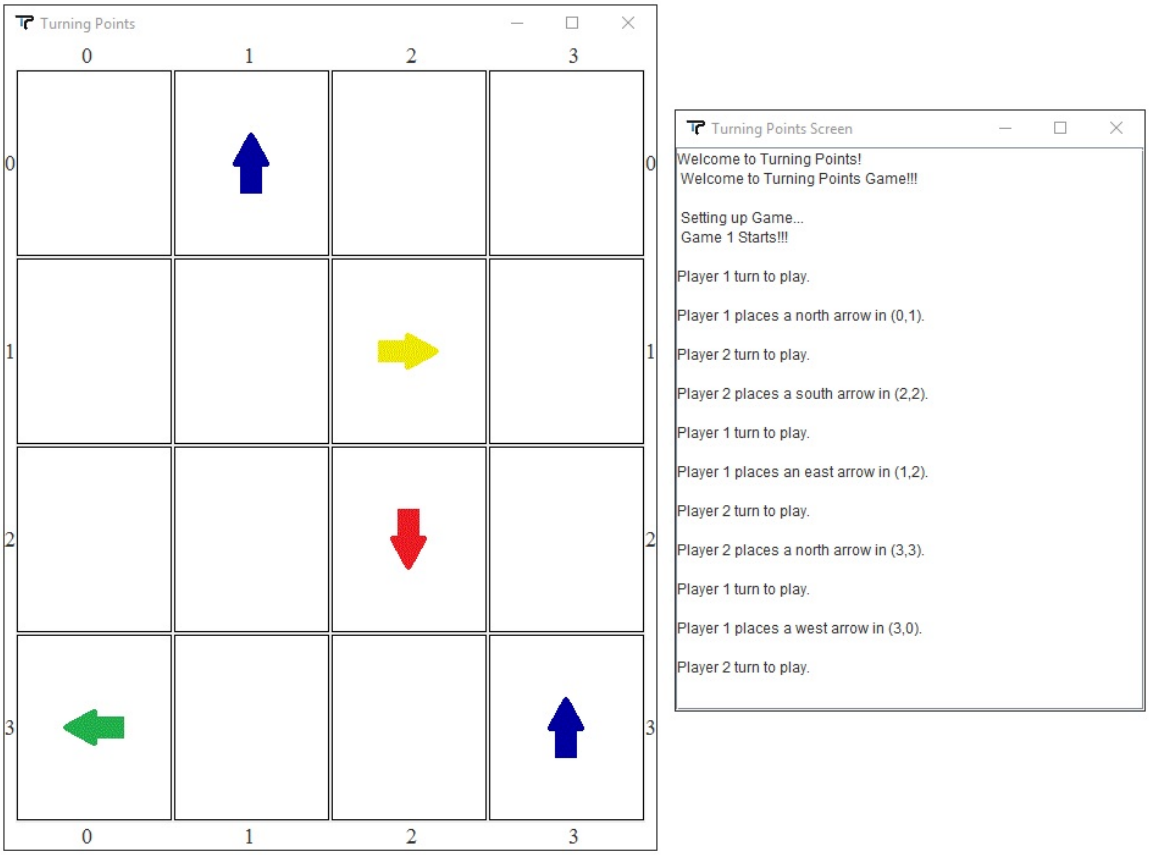
Για να μπορεί το παιχνίδι να παιχτεί από κάποιον χρήστη ενάντια στον υπολογιστή ή μεταξύ δύο χρηστών, ήταν απαραίτητη η ύπαρξη γραφικού περιβάλλοντος.

Το γραφικό περιβάλλον που δημιουργήθηκε αποτελείται από δύο οθόνες. Στη μία βλέπουμε το παιχνίδι και στην άλλη καταγράφεται ποιος παίκτης παίζει και ποια κίνηση επέλεξε. Οι κινήσεις, όταν παίζει κάποιος χρήστης, δίνονται από την κονσόλα της πλατφόρμας.

Η οθόνη που περιλαμβάνει το παιχνίδι κατασκευάζεται στην κλάση `Board`, η οποία είναι επέκταση της κλάσης `JFrame`. Μέσα στην κλάση αυτή δηλώνονται όλα τα γραφικά στοιχεία που χρησιμοποιούνται για την απεικόνιση του παιχνιδιού και γίνεται η σωστή διάταξη του board. Επιπλέον, δημιουργείται η συνάρτηση `move`, η οποία μας δείχνει γραφικά όλες τις εναλλαγές που επιφέρει μία κίνηση που μόλις εκτελέστηκε. Για να μπορεί να εμφανιστεί, όμως, το γραφικό στοιχείο, πρέπει να ενεργοποιηθεί από τη συνάρτηση `setup` της κλάσης `Game`.

Όσον αφορά τη δεύτερη οθόνη, είναι κι αυτή μία επέκταση της `JFrame` και ονομάζεται `Screen`. Αυτή η κλάση δημιουργεί ένα παράθυρο εξόδου και περιέχει τη συνάρτηση `print`. Και αυτό το παράθυρο πρέπει να ενεργοποιηθεί από τη `setup` για να εμφανιστεί.

Επομένως, μετά από κάθε κίνηση που πραγματοποιείται στο παιχνίδι, καλούνται επιπλέον η συνάρτηση `move` της κλάσης `Board` και η συνάρτηση `print` της κλάσης `Screen` με τα κατάλληλα ορίσματα. Στο Σχήμα 5.1 φαίνεται η πλήρης γραφική απεικόνιση του παιχνιδιού Turning Points.



Σχήμα 5.1: *Γραφικό Περιβάλλον για το Παιχνίδι Turning Points.*

Κεφάλαιο 6

Αποτελέσματα

Η βελτίωση και η εκμάθηση ενός πράκτορα να παίζει αποδοτικά ένα παιχνίδι είναι αποτέλεσμα μιας περίπλοκης και χρονοβόρας πειραματικής διαδικασίας. Για το λόγο αυτό πρέπει να δίνεται η δέουσα προσοχή στην εξαγωγή αποτελεσμάτων και στην ανάλυση αυτών.

6.1 Πειραματική Διαδικασία

Εκτελέστηκαν αρκετά πειράματα μάθησης για να εξαχθούν οι καλύτεροι παίκτες. Είναι αξιοσημείωτο ότι οι καλύτεροι που προέκυψαν είναι υπερβολικά δύσκολοι αντίπαλοι για τον άνθρωπο. Στην πειραματική διαδικασία ο πράκτοράς μας καλείται να εκπαιδευτεί ενάντια στους παρακάτω αντιπάλους:

- R (επιλέγει κινήσεις εντελώς τυχαία)
- S (αξιολογεί κινήσεις με βάση μόνο το score)
- MP (My Player – ο παίκτης που δημιουργήθηκε χειροκίνητα, Ενότητα 4.3)

Σε κάθε πείραμα που εκτελείται, το διάνυσμα βαρών του πράκτορά μας αρχικοποιείται σε μία από τις εξής επιλογές:

- Z (όλα τα βάρη αρχικοποιούνται με μηδενικά)
- S (όλα τα βάρη αρχικοποιούνται με μηδενικά, με εξαίρεση $w_1 = 1$)
- MP (τα βάρη αρχικοποιούνται στις τιμές που δημιουργήθηκαν χειροκίνητα)

Το κάθε πείραμα αποτελεί μία διαδικασία μάθησης. Στη διαδικασία αυτή ο πράκτοράς μας αντιμετωπίζει σε 10000 παιχνίδια τον αντίπαλο του. Τα παιχνίδια αυτά πραγματοποιούνται σε board 4×4 και ο πράκτοράς μας έχει όριο βάθους επέκτασης $D = 4$. Ανά 25 παιχνίδια οι δύο παίκτες αλλάζουν πλευρά, με σκοπό την πληρότητα της μάθησης και από τις δύο πλευρές του board. Ο ρυθμός μάθησης ξεκινάει από $\alpha = 0.1$ και υποδιπλασιάζεται κάθε 1000 επαναλήψεις. Κάθε 400 επαναλήψεις αποθηκεύεται το τρέχον διάνυσμα βαρών του πράκτορά μας, το οποίο ονομάζεται στιγμιότυπο. Τέλος, για να παρακολουθήσουμε τη σύγκλιση του διανύσματος βαρών, κάθε 200 επαναλήψεις υπολογίζεται και αποθηκεύεται η Ευκλείδια απόσταση (L_2 norm) του τρέχοντος διανύσματος βαρών και του προηγούμενου διανύσματος πριν από 200 επαναλήψεις.

Βάσει του παραπάνω σχεδιασμού ο πράκτορας επικεντρώνεται σε συγκεκριμένους τρόπους επιλογής κινήσεων, οι οποίοι ενδέχεται να μην καλύπτουν όλο το εύρος των πιθανών κινήσεων. Κρίθηκε, λοιπόν, αναγκαίο να προστεθεί και η δυνατότητα εξερεύνησης εντός της διαδικασίας. Οπότε, με πιθανότητα $e = 0.15$ ο παίκτης μας δεν ακολουθούσε την κίνηση που του υπεδείκνυε το δέντρο παιχνιδιού, αλλά πραγματοποιούσε κάποια τυχαία κίνηση. Έτσι, ο παίκτης μας περνούσε και από καταστάσεις που ίσως να μην αντιμετώπιζε ποτέ χωρίς εξερεύνηση, με αποτελέσμα να αποκτά εμπειρία σε μεγαλύτερο μέρος του χώρου των πιθανών καταστάσεων του παιχνιδιού. Τονίζεται ότι η τυχαία επιλογή κίνησης δεν έχει αρνητική επίδραση στη μάθηση, καθώς η ενημέρωση TD-Learning αξιοποιεί μεμονωμένες μεταβάσεις στο χώρο καταστάσεων.

Κατά τη διάρκεια ενός πειράματος γινόταν ο έλεγχος της απόδοσης του πράκτορα. Κάθε 400 παιχνίδια, το αποθηκευμένο στιγμιότυπο του πράκτορά μας από τη διαδικασία της μάθησης καλείται να αντιμετωπίσει τον αντίπαλο σε 100 παιχνίδια (50 από κάθε πλευρά) με σταθερό διάνυσμα βαρών (χωρίς ενημέρωση βαρών) και χωρίς εξερεύνηση. Για κάθε νίκη βαθμολογείται με τρεις πόντους, για κάθε ισοπαλία με έναν πόντο και για κάθε ήττα με μηδέν πόντους. Άρα, το ανώτατο όριο απόδοσης του παίκτη είναι το 300 και το κατώτατο το μηδέν. Η συγκομιδή των πόντων του πράκτορά μας είναι μία ένδειξη της απόδοσής του. Ο έλεγχος απόδοσης με πολλαπλά παιχνίδια κρίνεται απαραίτητος λόγω της τυχαίας επίλυσης ισοβαθμιών μεταξύ βέλτιστων κινήσεων.

Για κάθε πείραμα, στην επόμενη ενότητα, παρουσιάζεται η απόδοση του πράκτορά μας και η σύγκλιση του διανύσματος βαρών. Σημειώνεται ότι ο παίκτης που προκύπτει από μία διαδικασία μάθησης φέρει το όνομα *Αρχικοποίηση vs Αντίπαλος*. Εάν έχει χρησιμοποιηθεί εξερεύνηση σε ένα πείραμα στο όνομα του παίκτη προστίθεται και ένα e στο τέλος.

6.2 Αποτελέσματα Πειραμάτων

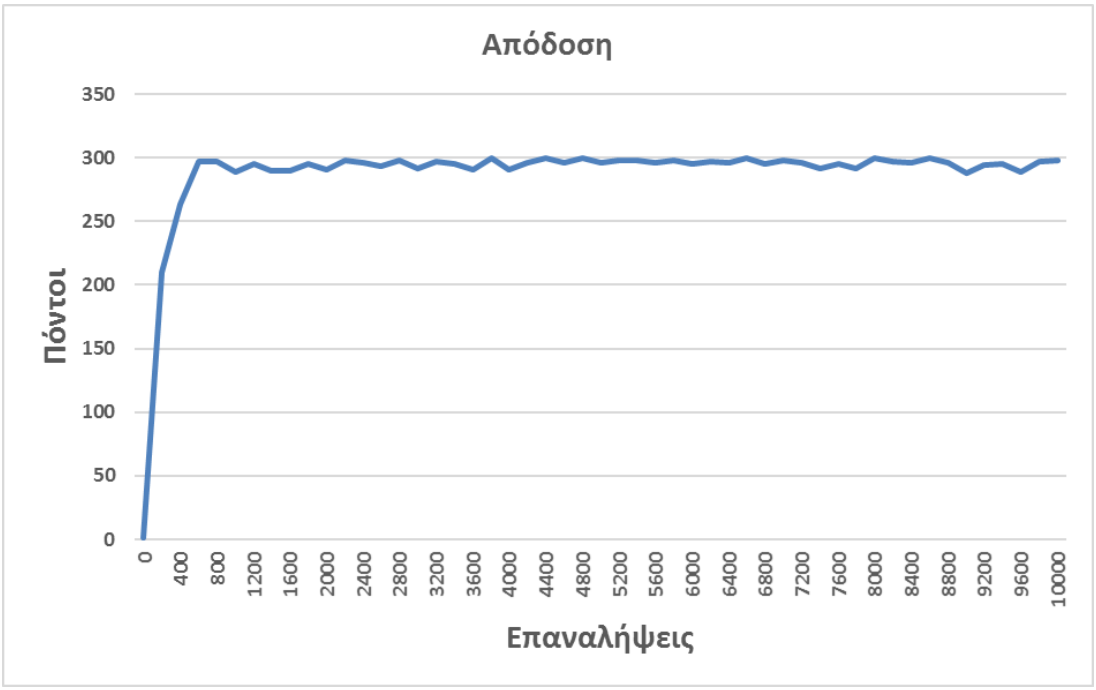
Το πρώτο πείραμα έγινε με τον πράκτορά μας ως Z και με αντίπαλο τον παίκτη R. Αυτό το πείραμα έγινε για να επιβεβαιώσουμε τη σωστή λειτουργία του αλγορίθμου μάθησης. Τα αποτελέσματα αυτού του πειράματος φαίνονται στο Σχήμα 6.1. Όπως φαίνεται στο σχήμα, ο παίκτης μας, αν και με μηδενικά βάρη στην αρχή, γρήγορα συγκλίνει και γίνεται πλήρως αποτελεσματικός ενάντια στον παίκτη R. Αυτό είναι λογικό, αφού θεωρητικά ο R είναι πολύ εύκολος αντίπαλος.

Τα πειράματα συνεχίστηκαν και τέθηκε ως αντίπαλος ο παίκτης S. Ο πράκτοράς μας τον αντιμετώπισε με όλες τις πιθανές αρχικοποιήσεις, δηλαδή ως Z, S και MP. Τα αποτελέσματα της μάθησης για αυτές τις τρεις περιπτώσεις φαίνονται στα Σχήματα 6.2, 6.3 και 6.4, αντίστοιχα. Σε αυτό το κομμάτι πειραμάτων παρατηρείται γρήγορη σύγκλιση του διανύσματος βαρών και σχεδόν άριστη απόδοση. Στις δύο πρώτες περιπτώσεις (ZvsS και SvsS) η αύξηση απόδοσης είναι πιο γρήγορη απ΄ ότι στην τρίτη (MPvsS). Αυτό οφείλεται στο γεγονός ότι στην τρίτη περίπτωση το αρχικό διάνυσμα βαρών έχει αρχικές τιμές μη μηδενικές, οι οποίες προσπαθούν να προσαρμοστούν στις σωστές τιμές κατά τη διάρκεια της μάθησης. Στις άλλες δύο περιπτώσεις υπάρχουν πολλά μηδενικά βάρη, τα οποία είναι πιο εύκολο να προσαρμοστούν στις σωστές τιμές.

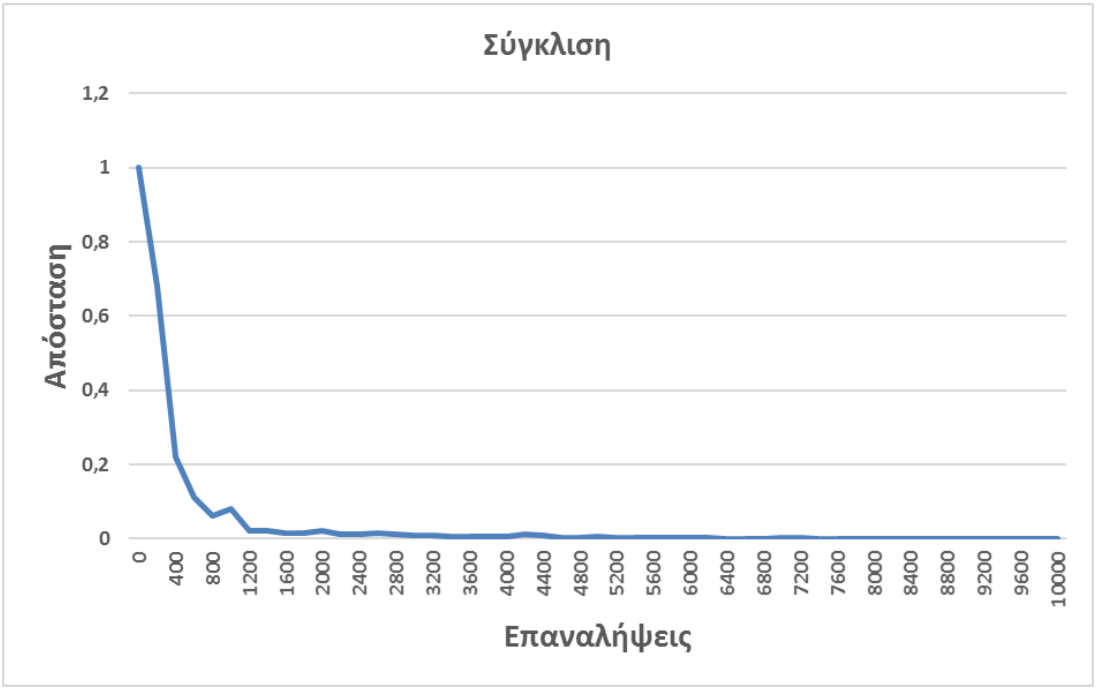
(α΄) *Απόδοση.*

(β΄) *Σύγκλιση διανύσματος βαρών.*

Σχήμα 6.1: *Παίκτης ZvsR.*



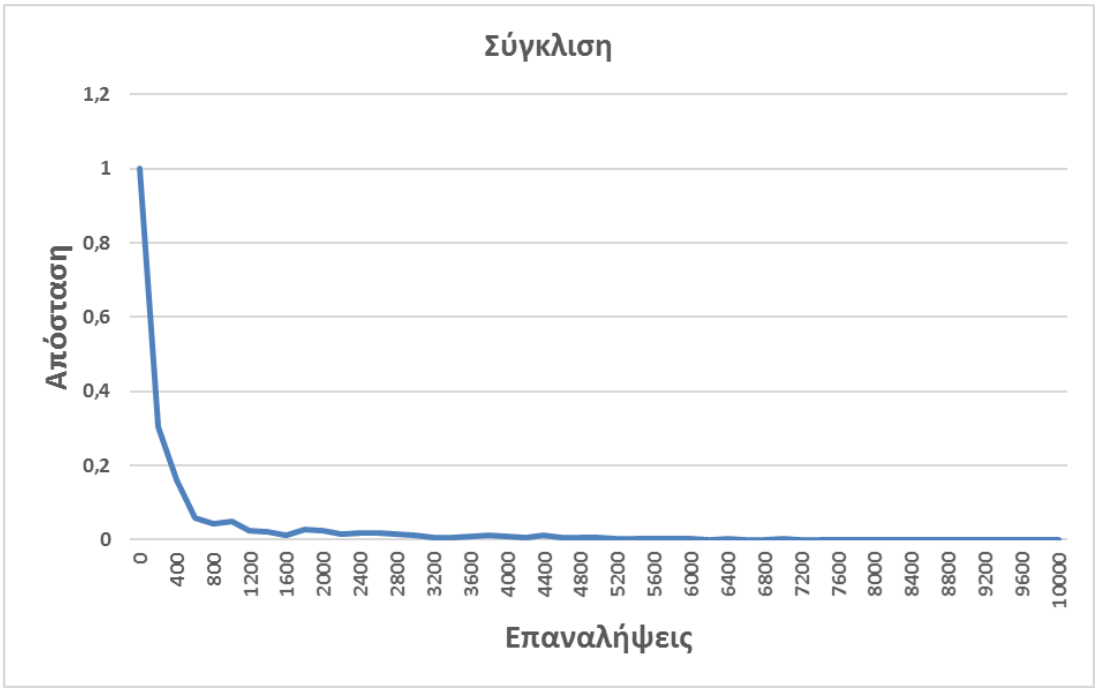
(α′) *Απόδοση.*



(β′) *Σύγκλιση διανύσματος βαρών.*

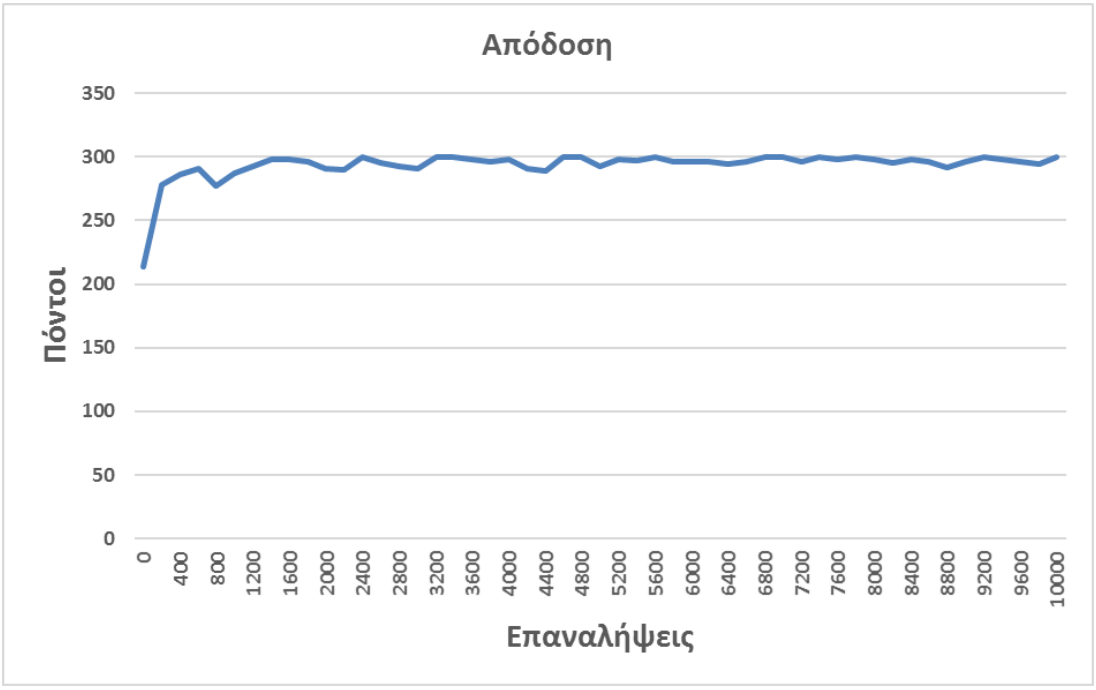
Σχήμα 6.2: *Παίκτης ZvsS.*

(α΄) *Απόδοση.*

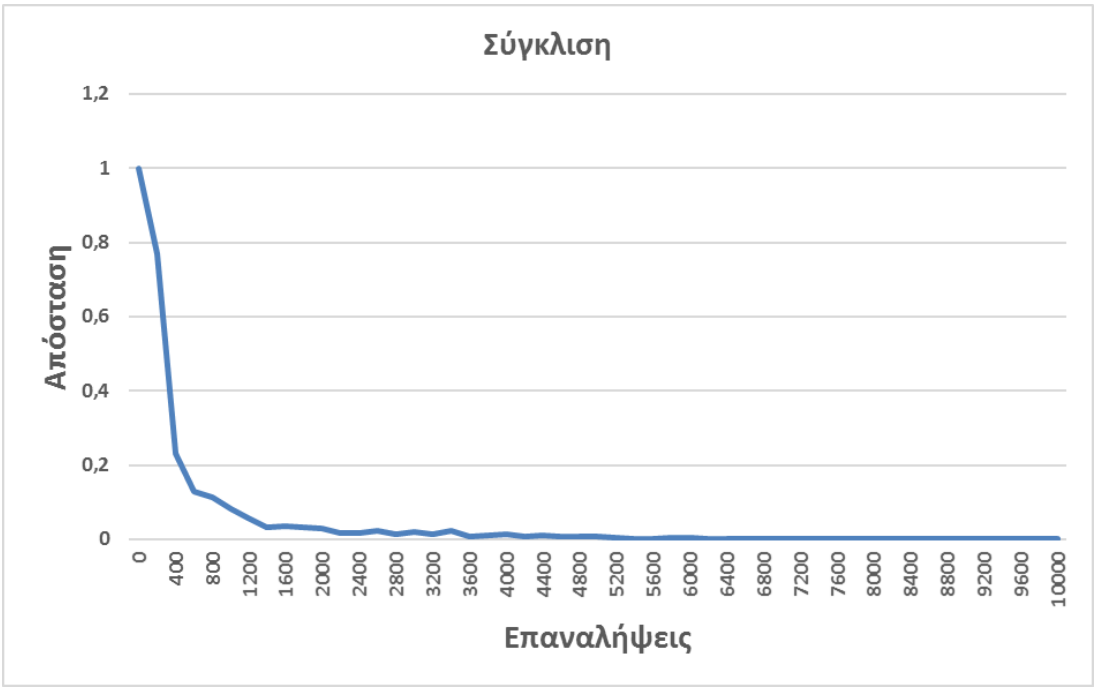


(β΄) *Σύγκλιση διανύσματος βαρών.*

Σχήμα 6.3: *Παίκτης SvsS.*



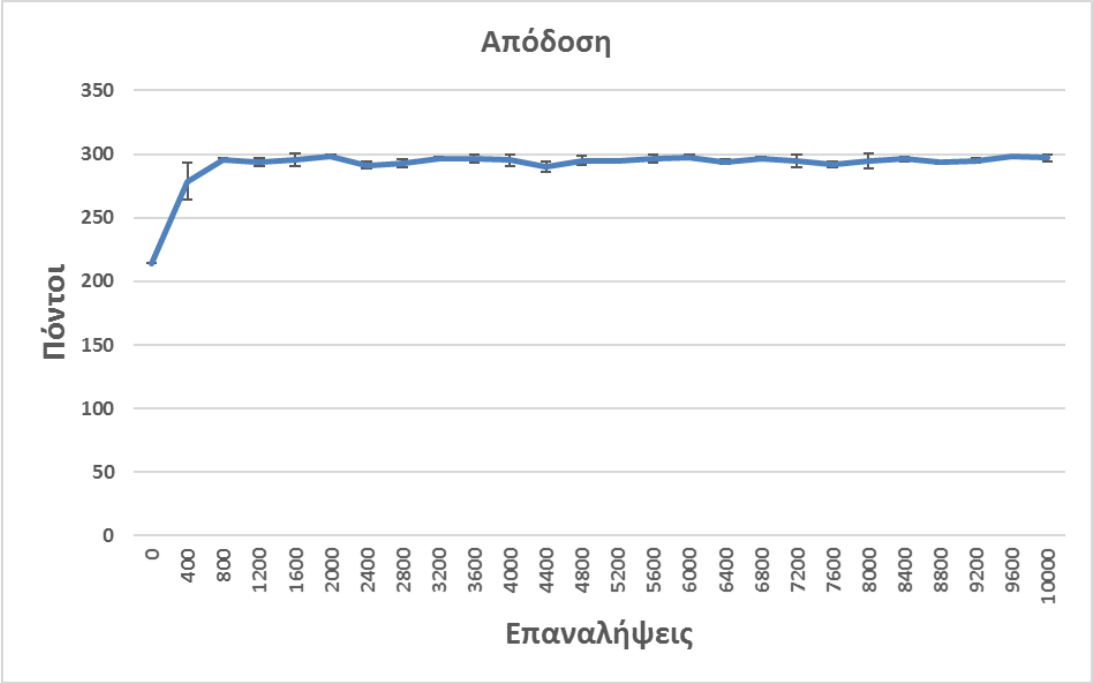
(α′) *Απόδοση.*



(β′) *Σύγκλιση διανύσματος βαρών.*

Σχήμα 6.4: *Παίκτης MPvsS.*

Επειδή, όπως αναφέρθηκε παραπάνω, στην περίπτωση MPvsS έχουμε πιο αργή αύξηση απόδοσης και σταθεροποίησή της, χρησιμοποιείται εξερεύνηση και επαναλαμβάνεται το πείραμα τρεις φορές. Στο Σχήμα 6.5 φαίνεται ο μέσος όρος της απόδοσης από αυτά τα πειράματα, όπως ακόμα και η διασπορά των τιμών από τις εκτελέσεις του πειράματος. Από τη διασπορά καταλαβαίνουμε ότι σε κάθε εκτέλεση του πειράματος ο παίκτης ήταν εξίσου αποτελεσματικός. Η σύγκλιση είναι όμοια με αυτή στο Σχήμα 6.4. Πρέπει να σημειωθεί ότι η σχεδόν άριστη απόδοση στα παραπάνω πειραμάτα επιτυγχάνεται, όχι μόνο λόγω της σωστής μάθησης, άλλα επειδή ό αντίπαλος S αποδεικνύεται σχετικά εύκολος.



Σχήμα 6.5: *Παίκτης MPvsSe.*

Στον επόμενο κύκλο πειραμάτων, αντίπαλος τέθηκε ο MP. Αυτή τη φορά, ο πράκτοράς μας δεν αντιμετώπισε τον αντίπαλό του με όλες τις πιθανές αρχικοποιήσεις. Τον αντιμετώπισε ως Z και MP. Η εξαίρεση του S έγινε διότι ο Z και ο S κρίθηκαν ισάξιοι αντίπαλοι για τον MP. Πιο συγκεκριμένα, και οι δύο έχαναν απέναντί του με ποσοστό μεγαλύτερο του 90%. Οπότε, για λόγους εξοικονόμησης χρόνου επιλέχτηκε μόνο ο Z, ο οποίος (λόγω των μηδενικών βαρών) θα προσέγγιζε ευκολότερα το βέλτιστο παίκτη. Τα αποτελέσματα της μάθησης για τις δύο αυτές περιπτώσεις φαίνονται στα Σχήματα 6.6 και 6.7. Σε αυτόν το κύκλο πειραμάτων παρατηρούνται διαφορετικά και ενδιαφέροντα αποτελέσματα σε σχέση με τον προηγούμενο.

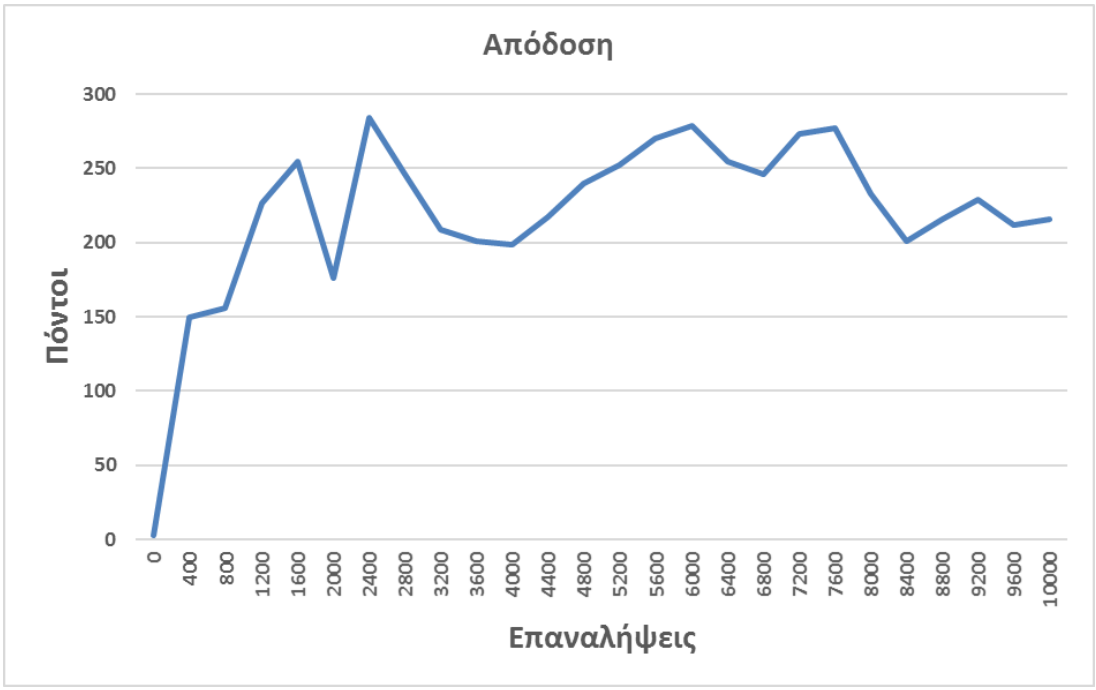
Ο παίκτης ZvsMP φαίνεται ότι συγκλίνει στις τιμές των βαρών του γρήγορα. Η απόδοσή του, ναι μεν φτάνει σε κάποια σημεία της μάθησης σε σχεδόν άριστο επίπεδο (ειδικά αν αναλογιστεί κανείς ότι είχε απέναντί του έναν αξιόλογο παίκτη), όμως δε σταθεροποιείται. Αυτό οφείλεται στο γεγονός ότι ακόμα και ελάχιστες μεταβολές του διανύσματος βαρών του, μπορούν να τον οδηγήσουν σε αλλαγές της απόδοσής του ενάντια σε έναν αξιόλογο παίκτη. Για το λόγο αυτό, η απόδοσή του αυξάνεται και μειώνεται συνεχώς. Μία σημαντική παρατήρηση είναι ότι η καμπύλη της απόδοσής του έχει την τάση να σταθεροποιηθεί προς το τέλος του πειράματος.

Ο παίκτης MPvsMP έχει κάποιες εναλλαγές στη σύγκλιση των βαρών του στο διάστημα επαναλήψεων [400,1200], όμως μετά συγκλίνουν κανονικά. Ενδιαφέρον, ωστόσο, προκαλεί η καμπύλη αποδοσής του, καθώς έχει πολλές εναλλαγές και δε φαίνεται να σταθεροποιείται ακόμα και όταν τα βάρη έχουν σχεδόν συγκλίνει. Επίσης, από το σχήμα φαίνεται ότι δεν υπάρχει τάση σταθεροποίησης στην καμπύλη απόδοσης. Τα παραπάνω αποτελέσματα προκύπτουν, λόγω του ότι ο πράκτοράς μας, αρχικά, αντιμετωπίζει τον κλώνο του. Έτσι, δύο αξιόλογοι παίκτες έρχονται αντιμέτωποι. Στην αρχή της μάθησης είναι ισάξιοι, γεγονός που αποφέρει μοιρασμένες νίκες και ήττες. Ο πράκτοράς μας, λοιπόν, τροποποιεί συνεχώς τις τιμές των βαρών του, εναλλάσοντας κατευθύνσεις (αύξηση/μείωση). Αυτό έχει ως αποτέλεσμα να υπάρχουν εναλλαγές και στην απόδοσή του. Ωστόσο, και υπό αυτές τις συνθήκες, σε κάποιες στιγμές της μάθησης αγγίζει την άριστη απόδοση.

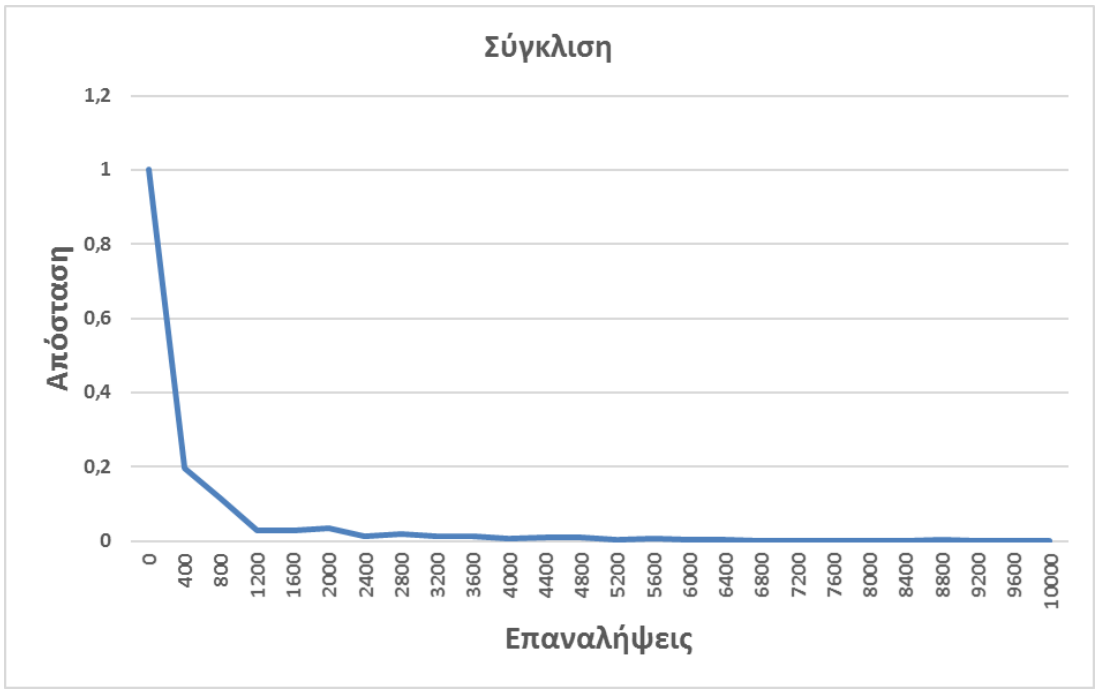
Τα αποτελέσματα των δύο τελευταίων πειραμάτων (ενάντια στον MP) ήταν ασταθή, με αποτελέσματα να οδηγήσουν στη διεξαγωγή δύο επιπλέον πειραμάτων. Στα πειράματα αυτά, επαναλαμβάνονται τα προηγούμενα με τη διαφορά ότι χρησιμοποιείται εξερεύνηση. Το κάθε πείραμα εκτελείται 10 φορές. Στις καμπύλες απόδοσης και σύγκλισης διανύσματος βαρών απεικονίζονται οι μέσοι όροι από τις 10 εκτελέσεις των πειραμάτων, καθώς και οι διασπορά των τιμών. Επιπλέον, με διακεκομμένες γραμμές φαίνονται οι καμπύλες μέγιστων και ελάχιστων τιμών. Οι παίκτες που προκύπτουν από αυτή τη διαδικασία είναι οι ZvsMPe και MPvsMPe, των οποίων οι καμπύλες απόδοσης και σύγκλισης διανύσματος βαρών φαίνονται στα Σχήματα 6.8 και 6.9, αντίστοιχα.

Στην πρώτη περίπτωση (ZvsMPe) παρατηρείται ότι υπάρχει σταθεροποίηση της απόδοσης και σύγκλιση του διανύσματος βαρών. Όσο περνάνε οι επαναλήψεις υπάρχει σταθεροποίηση της απόδοσης, κάτι το οποίο φαίνεται και από την καμπύλη των μέγιστων τιμών. Όσον αφορά τη διασπορά, μειώνεται όσο αυξάνονται οι επαναλήψεις. Αυτό αποδεικνύει ότι γενικά ο παίκτης μας έχει την τάση να βελτιώνεται, παρόλο που μεμονωμένα πειράματα μπορεί να μην οδηγήσουν σε καλό αποτέλεσμα. Σε αυτήν την περίπτωση, λοιπόν, δικαιώνεται η επιλογή επανεκτέλεσης του πειράματος με εξερεύνηση, αφού οι καμπύλες του ZvsMPe είναι δίνουν μία πιο γενική και ολοκληρωμένη εικόνα από αυτές του ZvsMP.

Στη δεύτερη περίπτωση (MPvsMPe) η απόδοση είναι περισσότερο σταθεροποιημένη απ΄ ότι στην περίπτωση του MPvsMP και η σύγκλιση είναι ομαλή. Η σταθεροποίηση της απόδοσης φαίνεται από τη μείωση των εναλλαγών στην απόδοση. Η διασπορά των τιμών, ωστόσο, είναι μεγάλη σε όλη τη διάρκεια της μάθησης. Αυτό συμβαίνει λόγω των χαμηλών ελάχιστων τιμών και των υψηλών μέγιστων. Το γεγονός αυτό υποδεικνύει ότι ο παίκτης που προκύπτει από την μάθηση, ναι μεν είναι καλύτερος από τον αντίπαλό του, αλλά λόγω του ότι έχουν πολλά όμοια στοιχεία (αρχικά ήταν ίδιοι) πολλές φορές δυσκολεύεται να τον κερδίσει. Και σε αυτήν την περίπτωση υπάρχει βελτιωμένη εικόνα σε σχέση με το αρχικό πείραμα.

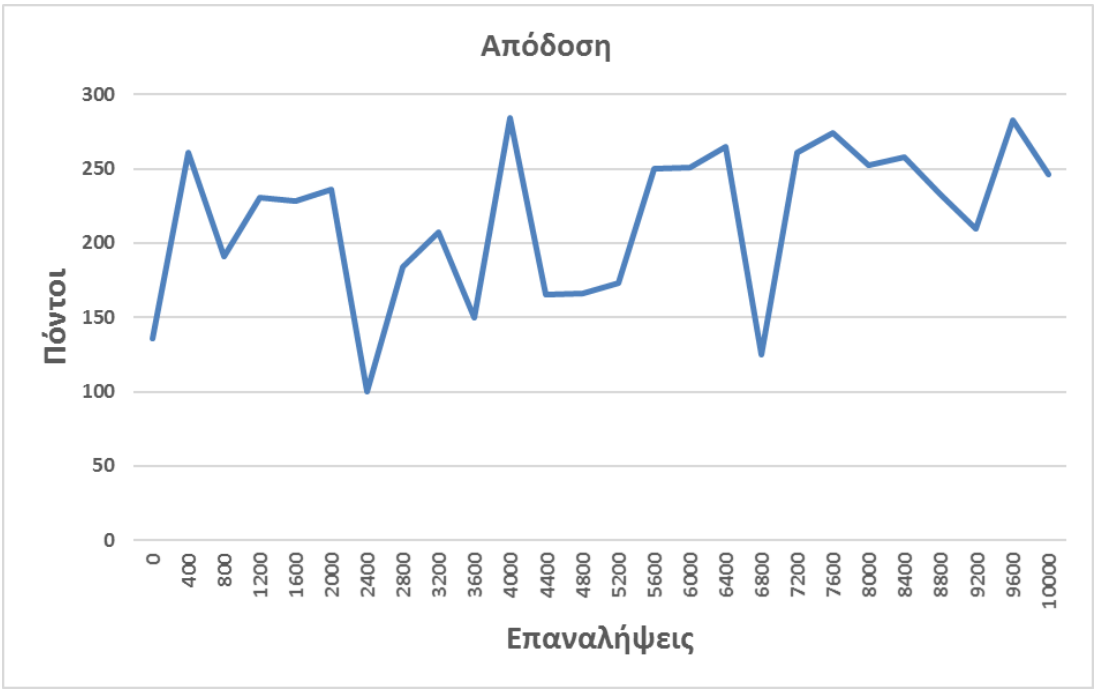


(α΄) *Απόδοση.*



(β΄) *Σύγκλιση διανύσματος βαρών.*

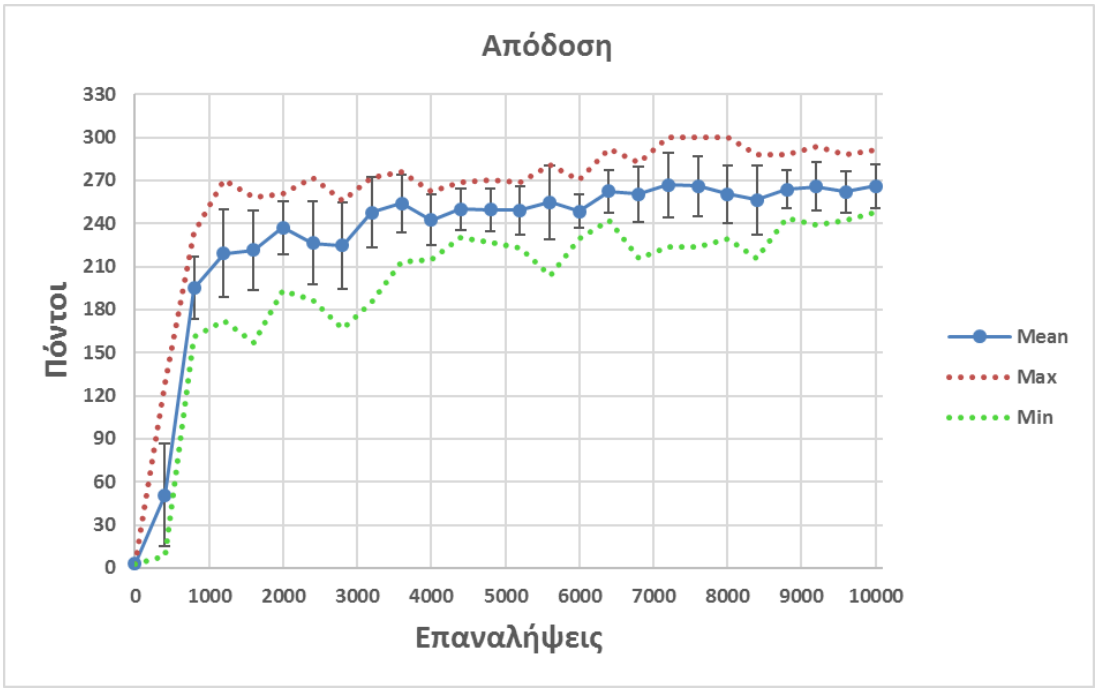
Σχήμα 6.6: *Παίκτης ZvsMP.*



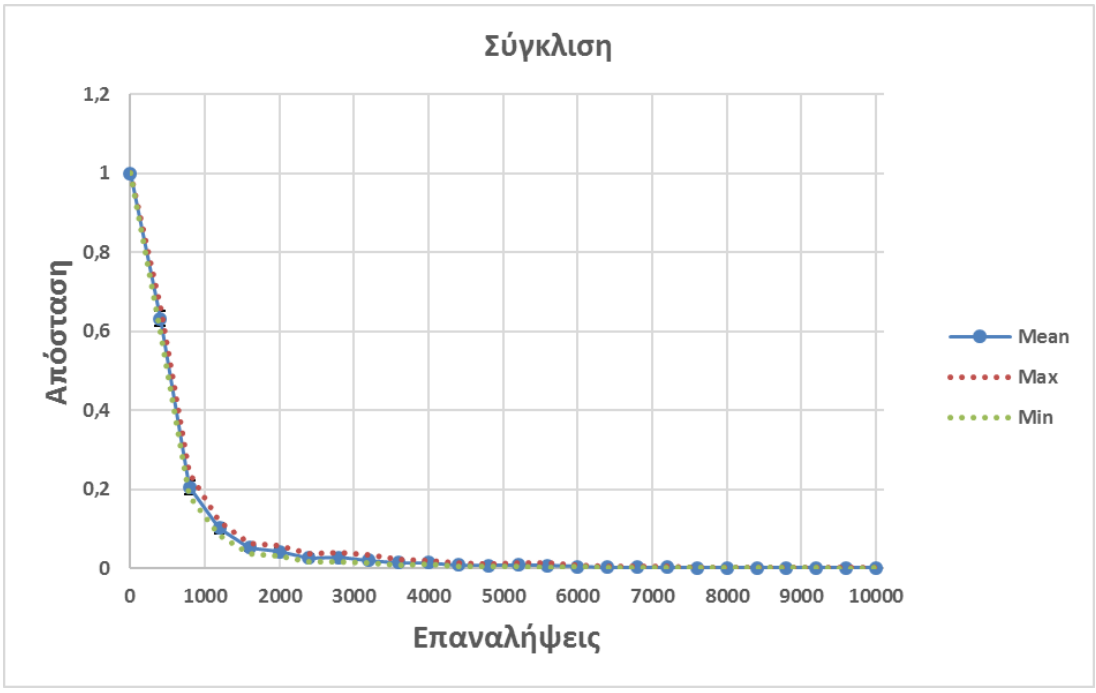
(α′) *Απόδοση.*

(β′) *Σύγκλιση διανύσματος βαρών.*

Σχήμα 6.7: *Παίκτης MPvsMP.*

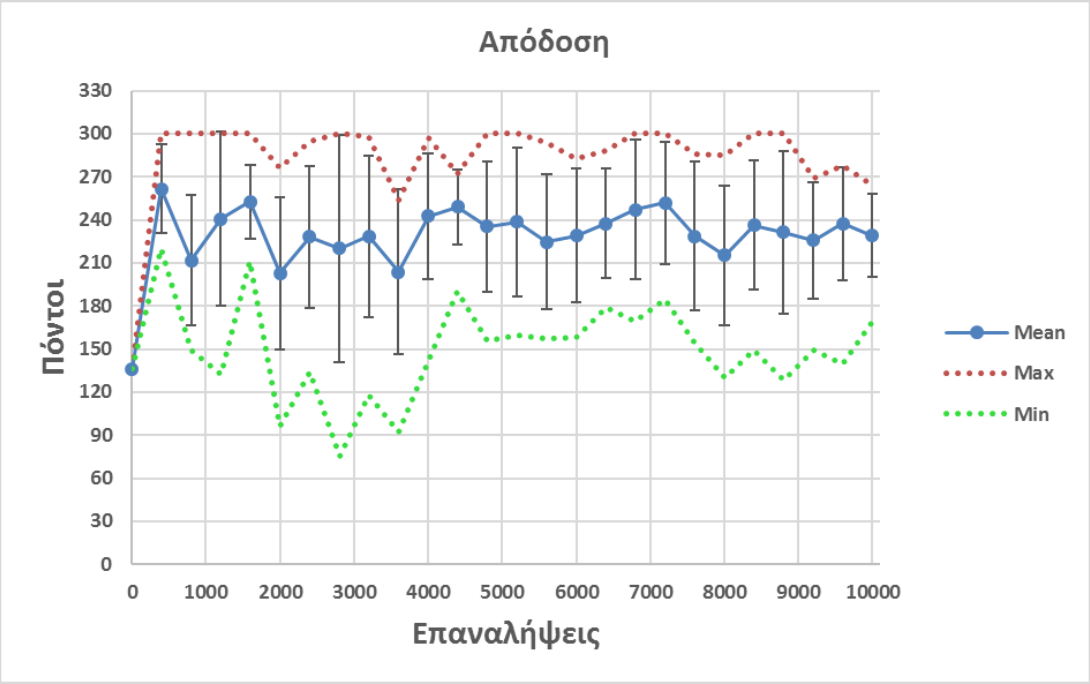


(α΄) *Απόδοση.*

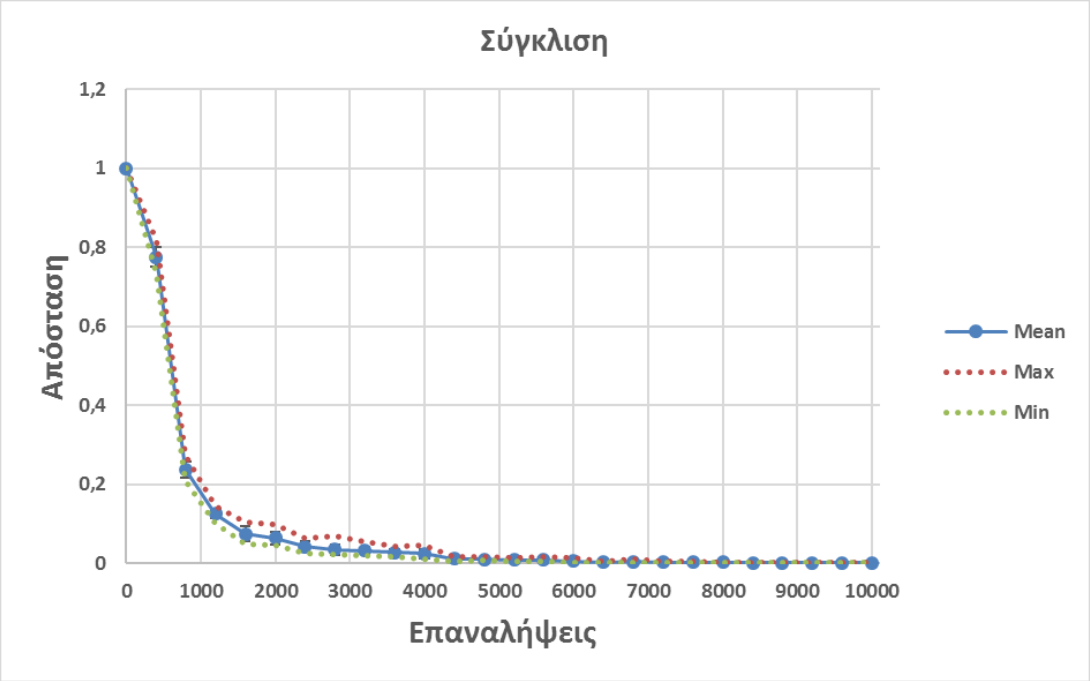


(β΄) *Σύγκλιση διανύσματος βαρών.*

Σχήμα 6.8: *Παίκτης ZvsMPe.*



(α΄) *Απόδοση.*



(β΄) *Σύγκλιση διανύσματος βαρών.*

Σχήμα 6.9: *Παίκτης MPvsMPe.*

6.3 Αξιολόγηση Αποτελεσμάτων

Κάθε πείραμα που εκτελέστηκε είχε σκοπό την εκμάθηση του πράκτορά μας και την εξαγωγή ενός διανυσμάτος βαρών (παίκτη) όσο το δυνατόν καλύτερο. Σε πολλά από τα πειράματα ο πράκτοράς μας είχε περισσότερα από ένα στιγμιότυπα που έφταναν απόδοση ίση με 300. Συνεπώς, επιλέχθηκε το στιγμιότυπο εκείνο που θα αντιπροσώπευε καλύτερα τον παίκτη κάθε πειράματος, δηλαδή το στιγμιότυπο εκείνο που είχε τη μέγιστη απόδοση και η διασπορά τιμών ήταν η ελάχιστη. Σε περίπτωση που υπήρχαν περισσότερα από ένα ισάξια στιγμιότυπα γινόταν τυχαία επιλογή μεταξύ τους. Στις περιπτώσεις που δεν υπολογίζεται η διασπορά επιλέγεται όποιο στιγμιότυπο έχει την καλύτερη απόδοση.

Συνοψίζοντας τα αποτελέσματα της πειραματικής διαδικασίας, προκύπτουν 10 παίκτες συμπεριλαμβανομένου του MP. Στον Πίνακα 6.1 καταγράφονται οι πάικτες αυτοί, καθώς και οι παράμετροι από τις οποίες προέκυψαν. Επίσης στον Πίνακα 6.2 καταγράφεται το τελικό διάνυσμα βαρών του κάθε παίκτη παίζοντας ως *Βόρειος*. Για το αντίστοιχο διάνυσμα κάθε παίκτη ως *Νότιος* αντιστρέφονται τα βάρη των χαρακτηριστικών που αφορούν ανατολικό/δυτικό προσανατολισμό και τα βάρη των χαρακτηριστικών που αφορούν βόρειο/νότιο προσανατολισμό.

Για την αξιολόγηση των αποτελεσμάτων οι παίκτες συμμετέχουν σε ένα τουρνουά στο οποίο κάθε παίκτης αντιμετωπίζει όλους τους υπόλοιπους σε παρτίδες των 100 παιχνιδιών (50 από κάθε πλευρά). Η τελική βαθμολογία κάθε παίκτη είναι το άθροισμα των πόντων που συνέλεξε. Το τουρνουά διεξάγεται σε board διαστάσεων 4×4 και έχει όριο βάθους επέκτασης $D = 4$. Οι πέντε πρώτοι στη βαθμολογία παραμένουν στην διαδικασία αξιολόγησης, ενώ οι υπόλοιποι αποκλείονται. Στη συνέχεια, οι πέντε που παρέμειναν συμμετέχουν σε δύο ακόμα τουρνουά. Το πρώτο διεξάγεται σε board διαστάσεων 5×5 με όριο βάθους επέκτασης $D = 4$. Το δεύτερο διεξάγεται σε board διαστάσεων 10×10, αλλά λόγω του υψηλού υπολογιστικού κόστους το όριο βάθους επέκτασης ορίζεται σε $D = 2$.

Με τα τουρνουά αυτά εξάγονται πολλά και ενδιαφέροντα συμπεράσματα για την απόδοση του κάθε παίκτη. Για το λόγο αυτό έχει μεγάλη σημασία η ανάλυσή και η παρουσίασή τους.

Στο πρώτο τουρνουά, όπου συμμετείχαν όλοι οι παίκτες, διεξήχθησαν 45 παρτίδες των 100 παιχνιδιών συνολικά. Κάθε παίκτης αντιμετώπισε εννέα αντιπάλους. Οπότε, η μέγιστη βαθμολογία για κάθε παίκτη είναι $9 \times 300 = 2700$ πόντοι. Τα αποτελέσματα του τουρνουά αυτού φαίνονται στο Σχήμα 6.10.

Σύμφωνα με τα παραπάνω αποτελέσματα οι πέντε καλύτεροι παίκτες είναι με σειρά καλύτερης επίδοσης οι ZvsMPe, MPvsS, ZvsMP, SvsS και ZvsS. Τελευταίος βγήκε ο MP που ήταν φυσιολογικό αφού ήταν παίκτης που δημιουργήθηκε με σκοπό να γίνει η μάθηση καλύτερη. Οι παίκτες MPvsMP και MPvsMPe είναι οι επόμενοι χειρότεροι παίκτες και αυτό ήταν αναμενόμενο αν θυμηθεί κανείς τις ασταθείς καμπύλες μάθησης που είχαν (Σχήματα 6.7 και 6.9, αντίστοιχα). Στη συνέχεια αποκλείστηκε ο ZvR ο οποίος έμαθε ενάντια στον πιο εύκολο αντίπαλο, τον R. Τέλος, αποκλείστηκε και ο MPvsSe.

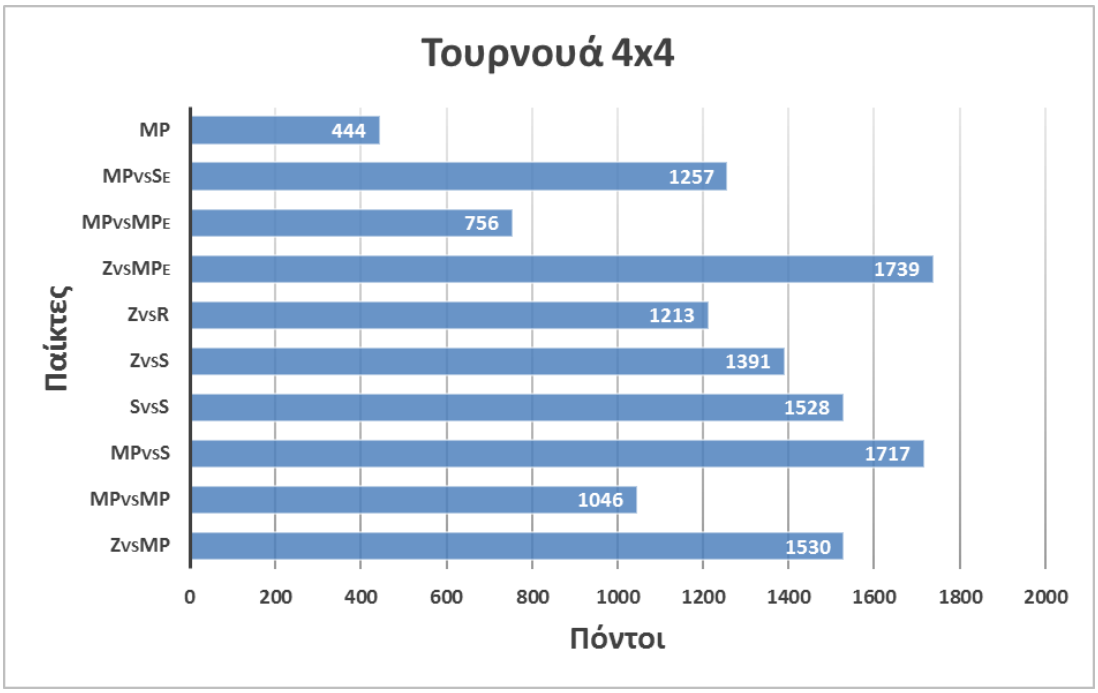
Οι πέντε καλύτεροι παίκτες αγωνίστηκαν στα επόμενα δύο τουρνουά. Σε αυτά τα τουρνουά διεξήχθησαν 10 παρτίδες των 100 παιχνιδιών. Κάθε παίκτης αντιμετώπισε τέσσερις αντιπάλους. Οπότε, η μέγιστη βαθμολογία για κάθε παίκτη είναι $4 \times 300 = 1200$ πόντοι. Τα αποτελέσματα του τουρνουά σε 5×5 φαίνονται στο Σχήμα 6.11 και του τουρνουά σε 10×10 στο Σχήμα 6.12.

Παίκτης	**Περιγραφή**
MP	Ο παίκτης που δημιουργήθηκε χειροκίνητα.
ZvsR	Παίκτης που προέκυψε από μάθηση χωρίς εξερεύνηση με αρχικοποίηση μηδενικά βάρη (Z) και αντίπαλο τον παίκτη που επιλέγει κίνηση τυχαία (R).
ZvsS	Παίκτης που προέκυψε από μάθηση χωρίς εξερεύνηση με αρχικοποίηση μηδενικά βάρη (Z) και αντίπαλο τον παίκτη που αξιολογεί με βάση το score (S).
SvsS	Παίκτης που προέκυψε από μάθηση χωρίς εξερεύνηση με αρχικοποίηση που αξιολογεί με βάση μόνο το score (S) και αντίπαλο τον παίκτη που αξιολογεί με βάση το score (S).
MPvsS	Παίκτης που προέκυψε από μάθηση χωρίς εξερεύνηση με αρχικοποίηση τα βάρη του MP και αντίπαλο τον παίκτη που αξιολογεί με βάση το score (S).
MPvsSe	Παίκτης που προέκυψε από μάθηση με εξερεύνηση με αρχικοποίηση τα βάρη του MP και αντίπαλο τον παίκτη που αξιολογεί με βάση το score (S).
ZvsMP	Παίκτης που προέκυψε από μάθηση χωρίς εξερεύνηση με αρχικοποίηση μηδενικά βάρη (Z) και αντίπαλο τον παίκτη MP.
MPvsMP	Παίκτης που προέκυψε από μάθηση χωρίς εξερεύνηση με αρχικοποίηση τα βάρη του MP και αντίπαλο τον παίκτη MP.
ZvsMPe	Παίκτης που προέκυψε από μάθηση με εξερεύνηση με αρχικοποίηση μηδενικά βάρη (Z) και αντίπαλο τον παίκτη MP.
MPvsMPe	Παίκτης που προέκυψε από μάθηση με εξερεύνηση με αρχικοποίηση τα βάρη του MP και αντίπαλο τον παίκτη MP.

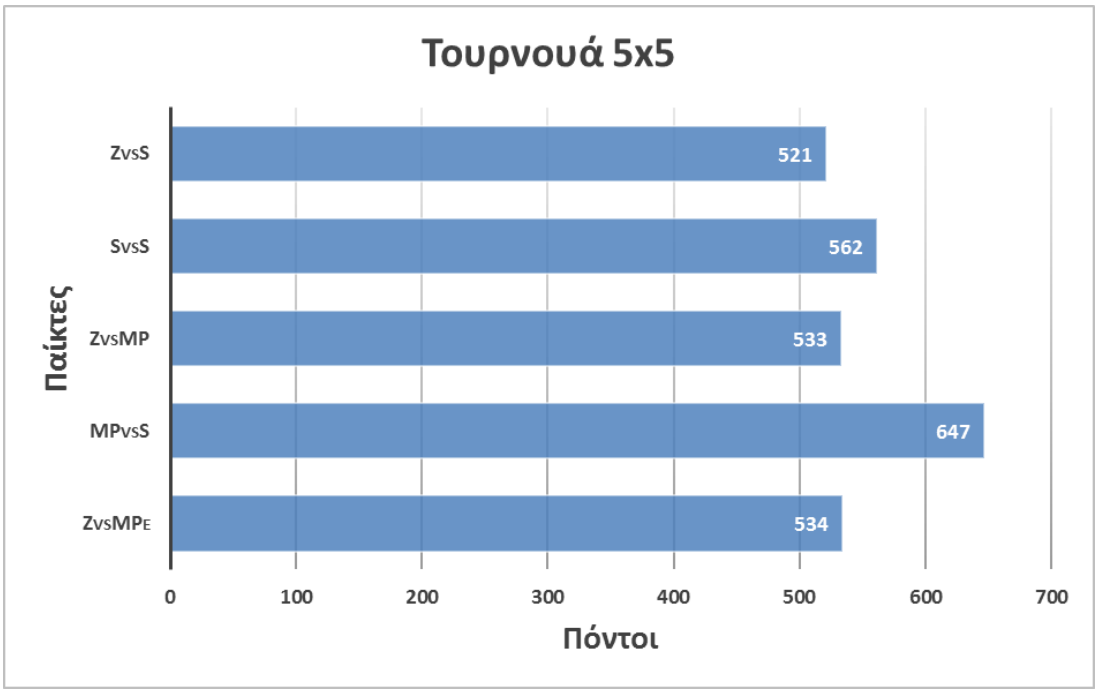
Πίνακας 6.1: *Παίκτες*

Παίκτης / Χαρακτηριστικό	MP	ZvsR	ZvsS	SvsS	MPvsS	MPvsSe	ZvsMP	MPvsMP	ZvsMPe	MPvsMPe
score	+1.000	+0.725	+0.627	+0.923	+0.790	+0.810	+0.633	+0.727	+0.608	+0.803
eastArrows	−0.250	+0.073	+0.135	−0.044	−0.022	−0.005	+0.121	+0.090	+0.085	−0.025
westArrows	+0.450	+0.352	+0.367	+0.228	+0.313	+0.300	+0.362	+0.246	+0.360	+0.260
northSafeArrows	+0.750	+0.266	+0.170	−0.060	+0.216	+0.265	+0.220	+0.172	+0.236	+0.347
southSafeArrows	−0.400	+0.070	+0.150	+0.076	−0.170	−0.172	+0.151	+0.113	+0.131	−0.180
northCornerArrows	+0.200	−0.002	+0.019	+0.010	−0.027	−0.024	+0.006	−0.016	+0.008	−0.020
southCornerArrows	−0.100	−0.005	+0.006	+0.001	+0.004	+0.002	+0.034	−0.006	+0.018	+0.026
northOpenArrows	−0.450	−0.226	−0.071	−0.136	−0.254	−0.258	−0.119	−0.310	−0.150	−0.275
southOpenArrows	+0.200	+0.220	+0.178	+0.253	+0.295	+0.300	+0.127	+0.261	+0.130	+0.276
northFragileArrows	−0.250	+0.140	+0.075	−0.163	+0.055	+0.022	+0.064	+0.035	+0.047	−0.010
southFragileArrows	+0.125	+0.320	+0.346	+0.280	+0.240	+0.216	+0.323	+0.232	+0.330	+0.186
eastArrPerSeq	−0.100	+0.007	+0.005	+0.003	+0.085	+0.079	−0.012	+0.008	−0.004	+0.015
eastArrInSeq	−0.100	+0.010	+0.007	+0.001	−0.046	−0.050	+0.011	+0.128	+0.016	+0.150
eastArrSeq	−0.450	+0.010	+0.003	−0.002	−0.421	−0.420	+0.018	−0.294	+0.056	−0.250
westArrPerSeq	+0.300	+0.004	+0.005	+0.010	+0.002	+0.010	−0.016	+0.035	+0.001	−0.085
westArrInSeq	+0.200	−0.014	−0.044	−0.054	−0.296	−0.300	−0.005	−0.290	+0.005	−0.006
westArrSeq	+0.750	+0.008	−0.037	−0.050	+0.307	+0.348	−0.001	+0.307	+0.004	+0.620
movesLeft	+0.200	+0.213	+0.261	+0.104	+0.145	+0.146	+0.217	+0.103	+0.200	+0.100

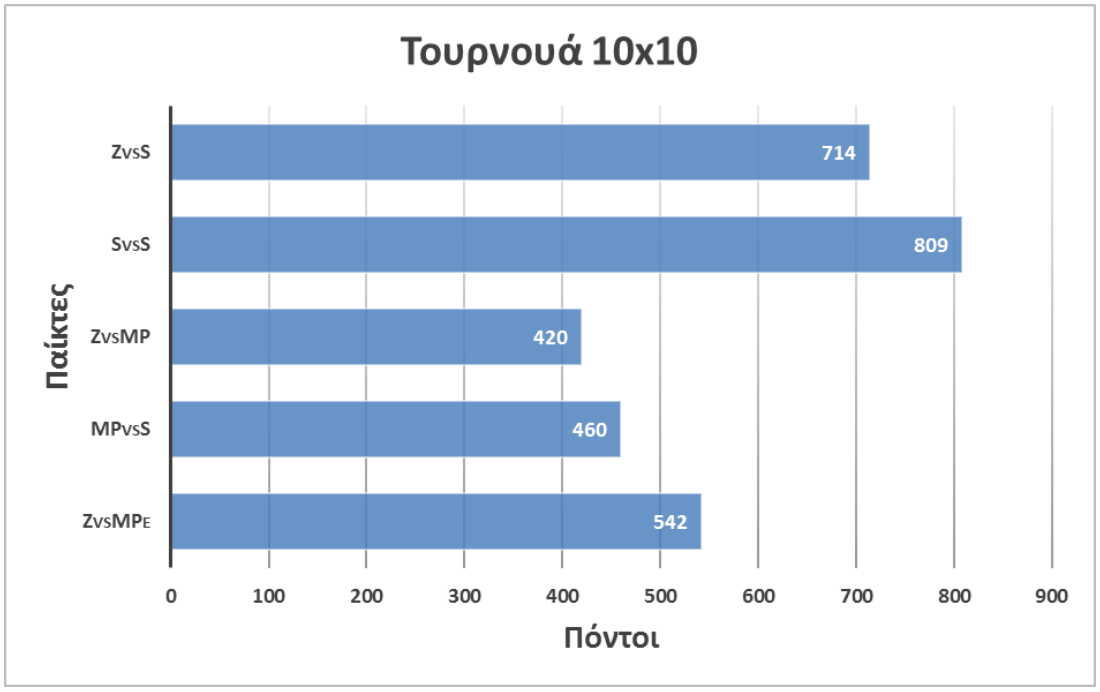
Πίνακας 6.2: *Τελικά Διανύσματα Βαρών των 10 Παικτών.*



Σχήμα 6.10: *Αποτελέσματα τουρνουά* 4×4.



Σχήμα 6.11: *Αποτελέσματα τουρνουά* 5×5.



Σχήμα 6.12: *Αποτελέσματα τουρνουά* 10×10.

Από τα σχήματα αυτά βγαίνει το παρακάτω συμπέρασμα. Όσο ανεβαίνουν οι διαστάσεις του board, οι παίκτες που αρχικοποιήθηκαν σε MP ή έμαθαν ενάντια σε MP ή και τα δύο, χάνουν συνεχώς έδαφος. Αυτό συμβαίνει διότι ο παίκτης MP δημιουργήθηκε βάσει του board 4×4. Οπότε, τα στοιχεία αυτά που τον έκαναν αξιόλογο και χρήσιμο για τη μάθηση, συνδέονταν κατά πολύ με τις συγκεκριμένες διαστάσεις. Αυτά τα στοιχεία πέρασε και στους παίκτες που τον χρησιμοποίησαν για τη μάθησή τους με οποιοδήποτε τρόπο.

Αντιθέτως, οι παίκτες στη μάθηση των οποίων συνέβαλλε με οποιοδήποτε τρόπο ο παίκτης S ανεβάζουν απόδοση όσο αυξάνονται οι διαστάσεις του board. Αυτό συμβαίνει γιατί ο παίκτης S αξιολογεί με βάση το score που αποτελεί το πιο σημαντικό χαρακτηριστικό σε οποιεσδήποτε διαστάσεις. Συνεπώς, στο πρώτο τουρνουά ξεχωρίζουν οι παίκτες ZvsMPe, MPvsS και ZvsMP. Στο δεύτερο έρχονται όλοι πολύ κοντά και ξεχωρίζει για λίγο ο MPvsS. Τέλος, στο τρίτο ξεχωρίζουν οι παίκτες ZvsS και SvsS. Σημειώνεται ότι η ανταγωνιστικότητα των παικτών διατηρείται σε board μεγαλύτερα του 4×4, κάτι το οποίο επιβεβαιώνεται από το γεγονός ότι καθένας από τους παίκτες έχει πλήρη αποτελεσματικότητα ενάντια στον R και στον S (βασικοί παίκτες ενάντια στους οποίους μπορεί να ελεγχθεί η ανταγωνιστικότητα ενός παίκτη) σε οποιοδήποτε board.

Με αυτά τα τουρνουά αποδείχτηκε ότι κάθε παίκτης έχει ένα ιδιαίτερο στοιχείο που τον κάνει αξιόλογο και αποδοτικό. Πρέπει να σημειωθεί ότι κανείς από τους πέντε δεν είναι εύκολος για έναν άνθρωπο. Ακόμα και αυτοί που αποκλείστηκαν είναι πολύ ικανοί και δύσκολοι αντίπαλοι.

Κεφάλαιο 7

Συμπεράσματα

7.1 Συζήτηση

Το επιτραπέζιο παιχνίδι Turning Points είναι ένα περίπλοκο παιχνίδι, διότι απαιτεί, πρώτον, σημαντική ικανότητα αντίληψης του αποτελέσματος που επιφέρει κάθε ενέργεια και, δεύτερον, ενισχυμένη δυνατότητα πρόβλεψης η οποία οδηγεί σε αξιόπιστες στρατηγικές.

Ο πράκτορας που υλοποιήθηκε στην εργασία αυτή μπορεί να προβλέψει εώς και πέντε κινήσεις. Όμως, για λόγους αποτελεσματικής μάθησης, που αναλύονται στην Ενότητα 4.2, το όριο βάθους επέκτασης που χρησιμοποιείται είναι $D = 4$. Το όριο αυτό επέτρεψε στον πράκτορα να επιλέγει κίνηση σε λιγότερο από ένα δευτερόλεπτο.

Για να δημιουργηθεί ο πράκτορας αυτός συνδυάστηκαν δύο αλγόριθμοι, ο Minimax με α–β Pruning και ο TD-Learning. Ο Minimax με α–β Pruning πραγματοποιεί την αναζήτηση της καλύτερης κίνησης για τον πράκτορα με βάση τη συνάρτηση αξιολόγηγης. Ο TD-Learning είναι υπεύθυνος για την εκμάθηση του πράκτορα ενάντια σε διάφορους αντιπάλους. Υπενθυμίζεται ότι το βασικό board έχει διαστάσεις 4×4.

Οι παραπάνω αλγόριθμοι οδήγησαν στην εξαγωγή αρκετών αξιόλογων παικτών. Οι περισσότεροι αποτελούν πολύ δύσκολους και έξυπνους αντιπάλους. Σημειώνεται ότι κάθε παίκτης είναι εξίσου αποτελεσματικός και σε board μεγαλύτερων ή μικρότερων διαστάσεων.

Παρατηρείται, λοιπόν, ότι η διαδικασία υλοποίησης αποτελεσματικού πράκτορα για το παιχνίδι Turning Points ήταν επιτυχής, αφού ο πράκτορας καταφέρνει να αντιμετωπίζει και να κερδίζει τον άνθρωπο με μεγάλη αποτελεσματικότητα.

7.2 Μελλοντικές Επεκτάσεις

Αρχικά, η εργασία αυτή μπορεί να βελτιωθεί ως προς τη χρήση μνήμης και την επιλογή αλγορίθμου αναζήτησης. Με τις κατάλληλες επιλογές και τροποποιήσεις μπορεί να επιτευχθεί μεγαλύτερο βάθος επέκτασης που θα βελτίωνε τον πράκτορα σε σημαντικό βαθμό.

Επίσης, θα μπορούσε η διαδικασία της μάθησης να εκτελεστεί με διάφορες παραλλαγές και αντιπάλους, όπως ακόμα να γίνει αναθεώρηση των χαρακτηριστικών της συνάρτησης αξιολόγησης. Έτσι, ο πράκτορας θα βελτιωνόταν ακόμα περισσότερο.

Τέλος, μελλοντικές επεκτάσεις της παρούσας εργασίας είναι, πρώτον, η προσθήκη δύο ακόμα παικτών (Ανατολικός και Δυτικός) και, δεύτερον, η υλοποίηση πράκτορα για την παραλλαγή του παιχνιδιού με εξαγωνικό board στο οποίο θα συμμετέχουν έξι παίκτες.

Βιβλιογραφία

[1] Stuart Russell και Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2η έκδοση, 2003.

[2] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας και Η. Σακελλαρίου. *Τεχνητή Νοημοσύνη*. Εκδόσεις Πανεπιστημίου Μακεδονίας, 3η έκδοση, 2011.

[3] J.von Neumann και O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[4] D.E. Knuth και R.W. Moore. *An Analysis of Alpha-Beta Pruning*. *Artificial Intelligence*, 6(4):293–326, 1975.

[5] Richard S. Sutton και Andrew G. Barto. *Reinforcement Learning: An Intro- duction*. The MIT Press, 1η έκδοση, 1998.

[6] Leslie Pack Kaelbling, Michael L. Littman και Andrew W. Moore. *Reinforcement Learning: A Survey*. *Journal of Artificial Intelligence Research 4*, σελίδες 237–285, 1996.

[7] Joseph Kisenwether. *Turning Points*, 2004. `http://web.archive.org/web/20151013181049/http://www.icehousegames.com/contest/icedes-1/turning-points/TurningPoints.htm`.

[8] Tom Mason. *Turning Points Flash Game*. `www.gabob.com/TurningPoints/game.html`.

[9] Claude E. Shannon. *Programming a Computer for Playing Chess*. *Philosophical Magazine*, 41(314), 1950.

[10] Arthur L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers*. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[11] Jacopo Festa και Stanislao Davino. *"IAgo Vs Othello": An artificial intelligence agent playing Reversi*.

[12] Michael Buro. *Logistello: A Strong Learning Othello Program*. *NEC Research Institute, Princeton, New Jersey*.

[13] Michael Buro. *Takeshi Murakami vs. Logistello*. *NEC Research Institute, Princeton, New Jersey*.

[14] Gerald Tesauro. *Temporal Difference Learning and TD-Gammon*. *Communications of the ACM*, 38(3), 1995.

[15] Walter J. Savitch. *Absolute Java*. Pearson, 5η έκδοση, 2012.

[16] Oracle. *Java Documentation*. `docs.oracle.com/en/java/`.