Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών υπολογιστών

Συστηματική Αναζήτηση και Ενισχυτική Μάθηση για το Επιτραπέζιο Παιχνίδι "Amazons"

Διπλωματική Εργασία

του

Γεωργίου Κωνσταντάκη

Εξεταστική Επιτροπή

Αναπληρωτής Καθηγητής Μιχαήλ Γ. Λαγουδάκης (Επιβλέπων)

Αναπληρωτής Καθηγητής Γεώργιος Χαλκιαδάκης

Αναπληρωτής Καθηγητής Αντώνιος Δεληγιαννάκης

Χανιά 2017

Technical
University
of Crete
School of Electrical and Computer Engineering

Systematic Search and Reinforcement Learning for the "Amazons" Board Game

Diploma Thesis

by

Georgios Konstantakis

Thesis Committee

Associate Professor Michail G. Lagoudakis (Supervisor)

Associate Professor Georgios Chalkiadakis

Associate Professor Antonios Deligiannakis

Chania 2017

Περίληψη

Τα παιχνίδια αποτελούσαν πάντα ένα πολύτιμο κομμάτι της έρευνας στον τομέα της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης λόγω των υψηλών δεξιοτήτων που απαιτούν. Η παρούσα διπλωματική εργασία επικεντρώνεται στο επιτραπέζιο παιχνίδι Amazons, το οποίο τα τελευταία χρόνια έχει αρχίσει να προσελκύει ερευνητές του τομέα της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης. Το Amazons είναι ένα παιχνίδι σκακιέρας που παίζεται από δύο παίκτες με εναλλασσόμενες κινήσεις. Κάθε παίκτης χειρίζεται 4 πούλια που κινούνται όπως οι βασίλισσες στο σκάκι, αλλά μετά από κάθε κίνηση πρέπει να τοποθετηθεί κάποιο μόνιμο εμπόδιο σε κάποια θέση στην σκακιέρα βάσει των κανόνων του παιχνιδιού. Νικητής είναι ο παίκτης που θα παγιδεύσει τον αντίπαλό του, ώστε να μην μπορεί να κινηθεί. Χαρακτηριστικό του παιχνιδιού είναι ο μεγάλος αριθμός επιλογών σε κάθε κίνηση. Στόχος της εργασίας είναι ο σχεδιασμός ενός αυτόνομου πράκτορα, ο οποίος θα είναι ικανός να παίξει ανταγωνιστικά το παιχνίδι, αλλά και ένα παραμετροποιήσιμο γραφικό περιβάλλον, μέσω του οποίου μπορούν οπτικοποιηθούν και να διεξαχθούν παιχνίδια μεταξύ διαφόρων παικτών (πρακτόρων ή ανθρώπων). Η στρατηγική του πράκτορά μας για την επιλογή κινήσεων βασίζεται στον αλγόριθμο αναζήτησης MiniMax με α-β Pruning σε συνδυασμό με μια προσθήκη εμπνευσμένη από τον αλγόριθμο Monte Carlo Tree Search. Σημαντικό ρόλο στην αξιολόγηση κινήσεων έχει η συνάρτηση αξιολόγησης που σχεδιάστηκε, τα βάρη της οποίας προσαρμόζονται μέσω του αλγορίθμου ενισχυτικής μάθησης TD-Learning κατά την εκτέλεση πολλών επαναλήψεων του παιχνιδιού. Ο συνδυασμός των παραπάνω οδήγησε στην δημιουργία διάφορων αποδοτικών παικτών, οι οποίοι αξιολογήθηκαν στα πλαίσια ενός τουρνουά.

Abstract

Games have always been a valuable subject of research in the fields of Artificial Intelligence and Machine Learning, because of the high level of sophistication they require. This thesis focuses on a board game called “Amazons”, which during the recent years has started attracting researchers from the field of Artificial Intelligence and Machine Learning. Amazons is a chess-board game played by two players taking alternating turns. Each player handles 4 checkers, whose movements are similar to the queen in chess, but after each move a permanent obstacle must be also placed on a chessboard position, according to the rules of the game. The player who will trap his opponent and will make him unable to move is the winner. Central feature of the game is the large number of choices that each player has at each turn. The goal of this thesis is to create an autonomous agent, which will be able to play this game competitively, but also to create a graphical environment, through which many games among different players (agents or people) can take place. Our agent’s strategy for choosing moves is based on the MiniMax search algorithm with alpha-beta Pruning, combined with an addition inspired by the Monte Carlo Tree Search Algorithm. An important role for the movements’ evaluation is the proposed evaluation function designed for the game, the weights of which are adapted through the reinforcement learning algorithm TD-Learning by repeatedly playing many games. The combination of the techniques mentioned above led to the creation of several efficient players who were evaluated through a tournament.

Περιεχόμενα

Σχήματα

Κεφάλαιο 1. Εισαγωγή

1.1 Περιγραφή και Συνεισφορά Εργασίας

Τα παιχνίδια αποτελούσαν πάντα ένα πολύτιμο κομμάτι της έρευνας στον τομέα της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης λόγω των υψηλών δεξιοτήτων που απαιτούν. Το Amazons είναι ένα παιχνίδι σκακιέρας το οποίο παίζεται από δύο παίκτες. Το παιχνίδι παίζεται σε σκακιέρα 10×10 (υπάρχουν κι άλλες παραλλαγές διαστάσεων, αλλά αυτή είναι η βασική έκδοση του παιχνιδιού) και ο κάθε παίκτης διαθέτει τέσσερα πούλια τα οποία ονομάζονται Amazons και τοποθετούνται σε συγκεκριμένες θέσεις της σκακιέρας. Οι παίκτες παίζουν εναλλάξ και κάθε πούλι μπορεί να κινηθεί πάνω στις κενές θέσεις της σκακιέρας, ακριβώς όπως και η βασίλισσα στο σκάκι. Κάθε παίκτης όταν κάνει μια κίνηση μετά πρέπει να «πυροβολήσει» προς κάποια κατεύθυνση κατά μήκος της οποίας σε κάποια επιλεγμένη θέση θα δημιουργηθεί ένα μόνιμο εμπόδιο το οποίο θα παραμείνει μέχρι το τέλος του παιχνιδιού. Χαρακτηριστικό του παιχνιδιού είναι ο μεγάλος αριθμός επιλογών σε κάθε κίνηση. Νικητής είναι ο παίκτης που θα παγιδεύσει τον αντίπαλό του ώστε να μην μπορεί να κινηθεί. Το παιχνίδι απαιτεί από τον παίκτη σημαντική αντίληψη του αποτελέσματος που επιφέρει κάθε κίνηση ώστε να παγιδεύσει τον αντίπαλό του χωρίς να βρεθεί σε δυσμενή θέση ο ίδιος.

Σκοπός της εργασίας είναι η δημιουργία και η εκπαίδευση ενός πράκτορα που θα παίζει το παιχνίδι Amazons ώστε να είναι ανταγωνιστικός απέναντι στον άνθρωπο αλλά και απέναντι σε άλλους πράκτορες. Το παιχνίδι δεν είναι δύσκολο ως προς τους κανόνες του, αλλά ως προς την κατάλληλη επιλογή κίνησης λόγω του μεγάλου αριθμού διαθέσιμων κινήσεων. Χρησιμοποιούμε δέντρο αναζήτησης για την ανάπτυξη του παιχνιδιού και η αναζήτηση της κατάλληλης κίνησης γίνεται μέσω του αλγορίθμου Monte Carlo Tree Search σε συνδυασμό με το α-β Pruning. Τα κριτήρια επιλογής για την εύρεση της κατάλληλης κίνησης παρέχονται μέσω μιας συνάρτησης αξιολόγησης η οποία κάνει μία εκτίμηση της εξέλιξης του παιχνιδιού από οποιαδήποτε πιθανή κατάσταση. Την διαδικασία εκμάθησης του πράκτορα την αναλαμβάνει ο αλγόριθμος TD Learning. Επίσης για να οπτικοποιηθεί το παιχνίδι και για να υπάρχει δυνατότητα αξιολόγησης δημιουργήθηκε ένα γραφικό περιβάλλον απεικόνισης το παιχνιδιού, στο οποίο ο χρήστης έχει την δυνατότητα να παίξει είτε απέναντι σε άλλους χρήστες είτε απέναντι στον υπολογιστή.

1.2 Επισκόπηση εργασίας

Στο Κεφάλαιο 2 διατυπώνεται εν συντομία το θεωρητικό υπόβαθρο το οποίο χρειάστηκε για την εργασία. Αναλύονται συνοπτικά έννοιες όπως δέντρο παιχνιδιού, μέθοδοι αναζήτησης και κάποιοι χρήσιμοι αλγόριθμοι που χρησιμοποιήθηκαν, δηλαδή MiniMax, α-β Pruning, Monte Carlo Tree Search. Αναλύεται ο ρόλος και η χρησιμότητα της συνάρτησης αξιολόγησης, όπως και οι τεχνικές μάθησης και ο αλγόριθμος TD Learning που χρησιμοποιήθηκε.

Στο Κεφάλαιο 3 γίνεται αναλυτική παρουσίαση του παιχνιδιού Amazons και των κανόνων του. Καθορίζονται επίσης οι στόχοι της παρούσας διπλωματικής εργασίας ως προς το παιχνίδι και περιγράφονται διάφορες σχετικές εργασίες από την υπάρχουσα βιβλιογραφία.

Στο Κεφάλαιο 4 περιγράφεται η προσέγγισή μας για την υλοποίηση του πράκτορα για το παιχνίδι Amazons. Αναλυτικότερα, παρουσιάζεται η διαδικασία της δημιουργίας του δέντρου παιχνιδιού, όπως επίσης η εφαρμογή του αλγορίθμου Monte Carlo Tree Search σε συνδυασμό με τον αλγόριθμο α-β Pruning. Επίσης περιγράφεται η προτεινόμενη συνάρτηση αξιολόγησης και αναλύεται η δομή και τα χαρακτηριστικά της. Τέλος, παρουσιάζεται η εφαρμογή του αλγορίθμου TD Learning στο συγκεκριμένο πρόβλημα που ήταν υπεύθυνος για την διαδικασία της μάθησης.

Στο Κεφάλαιο 5 καλύπτονται θέματα που σχετίζονται με την υλοποίηση της εργασίας. Αναφέρονται η γλώσσα υλοποίησης, οι κλάσεις που δημιουργήθηκαν, όπως και κάποιες βασικές συναρτήσεις που υλοποιήθηκαν. Επίσης δίνονται πληροφορίες για την δημιουργία του γραφικού περιβάλλοντος.

Στο Κεφάλαιο 6 παρουσιάζεται η πειραματική διαδικασία για την αξιολόγηση της αποτελεσματικότητας της σχεδίασής μας. Παρουσιάζονται τα αποτελέσματα της διαδικασίας μάθησης, οι διαφορετικοί πράκτορες που δημιουργήθηκαν και η συγκριτική αξιολόγησή τους μέσω ενός τουρνουά.

Στο Κεφάλαιο 7 γίνεται μια αποτίμηση της εργασίας, των αποτελεσμάτων που εξήχθησαν και τέλος παρουσιάζονται μελλοντικές βελτιώσεις και επεκτάσεις της εργασίας.

Κεφάλαιο 2. Γνωστικό Υπόβαθρο

2.1 Αναζήτηση

Ένας πράκτορας που έχει στην διάθεσή του πολλές άμεσες επιλογές άγνωστης αξίας μπορεί να αποφασίζει τι να κάνει εξετάζοντας πρώτα διάφορες δυνατές ακολουθίες ενεργειών που οδηγούν σε καταστάσεις γνωστής αξίας και μετά επιλέγοντας την ενέργεια που οδηγεί στην καλύτερη ακολουθία. Η διαδικασία εύρεσης μιας τέτοιας ενέργειας ονομάζεται αναζήτηση [1] [2].

2.1.1 Δέντρο Αναζήτησης

Η πιο διαδεδομένη μέθοδος επίλυσης προβλημάτων αναζήτησης είναι η συστηματική ανάπτυξη ενός δέντρου αναζήτησης [1] [2], το οποίο ομαδοποιεί με συστηματικό τρόπο όλες τις πιθανές ακολουθίες ενεργειών. Το δέντρο αναζήτησης δημιουργείται από την αρχική κατάσταση και τη συνάρτηση διαδόχων καταστάσεων. 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 κόμβων, ένας για κάθε διαθέσιμη ενέργεια, που αποτελούν την επέκταση του αρχικού κόμβου και τους θεωρούμε παιδιά του, καθώς βρίσκονται ένα επίπεδο βαθύτερα στο δέντρο. Αμέσως μετά, εξετάζεται ένας εκ των νέων κόμβων βάσει κάποιας στρατηγικής αναζήτησης 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 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.

Στα παιχνίδια εναλλασσόμενης κίνησης η παραπάνω διαδικασία αναζήτησης εκτελείται ως εξής. Η κατάσταση στην οποία βρίσκεται την συγκεκριμένη στιγμή το παιχνίδι θεωρείται ως ο κόμβος-ρίζα του δέντρου αναζήτησης και παιδιά του όλες οι

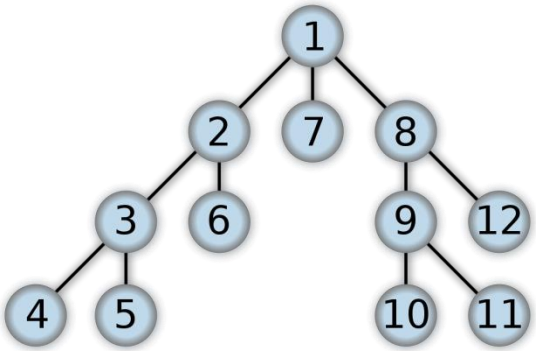
καταστάσεις που προκύπτουν από τις νόμιμες κινήσεις του παίκτη που έχει σειρά να παίξει. Στο επόμενο επίπεδο του δέντρου τα παιδιά θα δημιουργηθούν με βάση τις νόμιμες κινήσεις του αντιπάλου και ούτω καθεξής εναλλάξ ανά επίπεδο. Το δέντρο επεκτείνεται μέχρι τις καταστάσεις εκείνες στις οποίες το παιχνίδι τερματίζεται, σύμφωνα πάντα με τους κανόνες του εκάστοτε παιχνιδιού. Οι καταστάσεις αυτές ονομάζονται τερματικές και αποτελούν τις μοναδικές καταστάσεις, στις οποίες γνωρίζουμε το ακριβές τελικό αποτέλεσμα του παιχνιδιού, δηλαδή ποιος παίκτης είναι ο νικητής και ποιος ο ηττημένος.

2.1.2 Αναζήτηση πρώτα σε βάθος

Η αναζήτηση πρώτα σε βάθος (Depth-First Search, DFS) [1] [2] επιλέγει πάντα να κάνει επέκταση του πιο πρόσφατου κόμβου-παιδί του δέντρου αναζήτησης που δημιουργήθηκε και συνεχίζει σε βάθος του δέντρου μέχρι να βρεθεί ο ζητούμενος κόμβος ή μέχρι να φτάσει σε κόμβο που δεν έχει παιδιά. Στη συνέχεια, η αναζήτηση υπαναχωρεί στον πιο πρόσφατο ανεξερεύνητο κόμβο. Σε μια μη-αναδρομική υλοποίηση, όλοι οι νέοι ανεξερεύνητοι κόμβοι τοποθετούνται σε μια δομή στοίβας και η αναζήτηση πρώτα σε βάθος επιλέγει για επέκταση πάντα τον κόμβο που βρίσκεται στην κορυφή της.

Πιο συγκεκριμένα, επιλέγεται ο αρχικός κόμβος-ρίζα, τον επεκτείνουμε και δημιουργούμε τα παιδιά του. Στην συνέχεια παίρνουμε το πρώτο παιδί και επεκτείνοντάς το, δημιουργούνται τα δικά του παιδιά. Η παραπάνω διαδικασία θα επαναληφθεί σε όλο και μεγαλύτερο βάθος μέχρι εν τέλει να φθάσουμε σε κόμβο στόχο ή σε κάποιον τερματικό κόμβο του δέντρου (φύλλο). Αν δεν έχει βρεθεί ο κόμβος στόχος, τότε επιστρέφουμε ένα επίπεδο πίσω και επεκτείνουμε το δεύτερο παιδί και η διαδικασία συνεχίζεται.

Στο Σχήμα 1 φαίνεται ένα παράδειγμα αναζήτησης πρώτα σε βάθος. Κάθε κόμβος είναι αριθμημένος με βάση την σειρά που επεκτείνεται. Δηλαδή ο κόμβος με τον αριθμό 1 αναπτύσσεται πρώτος και δημιουργούνται οι τρεις κόμβοι παιδιά του. Στην συνέχεια το πρώτο παιδί του αναπτύσσεται, γι αυτό και είναι αριθμημένος με το 2 και δημιουργούνται τα δικά του παιδιά. Σειρά παίρνει ο κόμβος που είναι αριθμημένος με το 3 για να αναπτυχθεί και παράγει τα δύο παιδιά του. Όταν θα αναπτυχθεί με την σειρά του ο κόμβος 4 διαπιστώνεται ότι αποτελεί φύλλο του δέντρου, οπότε επιστρέφουμε στο κόμβο 3 και θα αναπτυχθεί το επόμενο παιδί του κόμβου 3, δηλαδή ο κόμβος 5. Επειδή κι αυτός αποτελεί φύλλο του δέντρου, επιστρέφουμε ακόμα παραπάνω στον κόμβο 2 και συνεχίζουμε με το επόμενο παιδί του. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο μέχρι να βρεθεί κατάσταση στόχου ή να μην υπάρχει ανεξερεύνητος κόμβος.



Σχήμα 1. Αναζήτηση Πρώτα σε Βάθος

2.1.3 Αναζήτηση με Υπαναχώρηση

Μια παραλλαγή της αναζήτησης πρώτα σε βάθος είναι η αναζήτηση με υπαναχώρηση (Backtracking Search) [1] [2]. Η διαφορά τους είναι ότι κατά την αναζήτηση με υπαναχώρηση δεν δημιουργούνται όλα τα παιδιά ενός κόμβου, όταν αυτός επεκτείνεται, αλλά μόνο το πρώτο με το οποίο συνεχίζεται η αναζήτηση (σε βάθος). 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 παιδιά ενός κόμβου δε δημιουργούνται μαζικά, αλλά διαδοχικά, όταν έρθει η σειρά του καθενός.

Με την συγκεκριμένη στατηγική 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, έτσι ώστε όταν έρθει η ώρα η αναζήτηση να υπαναχωρήσει από το παιδί στον πατέρα να επανέλθει η σωστή κατάσταση.

Οπότε, 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 εξοικονόμηση μνήμης (και χρόνου) καθώς αποφεύγονται όλες οι αντιγραφές κόμβων σε κάθε επέκταση και κερδίζουμε τον χώρο που θα καταλάμβαναν διαφορετικά.

2.1.4 Αναζήτηση Περιορισμένου Βάθους

Η αναζήτηση πρώτα σε βάθος, αλλά και η αναζήτηση με υπαναχώρηση, αντιμετωπίζουν πολλές δυσκολίες. Όταν το βάθος είναι πολύ μεγάλο, αναλώνουμε πάρα

πολύ χρόνο αναζήτησης σε ένα συγκεκριμένο υποδέντρο που μπορεί να μην περιέχει καν την λύση που αναζητούμε, ενώ η λύση μπορεί να βρίσκεται σε κάποιο άλλο πολύ πιο ρηχό υποδέντρο. Επίσης μπορεί 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 θέτουμε ένα όριο έτσι το δέντρο αναζήτησης επεκτείνεται μέχρι κάποιο προκαθορισμένο βάθος (όριο επέκτασης) και η αναζήτηση υπαναχωρεί άμεσα μόλις φτάσει σε αυτό το βάθος, με σκοπό την αποφυγή όσων αναφέρθηκαν παραπάνω. Για παράδειγμα στο Σχήμα 1 αν εφαρμόσουμε όριο επέκτασης το βάθος 2 τότε οι 4 κόμβοι που βρίσκονται στο χαμηλότερο επίπεδο δεν θα δημιουργηθούν.

Ωστόσο, η στρατηγική περιορισμένου βάθους 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 παρακάτω.

2.2 Αλγόριθμοι Αναζήτησης

2.2.1 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. 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.

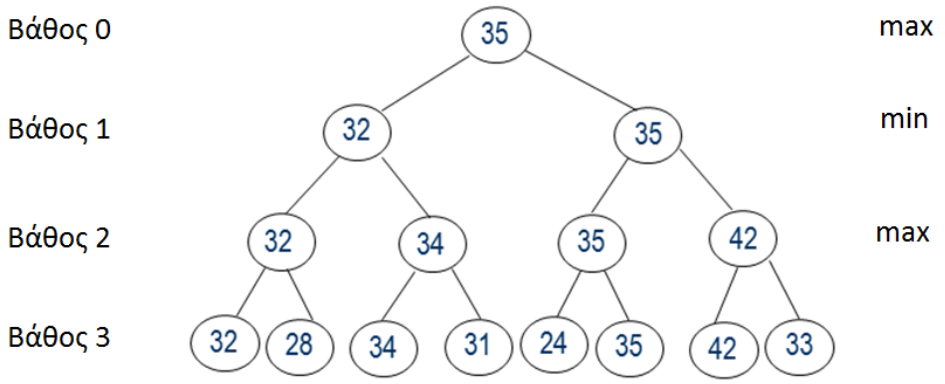
Λύση στο παραπάνω πρόβλημα δίνει ο αλγόριθμος MiniMax [1] [2] ο οποίος λειτουργεί ως εξής: Έχουμε δύο παίκτες. Ο ένας παίζει ως max και ο άλλος ως min. Η στρατηγική με την οποία παίζει ο παίκτης Max , όπως υποδηλώνει και το όνομα του, είναι

να μεγιστοποιήσει τις απολαβές του, ενώ σε αντίθεση ο παίκτης Min προσπαθεί να ελαχιστοποιήσει το κέρδος του max που αυτό συνήθως οδηγεί στο να μεγιστοποιήσει τις δικές του απολαβές, δεδομένου ότι οι δύο παίκτες έχουν αντικρουόμενα συμφέροντα. Τέτοιου είδους παιχνίδια είναι γνωστά ως παιχνίδια μηδενικού αθροίσματος (zero-sum).

Ο αλγόριθμος MiniMax βασίζεται στην αναζήτηση με υπαναχώρηση. Το δέντρο του παιχνιδιού αναπτύσσεται όπως αναφέρθηκε παραπάνω. Η διαφορά είναι ότι κάθε επίπεδο του δέντρου αντιστοιχεί σε κίνηση του ενός ή του άλλου παίκτη, με τον Max να έχει τον πρώτο λόγο στη ρίζα του δέντρου και στην συνέχεια εναλλάσσονται με τον Min ανάλογα με το ποιος παίκτης παίζει. 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 του παιχνιδιού. Η απόδοση τιμής στους ενδιάμεσους κόμβους γίνεται από κάτω προς τα πάνω δηλαδή 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 διακρίνουμε δύο περιπτώσεις, μία για κάθε παίκτη. Στην περίπτωση που παίζει ο Max παίκτης επιλέγει από τους κόμβους-παιδιά του αυτόν με την μεγαλύτερη τιμή, ενώ αντίθετα ο Min παίκτης επιλέγει το κόμβο-παιδί του με την 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 (και κατ' επέκταση χειρότερη για τον παίκτη Max). 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 της ρίζας με τη μέγιστη τιμή (αυτή που προωθήθηκε στη ρίζα).

Για να καταλάβουμε καλύτερα πως λειτουργεί ο αλγόριθμος μπορούμε να δούμε την λειτουργία του στο Σχήμα 2.



Σχήμα 2. Παράδειγμα MiniMax

Στο παραπάνω δέντρο όπως βλέπουμε έχουμε ένα ιδεατό παιχνίδι που τελειώνει μετά από 3 κινήσεις. Ξεκινάει και παίζει ο Max παίκτης, στην συνέχεια κάνει κίνηση ο Min παίκτης και τελειώνει με κίνηση του παίκτη Max. Ο αλγόριθμος MiniMax, όπως προαναφέρθηκε, αποδίδει τις κατάλληλες τιμές στους κόμβους από κάτω προς τα πάνω. Στο τελευταίο επίπεδο, δηλαδή σε βάθος 3 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, παίζει ο παίκτης Max και επιλέγει την μεγαλύτερη τιμή από τις τιμές των παιδιών του. Όπως βλέπουμε από τις τιμές 32 και 28 επιλέγει την τιμή 32. Με τον ίδιο τρόπο επιλέγονται και οι τιμές στους κόμβους που βρίσκονται στο ίδιο επίπεδο, αλλά σε διαφορετική διαδρομή του δέντρου. Στο αμέσως υψηλότερο επίπεδο, παίζει ο παίκτης Min που καλείται κι αυτός να επιλέξει με την σειρά του την ελάχιστη τιμή από τις τιμές των κόμβων-παιδιών του. Όπως βλέπουμε στο παράδειγμα μεταξύ των τιμών 32 και 34 επιλέγει το 32 και από την άλλη πλευρά του δέντρου επιλέγει αντίστοιχα μεταξύ του 35 και 42 το 35. Τέλος, φτάνοντας στη ρίζα του δέντρου, όπου παίζει ξανά ο Max, επιλέγεται ως τελική (βέλτιστη) κίνηση εκείνη που οδηγεί στο παιδί με την μεγαλύτερη τιμή η οποία είναι το 35.

Ακολουθώντας αυτή τη διαδικασία σε κάθε κόμβο και δημιουργώντας το δέντρο με πρώτα σε βάθος αναζήτηση προσδίδουμε τελικά στην αρχική μας κατάσταση μία τιμή. Η τιμή αυτή ονομάζεται τιμή MiniMax και για να βρούμε το μονοπάτι που μας την προσφέρει, απλά αναζητούμε ποιο ή ποια από τα παιδιά-καταστάσεις έχουν αυτήν την τιμή.

Στο Σχήμα 3 βλέπουμε τον ψευδοκώδικα για τον αλγόριθμο MiniMax

```
01 function minimax(node, depth, maximizingPlayer)
02     if depth=0 or node is a terminal node
03             return the heuristic value of node
04     if maximizingPlayer
05             best value :=-∞
06             for each child of node
07                     v := minimax(child, depth-1, FALSE)
08                     bestValue := max(bestValue, v)
09             return bestValue
10     else        (* minimizing player *)
11             bestValue := +∞
12             for each child of node
13                     v:= minimax(child, depth-1, TRUE)
14                     bestValue := min(bestValue, v)
15             return bestValue
```

Σχήμα 3 Ψευδοκώδικας για τον αλγόριθμο Minimax

2.2.2 Alpha-Beta Pruning

Η πολυπλοκότητα του αλγορίθμου MiniMax αυξάνεται εκθετικά ως προς το βάθος του δέντρου. Όσες περισσότερες διαθέσιμες κινήσεις υπάρχουν για κάθε παίκτη σε κάποιο παιχνίδι τόσο μεγαλύτερος είναι ο ρυθμός της εκθετικής αύξησης, άρα κατά συνέπεια και το δέντρο αναζήτησης γίνεται αρκετά πιο μεγάλο όσο πάμε σε μεγαλύτερο βάθος. Έτσι η εύρεση της καλύτερης δυνατής κίνησης με τις καλύτερες απολαβές γίνεται αρκετά χρονοβόρα αφού πρέπει να αναπτυχθεί και να διαπεραστεί ολόκληρο το δέντρο.

Για να επιλυθούν τα παραπάνω προβλήματα πρέπει να μειωθούν οι κόμβοι τους οποίους έχουμε να εξερευνήσουμε, δηλαδή να «κλαδευτεί» το δέντρο. Όπως μπορούμε να παρατηρήσουμε στον αλγόριθμο MiniMax, κάποιοι κόμβοι ή ακόμα και ολόκληρα υποδέντρα δεν επηρεάζουν την τιμή MiniMax και την τελική απόφαση στην ρίζα. Οπότε η βασική ιδέα είναι ότι μπορούν να αποκοπούν τέτοιοι κόμβοι ή τέτοια υποδέντρα ώστε να εξοικονομήσουμε πολύτιμο χρόνο. Αυτή ακριβώς την ιδέα έρχεται να υλοποιήσει ο αλγόριθμος α-β pruning [1] [2] [3]. Αποκόπτοντας υποδέντρα και κόμβους οι οποίοι δεν επηρεάζουν το τελικό αποτέλεσμα, φέρνει τα ίδια αποτελέσματα με τον αλγόριθμό MiniMax σε πολύ μικρότερο χρόνο. Ο αλγόριθμος α-β pruning αποτελεί δηλαδή μια βελτίωση του αλγόριθμου MiniMax.

```
01 function alphabeta(node, depth, α, β, maximizingPlayer)
02     if depth=0 or node is a terminal node
03           return the heuristic value of node
04     if maximizingPlayer
05           v := -∞
06           for each child of node
07                 V := max(v, alphabeta(child, depth-1, α, β, FALSE))
08                 α := max(α, v)
09                 if β ≤ α
10                       break (* β cut-off *)
11           return v
12     else
13           v := +∞
14           for each child of node
15                 v := min(v, alphabeta(child, depth-1, α, β, TRUE))
16                 β := min(β, v)
17                 if β ≤ α
18                       break (* α cut-off *)
19           return v
```

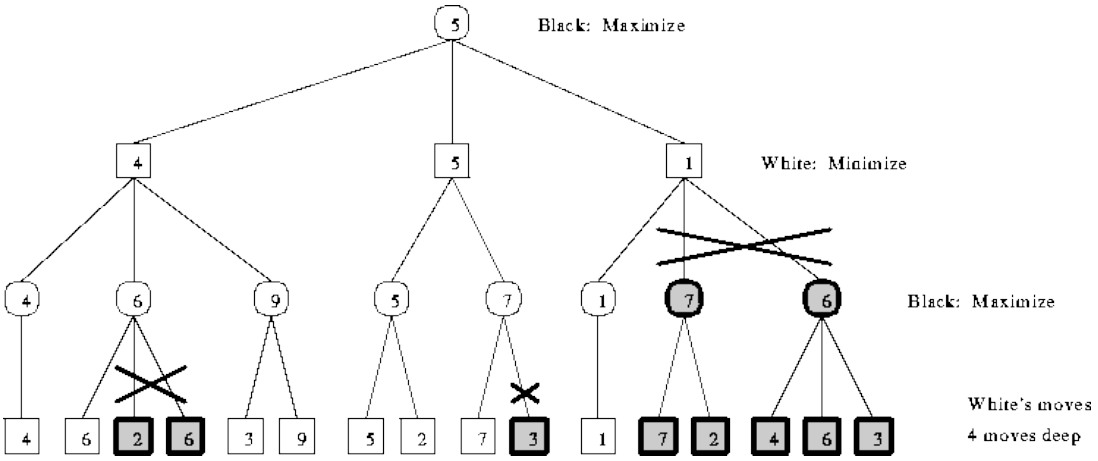
Σχήμα 4 Ψευδοκώδικας για τον αλγόριθμο α-β Pruning

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 μεγαλύτερες τιμές), ενώ σε κάθε κόμβο που παίζει ο Min ανανεώνεται η τιμή του β (μόνο προς μικρότερες τιμές). 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. Ο αλγόριθμος αναζήτησης MiniMax με α-β Pruning φαίνεται στο Σχήμα 4.

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 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 5 φαίνεται το αποτέλεσμα μετά την εφαρμογή του αλγορίθμου MiniMax με α-β Pruning. Οι επτά κόμβοι που αποκόπτονται είναι περιττοί για την εύρεση της τιμής MiniMax στην ρίζα και της βέλτιστης κίνησης. Επισημαίνεται ότι η τελική τιμή στη ρίζα ταυτίζεται με αυτήν του αλγορίθμου MiniMax παρότι δεν εξερευνάται ένα μεγάλο μέρος του δέντρου.



Σχήμα 5. Παράδειγμα α-β Pruning

2.2.3 Monte Carlo Tree Search

Παρόλο που οι παραπάνω αλγόριθμοι έχουν αρκετά καλά αποτελέσματα, σε παιχνίδια που έχουν αρκετά μεγάλο εύρος πιθανών κινήσεων όπως το σκάκι, το Go, το Amazons, κ.α. λόγω πυκνότητας του δέντρου είναι αρκετά χρονοβόρο να αναζητήσουν σε μεγάλο βάθος για την βέλτιστη κίνηση και να επιστρέψουν ένα καλό αποτέλεσμα. Για το λόγο αυτό σε τέτοια παιχνίδια με μεγάλο παράγοντα διακλάδωσης (branching factor) δεν είναι αποδοτική λύση το να εξερευνήσουμε όλο το δέντρο γιατί είναι αρκετά χρονοβόρο σε σημείο απαγορευτικό σε κάποιες περιπτώσεις ακόμα και αν χρησιμοποιήσουμε κάποιο αλγόριθμο αποκοπής, όπως το α-β pruning.

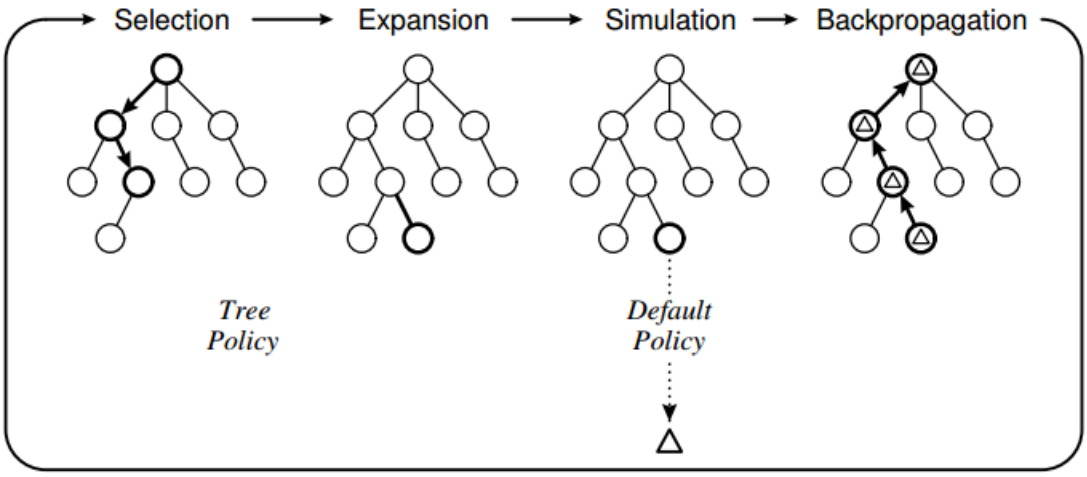
Όταν λοιπόν έχουμε παιχνίδια με αρκετά μεγάλο αριθμό κινήσεων όπως το Go, το Amazons, κλπ στα οποία αλγόριθμοι, όπως ο MiniMax και ο α-β pruning δεν μπορούν να χρησιμοποιηθούν λόγω περιορισμένου χρόνου, πρέπει να χρησιμοποιήσουμε μια άλλη

στρατηγική, η οποία δεν θα απαιτεί να εξετάσει κάθε κόμβο του δέντρου για να βρει την βέλτιστη λύση. Μία τέτοια στρατηγική υλοποιεί και ο αλγόριθμος Monte Carlo Tree Search (MCTS) [4] [5]. Η βασική ιδέα του Monte Carlo Tree Search είναι αρκετά απλή. Ξεκινάει ο αλγόριθμος και διατρέχει το τρέχον δέντρο (αρχικά μόνο η ρίζα) έως ότου φτάσει στον καλύτερο κόμβο που δεν έχουν επεκταθεί ακόμη όλα τα παιδιά του. Έπειτα επιλέγει μία ανεξερεύνητη κίνηση με βάση κάποια πολιτική (μπορεί να είναι και τυχαία η επιλογή) και επεκτείνει τον κόμβο ώστε να δημιουργήσει τον αντίστοιχο κόμβο-παιδί, ο οποίος προστίθεται στο δέντρο. Από αυτόν τον κόμβο ξεκινάει μια πλήρης προσομοίωση της συγκεκριμένης παρτίδας του παιχνιδιού ακολουθώντας μια προκαθορισμένη πολιτική (μπορεί να είναι και εντελώς τυχαία) μέχρι να φτάσουμε σε τελικό αποτέλεσμα, δηλαδή νίκη ή ήττα. Έπειτα ενημερώνεται ο κόμβος αυτός σχετικά με το αποτέλεσμα, το οποίο και διαδίδεται προς την ρίζα του δέντρου σύμφωνα με την λογική του MiniMax. Αυτό γίνεται επαναληπτικά έως ότου εξαντληθεί ο διαθέσιμος χρόνος. Έτσι, όσο περισσότερο χρόνο διαθέτει ο αλγόριθμος τόσο μεγαλύτερο μέρος του δέντρου θα εξετάσει και οι κινήσεις-κόμβοι με τις περισσότερες νίκες (από τις προσομοιώσεις) είναι πιθανότερο να επιλεχθούν.

Πιο αναλυτικά τα τέσσερα βήματα που κάνει ο αλγόριθμος MCTS είναι:

1. Selection: Ξεκινώντας από τον κόμβο-ρίζα, διατρέχουμε συστηματικά το δέντρο προς το καλύτερο κόμβο-παιδί μέχρι να φτάσουμε σε έναν επεκτάσιμο κόμβο. ‘Ενας κόμβος είναι επεκτάσιμος εάν αντιπροσωπεύει μια μη τερματική κατάσταση και δεν έχουν επεκταθεί ακόμη όλα τα παιδιά του.
2. Expansion: Ο επιλεγμένος κόμβος αναπτύσσεται για να παραχθεί ένα ακόμη παιδί του, το οποίο προστίθεται στο δέντρο.
3. Simulation: Από το νέο κόμβο-παιδί ξεκινάει μια προσομοίωση του παιχνιδιού μέχρι την λήξη του με βάση μια προκαθορισμένη πολιτική επιλογής κινήσεων ώστε να παραχθεί μια αναμενόμενη τιμή για τον κόμβο με βάση το αποτέλεσμα από την προσομοίωση (δηλαδή νίκη ή ήττα).
4. Backpropagation: Το αποτέλεσμα της προσομοίωσης επιστρέφει και ενημερώνονται τα στατιστικά για τον συγκεκριμένο κόμβο καθώς και για κάθε κόμβο κατά μήκος της διαδρομής προς την ρίζα.

Η παραπάνω διαδικασία φαίνεται και στο Σχήμα 6 [6] .



Σχήμα 6. Λειτουργία Monte Carlo Tree Search

Ο αλγόριθμος του Monte Carlo Tree Search γραμμένος σε ψευδοκώδικα φαίνεται στο Σχήμα 7.

```
01 function MCTS_find_move(state)
02      treeSearch_tree := tree(state)
03      while (time is remaining)
04              Node new_node := find_new_node(treeSearch_tree)
05              v := simulate_evaluate(new_node)
06              treeSearch_tree.add_backpropagate(v,new_node)
07      return best_move(treeSearch_tree)
08
09 function find_new_node(tree)
10      node := root(tree)
11      while (number_unvisited_children(node) = 0)
12              node := find_best_child(node)
13      node := random_unvisited_child(node)
14      return node
```

Σχήμα 7 Ψευδοκώδικας για τον αλγόριθμο MCTS

2.3 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 σε συνδυασμό με κάποια συνάρτηση αξιολόγησης.

Στα παιχνίδια 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] [7], η οποία θα αποδώσει κατάλληλη τιμή στον κόμβο ανάλογη με την εκτίμηση της ποιότητας της αντίστοιχης κατάστασης.

Μια συνάρτηση αξιολόγησης επιστρέφει μια εκτίμηση της αναμενόμενης χρησιμότητας του παιχνιδιού από μια δεδομένη θέση. Έχει ως στόχο λοιπόν να αξιολογεί τις ενδιάμεσες καταστάσεις ενός παιχνιδιού, ώστε να μπορεί να αποφανθεί ποιες από αυτές ενδέχεται να οδηγήσουν τον παίκτη σε επιθυμητό αποτέλεσμα. Ωστόσο, η συνάρτηση αξιολόγησης 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 αφού δεν είναι όλα τα χαρακτηριστικά το ίδιο σημαντικά. Πρέπει να δοθεί 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:

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

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
- 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.

2.4 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.

Οπότε απαιτείται από έναν ορθολογικό πράκτορα όχι μόνο να συλλέγει πληροφορίες αλλά και να μαθαίνει όσο τον δυνατόν περισσότερα αντιλαμβάνεται. Για να επιτευχθεί αυτό, 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. Ο τομέας της μηχανικής μάθησης διακρίνει συνήθως τρείς περιπτώσεις μάθησης: Επιβλεπόμενη (supervised learning), μη επιβλεπόμενη (unsupervised learning) και ενισχυτική (reinforcement learning). Η καταλληλότερη για τους σκοπούς της εργασίας είναι η ενισχυτική μάθηση.

2.4.1 Ενισχυτική Μάθηση

Ενισχυτική μάθηση [8] [7] [1] είναι μία μορφή μάθησης, κατά την οποία ένας πράκτορας τοποθετείται σε ένα περιβάλλον και πρέπει να μάθει να συμπεριφέρεται επιτυχώς μέσα σε αυτό μέσω δοκιμής και αποτυχίας (trial and error). Ο πράκτορας χρειάζεται να γνωρίζει ότι κάτι καλό έχει συμβεί όταν κερδίζει, καθώς και ότι έχει συμβεί κάτι κακό όταν χάνει. Αυτό το είδος ανάδρασης ονομάζεται ανταμοιβή (reward) ή ενίσχυση

(reinforcement). Χρησιμοποιείται σε πολλές εφαρμογές, όπως στον έλεγχο κίνησης ρομπότ, 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 έτσι ώστε να χρησιμοποιήσει αυτές τις πληροφορίες για να μάθει μια συνάρτηση αξιολόγησης που δίνει εύλογα ακριβείς εκτιμήσεις της πιθανότητας νίκης από οποιαδήποτε δεδομένη κατάσταση. 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 ζημιώνεται 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.

2.4.2 TD-Learning

Μία από τις κυριότερες μεθόδους ενισχυτικής μάθησης είναι η μάθηση χρονικών διαφορών (Temporal Difference Learning, TD-Learning) [7] [8]. Σκοπός της είναι να προσπαθήσει να ελαχιστοποιήσει τη διαφορά αξιολόγησης μεταξύ διαδοχικών καταστάσεων.

Σε αυτή τη μορφή μάθησης, προσπαθούμε να μάθουμε τις σωστές τιμές που πρέπει να έχουν τα βάρη μας, χρησιμοποιώντας τις μεταβάσεις από κατάσταση σε κατάσταση. Στόχος, σε κάθε βήμα, είναι να εξαλείψουμε την διαφορά στις τιμές που δίνει η συνάρτηση αξιολόγησης ανάμεσα σε δύο διαδοχικές καταστάσεις (s και s'), όταν ο παίκτης μας επιλέγει την κίνηση που φαίνεται καλύτερη βάσει της τρέχουσας γνώσης του. 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:

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

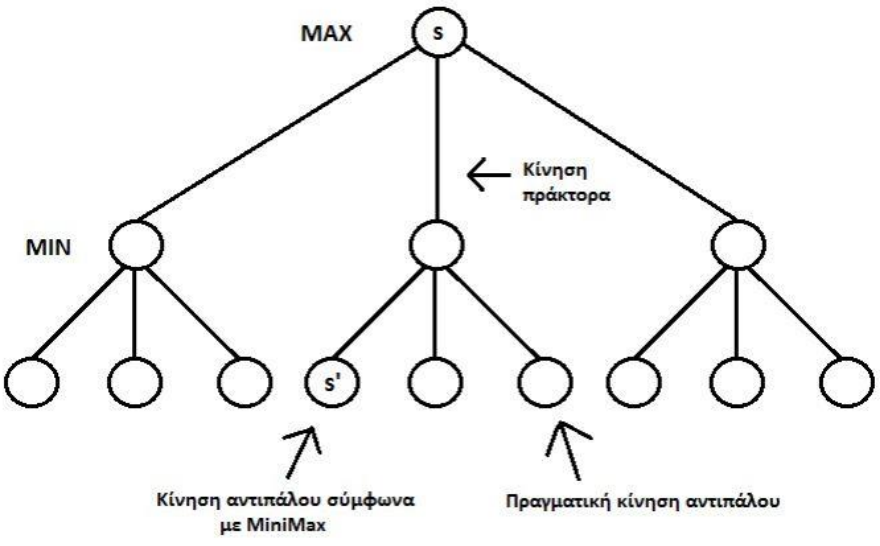
όπου:

- 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.

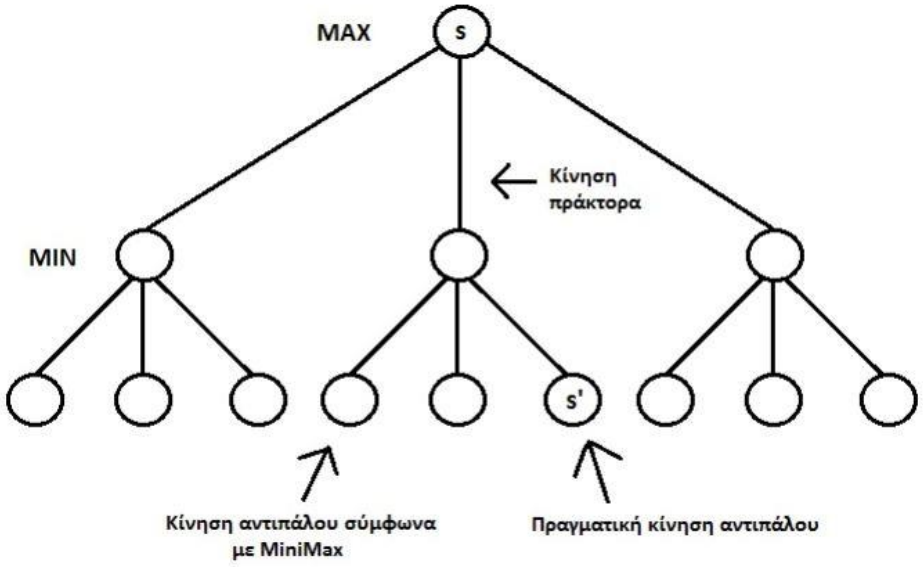
Όσον αφορά την κατάσταση s', αυτή μπορεί να επιλεχθεί με δύο διαφορετικούς τρόπους. Στην πρώτη περίπτωση που φαίνεται στο Σχήμα 8 [9], επιλέγεται να είναι η κατάσταση η οποία προβλέπεται ότι θα προκύψει μετά την βέλτιστη κίνηση του πράκτορα και την βέλτιστη κίνηση του αντιπάλου του, σύμφωνα με το δέντρο παιχνιδιού και την αναζήτηση MiniMax με a-b pruning που έκανε ο πράκτορας. Στη δεύτερη περίπτωση που φαίνεται στο Σχήμα 9 [9], 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



Σχήμα 8. Μάθηση Χρονικών Διαφορών: Περίπτωση 1η

επαναληπτικής διαδικασίας, στα πρώτα στάδια πρέπει να δοθεί 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.



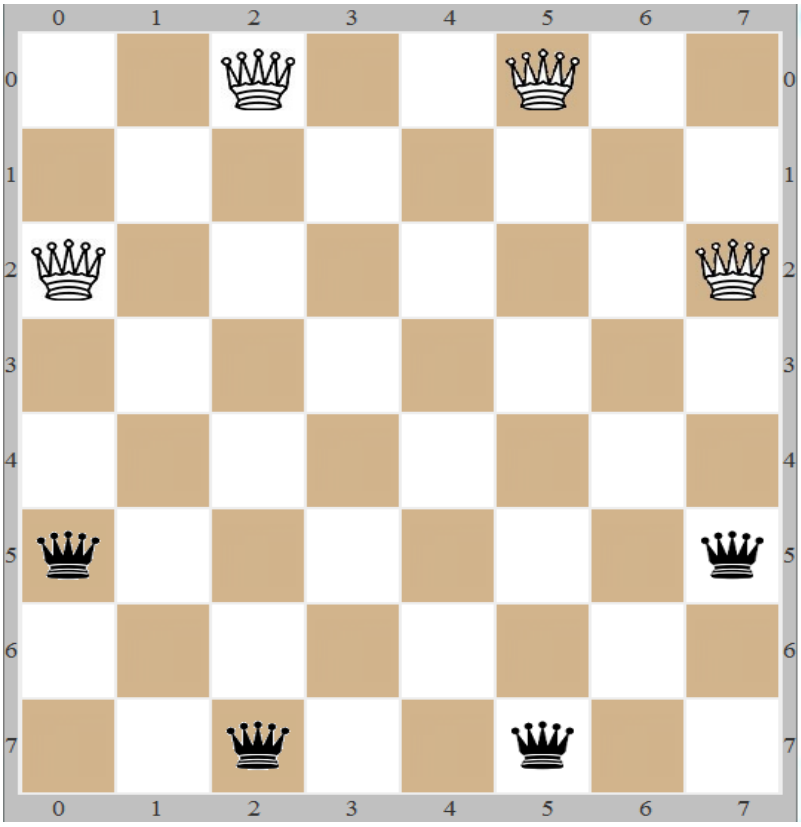
Σχήμα 9. Μάθηση Χρονικών Διαφορών: Περίπτωση $2^{η}$

Κεφάλαιο 3. Περιγραφή Παιχνιδιού “Amazons”

3.1 Amazons

Το παιχνίδι Amazons δημιουργήθηκε από τον Walter Zamkauskas το 1988 στην Αργεντινή [10] [11]. Είναι ένα επιτραπέζιο παιχνίδι 2 παικτών (άσπρος-μαύρος / βόρειος-νότιος) το οποίο είναι πλήρως παρατηρήσιμο, δηλαδή ο κάθε παίκτης γνωρίζει πλήρως την κατάσταση του παιχνιδιού στην οποία βρίσκονται σε οποιαδήποτε στιγμή. Επίσης είναι αιτιοκρατικό, που σημαίνει ότι δεν περιλαμβάνει κάποιας μορφής τυχαιότητα για παράδειγμα με την μορφή καρτών ή ζαριών. Το παιχνίδι επίσης κατατάσσεται στην κατηγορία των territory games, όπως το Go.

Το παιχνίδι Amazons παίζεται σε ένα τετράγωνο ταμπλώ τύπου σκακιέρας διαστάσεων 10×10 (υπάρχουν και παραλλαγές με μικρότερες διαστάσεις σκακιέρας) . Ο κάθε παίκτης διαθέτει 4 πούλια που ονομάζονται Αμαζόνες (Amazons), τα οποία ξεκινάνε σε προκαθορισμένες θέσεις πάνω στο ταμπλό όπως φαίνεται στο σχήμα 10. Αυτή είναι και η αρχική κατάσταση του παιχνιδιού.

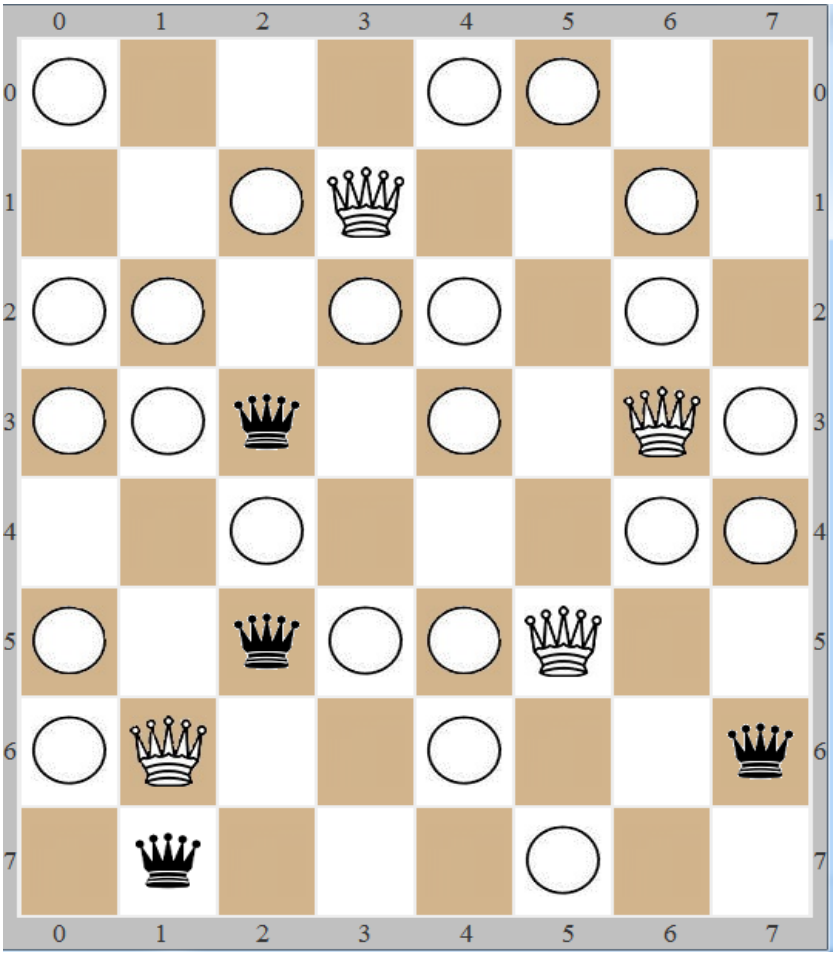


Σχήμα 10. Αρχική κατάσταση παιχνιδιού

Η κάθε Αμαζόνα (Amazon) μιμείται την κίνηση της βασίλισσας από το σκάκι, δηλαδή μπορεί να κινηθεί προς οποιαδήποτε κατεύθυνση για όσες θέσεις επιθυμεί ο παίκτης χωρίς όμως να περάσει πάνω από κάποια άλλη Αμαζόνα ή κάποιο εμπόδιο. Όταν φθάσει στην θέση που επιθυμεί πρέπει να ρίξει ένα βέλος (arrow) προς οποιαδήποτε κατεύθυνση επιθυμεί, αρκεί να μην περάσει πάνω από κάποια άλλη Αμαζόνα ή κάποιο εμπόδιο. Αυτό το βέλος δημιουργεί ένα μόνιμο εμπόδιο στην κενή θέση όπου θα επιλεγεί να τοποθετηθεί κατά μήκος της κατεύθυνσης που επιλέχθηκε, δηλαδή για το υπόλοιπο του παιχνιδιού θα παραμείνει στην συγκεκριμένη θέση χωρίς να μπορεί να μετακινηθεί ή να αφαιρεθεί. Γενικά, πάνω από κάποια θέση-εμπόδιο δεν μπορεί να περάσει ούτε κάποια Αμαζόνα ούτε κάποιο άλλο βέλος ανεξάρτητα από το ποιού παίκτη Αμαζόνα το τοποθέτησε. Μια ενδιάμεση κατάσταση του παιχνιδιού φαίνεται στο σχήμα 11.

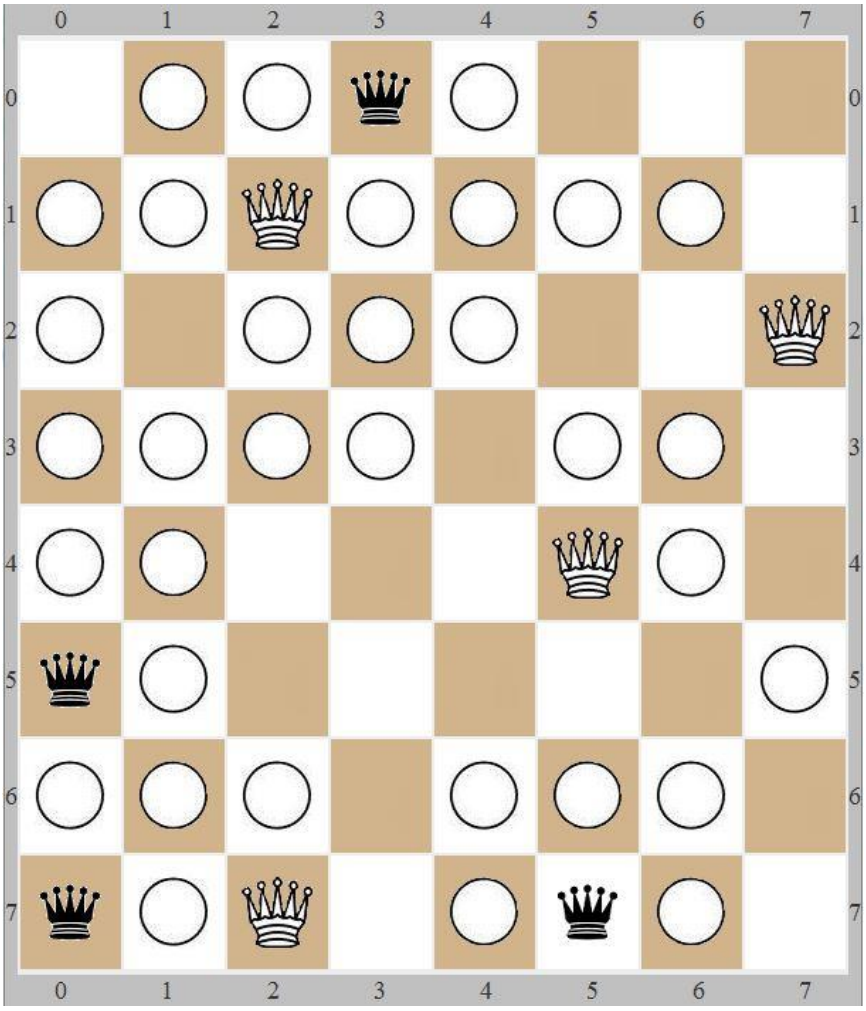
Οπότε μια ολοκληρωμένη κίνηση ενός παίκτη αποτελείται από δύο διαδοχικά στάδια:

1. Κίνηση κάποιας δικής του Αμαζόνας προς οποιαδήποτε διαθέσιμη κατεύθυνση
2. Ρίψη βέλους από αυτή την Αμαζόνα προς οποιαδήποτε διαθέσιμη κατεύθυνση



Σχήμα 11. Ενδιάμεση κατάσταση του παιχνιδιού

Στόχος του κάθε παίκτη είναι να παγιδεύσει τις Αμαζόνες του αντίπαλου παίκτη, ώστε να μην μπορούν να κινηθούν, εφόσον το παιχνίδι τελειώνει όταν κάποιος παίκτης δεν μπορεί να κάνει κίνηση οπότε και χάνει.



Σχήμα 12. Τελική κατάσταση παιχνιδιού. Νικητής ο άσπρος παίκτης

3.2 Στόχος Εργασίας

Στόχος της εργασίας αυτής είναι η σχεδίαση και η ανάπτυξη ενός αυτόνομου πράκτορα που θα μπορεί να παίζει όσο το δυνατόν καλύτερα το παιχνίδι Amazons και θα έχει την δυνατότητα να βελτιώνεται μέσω τεχνικών μηχανικής μάθησης. Επίσης στόχος της εργασίας μας είναι η σκακιέρα του παιχνιδιού να είναι πλήρως παραμετροποιήσιμη ώστε να υπάρχει η δυνατότητα επιλογής διαφορετικού μεγέθους σκακιέρας. 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 την σχεδίαση και την ανάπτυξη του πράκτορα, στόχος είναι

να χρησιμοποιηθούν, αρχικά, τεχνικές αναζήτησης, όπως MiniMax, η βελτίωσή της α-β Pruning και Monte Carlo Tree Search, 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 Learning, TD-Learning), κατά την οποία ο πράκτορας μαθαίνει να βελτιώνει την απόδοσή του στο παιχνίδι. Τέλος, στόχος είναι να συγκριθούν οι διάφοροι παίκτες, που θα κατασκευαστούν ή θα προκύψουν ως στιγμιότυπα του πράκτορα με διαφορετικούς τρόπους μάθησης, μέσα από τη διεξαγωγή τουρνουά και την εξέταση των αποτελεσμάτων.

3.3 Σχετικές Εργασίες

Τον τελευταίο καιρό αποτελεί αντικείμενο έρευνας το παιχνίδι Amazons τόσο στον τομέα της Τεχνητής Νοημοσύνης όσο και της Μηχανικής Μάθησης. Κάποιες σχετικές εργασίες που έχουν δημοσιευθεί σχετικά με το συγκεκριμένο παιχνίδι περιγράφονται παρακάτω.

Ο Theodore Tegos [12] έφτιαξε έναν πολύ δυνατό παίκτη για το παιχνίδι Amazons, το Antiope όπως και το ονόμασε, το οποίο ήταν σύμφωνα με τον ίδιο το πρώτο πρόγραμμα που χρησιμοποιούσε endgame database για το παιχνίδι. Ως endgame database ορίζεται μία βάση δεδομένων στην οποία έχουν προϋπολογιστεί και αποθηκευτεί οι καλύτερες κινήσεις για όλες τις καταστάσεις που βρίσκονται κοντά στο τέλος του παιχνιδιού. Αυτή η τεχνική έχει χρησιμοποιηθεί ήδη σε παιχνίδια όπως το Checkers και το Σκάκι με μεγάλη επιτυχία.

Ο Martin Müller το 2001 [13] απέδειξε ότι για σκακιέρα 5×5, αν έχει τέσσερις Αμαζόνες ο κάθε παίκτης, τότε το παιχνίδι γίνεται first player win, δηλαδή ο παίκτης που παίζει πρώτος μπορεί να νικήσει πάντα ανεξάρτητα από το πως παίζει ο αντίπαλος. Για να το αποδείξει χρησιμοποίησε ένα συνηθισμένο δέντρο παιχνιδιού και έκανε χρήση του αλγορίθμου MiniMax με κάποιες προσθήκες ώστε να μειώσει το μέγεθος και το βάθος του δέντρου.

Ο Jiaxning Song [14] από την στιγμή που είχε αποδειχθεί ήδη ότι η σκακιέρα 5×5 είναι first player win, χρησιμοποιώντας έναν improved solver, όπως χαρακτηριστικά αναφέρει στο paper του, αποδεικνύει ότι και οι σκακιέρες διαστάσεων 4×5, 5×4, 4×6, 5×6 και 4×7 είναι first player win, ενώ η σκακιέρα διαστάσεων 6×4 είναι second player win. Στο συγκεκριμένο paper αναφέρεται κυρίως στις αποδείξεις σχετικά με την σκακιέρα 5×6.

Ο Jens Lieberum [15] δημοσίευσε ένα paper όπου παρουσιάζει μια συνάρτηση αξιολόγησης που χρησιμοποιείται από έναν πράκτορα, τον viz, ο οποίος είναι αρκετά

ανταγωνιστικός απέναντι και σε επαγγελματίες παίκτες του παιχνιδιού. Περιγράφονται οι εξισώσεις που χρησιμοποιήθηκαν και εξηγεί την λογική και τους στόχους που θέλει να επιτύχει με αυτές. Επίσης, παρουσιάζονται κάποια στοιχεία σχετικά με συγκεκριμένα end game προβλήματα του παιχνιδιού. Τέλος, αναλύεται ένα παιχνίδι σε κάποιο τουρνουά απέναντι σε δύο γνωστούς πράκτορες του παιχνιδιού και παρουσιάζονται νέα features που θα μπορούσε η συνάρτηση αξιολόγησής του να χρησιμοποιήσει.

Οι Julien Kloetzer , Hiroyuki Iida , και Bruno Bouzy [16] δημοσίευσαν ένα άρθρο στο οποίο ισχυρίζονται ότι χρησιμοποιώντας τη μέθοδο Monte Carlo κατάφεραν να ξεπεράσουν τη μεγάλη δυσκολία του παιχνιδιού, δηλαδή το πολύ μεγάλο παράγοντα διακλάδωσης. Έτσι, παρουσιάζουν στο άρθρο τους πως εφαρμόζουν το MC στο παιχνίδι Amazons, ώστε να γίνει ο πράκτοράς τους πιο ανταγωνιστικός.

Επίσης, υπάρχουν αρκετοί πράκτορες για το παιχνίδι Amazons που αναπτύσσονται αυτήν την στιγμή και για τον λόγο αυτό δεν δίνονται πληροφορίες δημόσια μιας και συμμετέχουν σε ανταγωνιστικά τουρνουά [17]. Κάποιοι τέτοιοι πράκτορες είναι ο Invader [18], ο Arrow 2 [19], και ο Gamazons [20].

Κεφάλαιο 4. Η Προσέγγισή μας

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 Amazons. Αν και το παιχνίδι παίζεται συνήθως σε σκακιέρα με διαστάσεις 10×10, στην δική μας υλοποίηση επικεντρωθήκαμε σε σκακιέρα με διαστάσεις 8×8. Ωστόσο, λόγω της διαθέσιμης παραμετροποίησης το παιχνίδι και ο πράκτορας μπορούν να λειτουργήσουν σε οποιαδήποτε επιθυμητή διάσταση.

4.1 Χρήση Αναζήτησης MiniMax

Το δέντρο αναζήτησης MiniMax [1] [2] είναι αυτό που μας βοηθάει να υπολογίσουμε όλους τους συνδυασμούς κινήσεων των δύο παικτών δηλαδή του παίκτη Max που είναι πάντα ο πράκτοράς μας και του Min που αντιπροσωπεύει τον αντίπαλο. Υπολογίζοντας όλους τους πιθανούς συνδυασμούς μας δίνεται η δυνατότητα να επιλέξουμε την πιο προσοδοφόρα κίνηση για τον πράκτορά μας.

Κάθε φορά που παίζει ο πράκτορας μας κατασκευάζεται το δέντρο παιχνιδιού. Η κατάσταση που βρίσκεται το παιχνίδι την συγκεκριμένη στιγμή αποτελεί και τον αρχικό κόμβο του δέντρου, την ρίζα. Κάθε επόμενος κόμβος που μπαίνει στο δέντρο αποτελεί μια πιθανή μελλοντική εξέλιξη του παιχνιδιού, εφόσον γίνουν μία ή περισσότερες κινήσεις ανάλογα με το πόσο βαθιά στο δέντρο έχουμε φθάσει από τον κόμβο-ρίζα. Η επέκταση των κόμβων του δέντρου γίνεται με βάση τον αλγόριθμο αναζήτησης με υπαναχώρηση. Στο πρώτο επίπεδο του δέντρου βρίσκεται ο κόμβος-ρίζα που όπως αναφέραμε περιέχει την κατάσταση του παιχνιδιού την συγκεκριμένη στιγμή. Στο επόμενο επίπεδο δημιουργούνται οι κόμβοι-παιδιά οι οποίοι αντιστοιχούν σε όλες τις διαθέσιμες νόμιμες κινήσεις που μπορεί να εκτελέσει ο πράκτοράς μας, ο παίκτης Max, δηλαδή να μετακινήσει σε κάποιο σημείο της σκακιέρας μια Αμαζόνα και να «πυροβολήσει» και ένα βέλος με βάση τους κανόνες. Στο επόμενο επίπεδο παίζει ο Min και περιέχει αντίστοιχα τις κινήσεις που θα μπορεί να κάνει ο επόμενος παίκτης, αφότου εκτελέσει την κίνησή του ο πράκτοράς μας. Με το τρόπο αυτό παίζουν εναλλάξ οι δύο παίκτες αρχικά ο Max και έπειτα ο Min μέχρι να φτάσουμε σε κάποια τελική κατάσταση. Επειδή όμως υπολογιστικά είναι αρκετά χρονοβόρο να προσπαθούμε να φθάσουμε σε τελική κατάσταση του παιχνιδιού, συνήθως επιλέγουμε ένα όριο βάθους D, όπου και σταματάμε. Οπότε, ουσιαστικά ο πράκτοράς μας εξετάζει συστηματικά τις επόμενες D κινήσεις του παιχνιδιού. Στην παρούσα εργασία το

μέγιστο βάθος που εξετάσαμε ήταν το D=3 μιας και ο χρόνος ήταν ήδη αρκετά μεγάλος και αν πηγαίναμε βαθύτερα γινόταν απαγορευτικός. Για τον λόγο αυτό κάποιες φορές σταματούσαμε στο βάθος D=2.

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, γίνεται αξιολόγηση του παιχνιδιού στην εκάστοτε κατάσταση. Εφαρμόζοντας τον αλγόριθμο MiniMax επιλέγεται η καταλληλότερη κίνηση για τον πράκτορά μας. Πρέπει να σημειωθεί ότι στην περίπτωση ισοδύναμων κινήσεων βάσει αξιολόγησης, οι κινήσεις αυτές εισάγονται σε μία λίστα από την οποία επιλέγεται η τελική κίνηση τυχαία.

4.2 Προσθήκη α-β Pruning

Όπως αναφέραμε και παραπάνω, η αναζήτηση τερματίζεται όταν φθάσουμε στο μέγιστο βάθος D. Εφόσον γίνεται χρήση του αλγορίθμου MiniMax η αναζήτηση επιλογής κίνησης είναι αρκετά χρονοβόρα ακόμα και για το βάθος D=3 που χρησιμοποιούμε για τον λόγο ότι ο αλγόριθμος χρειάζεται να αναπτύξει όλους τους κόμβους μέχρι αυτό το βάθος. Με δεδομένο τον μεγάλο παράγοντα διακλάδωσης του παιχνιδιού ο οποίος ξεπερνάει το 1600 (4 Αμαζόνες × 20 πιθανές μετακινήσεις × 20 πιθανές τοποθετήσεις βέλους) στην αρχική κατάσταση, ο αριθμός αυτός είναι υπερβολικά μεγάλος (εκθετικός), περίπου 1600^D για βάθος D. Οπότε, για να μετριαστεί αυτό το πρόβλημα χρησιμοποιήσαμε την βελτίωση α-β Pruning του αλγορίθμου MiniMax.

Ο αλγόριθμος MiniMax με α-β Pruning [1] [2] [3] , βοηθάει στο να αποκόπτονται οι κόμβοι του δέντρου που δεν έχουν κάποια χρησιμότητα. Έτσι οι αποκομμένοι κόμβοι δεν επεκτείνονται και μειώνονται οι συνολικοί κόμβοι που εξετάζονται. Με αυτόν τον τρόπο μειώνεται ο χρόνος που είναι απαραίτητος για την επιλογή κάποιας κίνησης. Ενώ ο χρόνος επιλογής κίνησης μειώθηκε με την χρήση του α-β Pruning παρ' όλα αυτά δεν μπορέσαμε να πάμε σε βάθος μεγαλύτερο του 3, για επιλογή κίνησης σε ένα λογικό χρονικό διάστημα (μέχρι 3 λεπτά). Οπότε, το όριο αυτό δεν μπορούσε να ξεπεραστεί με το υλικό που διαθέταμε και καταλήξαμε στη χρήση ορίου βάθους D=3.

4.3 Προσθήκη Monte Carlo Tree Search

Για να αντισταθμίσουμε το μικρό βάθος αναζήτησης προσθέσαμε κάποιες ιδέες που δανειστήκαμε από τον αλγόριθμο Monte Carlo Tree Search [4] [5]. Η δική μας προσθήκη έχει ως εξής. Αρχικά τρέχουμε πλήρως την αναζήτηση MiniMax με α-β Pruning μέχρι το βάθος D. Κατόπιν, κρατάμε τις 5 πιο ελπιδοφόρες κινήσεις για τον παίκτη μας. Με το πιο ελπιδοφόρες εννοούμε αυτές που είχαν την καλύτερη βαθμολογία με βάση την τιμή MiniMax στα παιδιά της ρίζας.

Έπειτα, για κάθε μία από τις 5 αυτές κινήσεις τρέχουμε μια προσομοίωση του υπόλοιπου παιχνιδιού για να δούμε τι εξέλιξη θα είχε το παιχνίδι σε κάθε περίπτωση. Κατά την προσομοίωση δεν χρησιμοποιούμε τυχαία στρατηγική, αλλά την καλύτερη που διαθέτει ο παίκτης μας αυτή την στιγμή (α-β Pruning με βάθος 2) τόσο γι' αυτόν όσο και για τον αντίπαλο. Με τον τρόπο αυτό εξετάζουμε για κάθε μία απο τις 5 αυτές κινήσεις αν θα έφερναν μια πιθανή νίκη ή ήττα στον πράκτορα μας. Έτσι αν μία κίνηση καταλήξει σε νίκη την θεωρούμε «καλή», ενώ αν καταλήξει σε ήττα την θεωρούμε «κακή». Αν οι «καλές» κινήσεις είναι παραπάνω από μία, επιλέγουμε στην τύχη μια από αυτές, ενώ αν όλες είναι «κακές», τότε επιλέγουμε τυχαία μια από αυτές.

Πειραματικά παρατηρήσαμε ότι η παραπάνω προσθήκη του αλγορίθμου MCTS στην αναζήτηση MiniMax με α-β Pruning σε βάθος D=2 οδηγούσε σε απόδοση συγκρίσιμη με την αναζήτηση MiniMax με α-β Pruning σε βάθος D=3. Έτσι καταφέραμε να μειώσουμε το όριο βάθους από το πολύ χρονοβόρο D=3 στο ανεκτό D=2.

4.4 Συνάρτηση Αξιολόγησης

Αφού περιγράψαμε τους αλγόριθμους αναζήτησης και τα όρια βάθους, συνεχίζουμε με την συνάρτηση αξιολόγησης [1] [7]. Η αρχική ιδέα ήταν να κάνουμε χρήση των διαθέσιμων κινήσεων ως χαρακτηριστικό αξιολόγησης μιας κατάστασης. Ενώ αρχικά φαινόταν καλή ιδέα δεν είχε τόσο καλά αποτελέσματα για το απλό λόγο ότι δεν μπορούσαμε να έχουμε μια σφαιρική άποψη για την κατάσταση του παιχνιδιού ανα πάσα στιγμή. Έτσι ενώ μπορεί να επιλέγαμε μια κίνηση η οποία άφηνε τον αντίπαλο με λιγότερες διαθέσιμες κινήσεις μπορεί μελλοντικά να έφερνε τον πράκτορα μας σε δυσχερή κατάσταση και να έχανε το παιχνίδι. Για τον λόγο αυτό αποφασίσαμε ότι το να επιλέγουμε κίνηση μόνο με βάση τις διαθέσιμες κινήσεις δεν ήταν καλή πρακτική.

Οπότε αναζητήσαμε κάποια χαρακτηριστικά (features) που προσδιορίζουν την κατάσταση του παιχνιδιού και μας δίνουν μια σφαιρική άποψη της κατάστασης του

παιχνδιού σε κάθε στιγμή, ώστε η αξιολόγηση να βασιστεί σε αυτά και να βοηθήσουν στην λήψη αποφάσεων σχετικά με την εύρεση της κατάλληλης κίνησης. Παίρνοντας ιδέες απο την υπάρχουσα βιβλιογραφία, καταλήξαμε σε ένα σύνολο 12 χαρακτηριστικών τα οποία φαίνονται στον παρακάτω πίνακα. Για λόγους ισονομίας, όλα τα χαρακτηριστικά κανονικοποιούνται στο διάστημα [0,1] με βάση τις μέγιστες και ελάχιστες τιμές τους.

Feature	**Περιγραφή**
F1	Διαθέσιμες κινήσεις του πράκτορά μας
F2	Διαθέσιμες κινήσεις του αντιπάλου
F3	Αμαζόνες του πράκτορά μας σε γωνίες
F4	Αμαζόνες του αντιπάλου σε γωνίες
F5	Αμαζόνες του πράκτορά μας με μια μόνο διαθέσιμη κατεύθυνση κίνησης
F6	Αμαζόνες του αντιπάλου με μια μόνο διαθέσιμη κατεύθυνση κίνησης
F7	Αμαζόνες του πράκτορά μας που δεν έχουν τίποτα γύρω τους
F8	Αμαζόνες του αντιπάλου που δεν έχουν τίποτα γύρω τους
F9	Αμαζόνες του πράκτορά μας που είναι παγιδευμένες
F10	Αμαζόνες του αντιπάλου που είναι παγιδευμένες
F11	Αν υπάρχει μία Αμαζόνα του πράκτορά μας σε κάθε τεταρτημόριο
F12	Αν υπάρχει μία Αμαζόνα του αντιπάλου σε κάθε τεταρτημόριο

Τα χαρακτηριστικά F5 και F6 αναφέρονται σε Αμαζόνες οι οποίες μπορούν να κινηθούν προς μια μόνο κατεύθυνση επειδή βρίσκονται σε σημείο τέτοιο που γύρω τους υπάρχουν άλλες Αμαζόνες ή βέλη ή βρίσκονται στα όρια της σκακιέρας. Τα χαρακτηριστικά F7 και F8 αναφέρονται σε Αμαζόνες όπου όλα τα γειτονικά κελιά είναι κενά. Τέλος τα χαρακτηριστικά F11 και F12 αναφέρονται στο, αν χωρίσουμε σε 4 ίσα μέρη τη σκακιέρα (αν το επιτρέπουν οι διαστάσεις), να υπάρχει σε κάθε τέτοιο τεταρτημόριο και μια Αμαζόνα.

Συνολικά τα παραπάνω χαρακτηριστικά μας δίνουν μία γενική ιδέα για το πως είναι η κατάσταση του παιχνιδιού κάθε στιγμή και όταν θέλουμε να αποφασίσουμε κάποια κίνηση έχουμε σφαιρική άποψη για το παιχνίδι. Με τα χαρακτηριστικά και τα βάρη που τους αντιστοιχούν υπολογίζεται η συνάρτηση αξιολόγησης. Τα βάρη αντιπροσωπεύουν τη συμβολή του κάθε χαρακτηριστικού στην αξιολόγηση 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 12 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:

$$\text{Value}(s) = w_1F_1(s) + w_2F_2(s)+\ldots+w_{12}F_{12}(s) = \sum_{i=1}^{12} w_i F_i(s)$$

Όπου:

- s η κατάσταση του παιχνιδιού που αξιολογείται,
- Value(s) η τιμή αξιολόγησης της κατάστασης s,
- F_i η τιμή του χαρακτηριστικού *i*
- w_i το βάρος του χαρακτηριστικού *i*

Στην περίπτωση που μία κατάσταση σε φύλλο του δέντρου είναι τερματική κατάσταση του παιχνιδιού τότε δεν χρησιμοποιείται η συνάρτηση αξιολόγησης, αλλά βαθμολογείται με +100 αν ο πράκτορας μας κερδίζει, ενώ με -100 αν χάνει.

4.5 Προσθήκη 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 πράκτορας μας αντιμετωπίζει τον αντίπαλο σε πάρα πολλά παιχνίδια. Επειδή στο παιχνίδι μας ο αριθμός των κινήσεων που χρειάζεται για να ολοκληρωθεί ένα παιχνίδι δεν είναι σταθερός, αποφασίσαμε η μάθηση να γίνει για κάποιο αριθμό κινήσεων και όχι για ένα συγκεκριμένο αριθμό παιχνιδιών. Μετά από κάθε παιχνίδι 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 [7] [8].

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

όπου:

- 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 στο Κεφάλαιο 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. Για τον ρυθμό μάθησης δοκιμάστηκαν διάφοροι τρόποι μείωσης και τελικά επιλέχθηκε η εκθετική μείωση. Συγκεκριμένα, το α να υποδιπλασιάζεται μετά από έναν συγκεκριμένο αριθμό κινήσεων.

Ανανεώνοντας, λοιπόν, τα βάρη του μετά από κάθε κίνηση, ο παίκτης μας καταλήγει στο τέλος με ένα νέο διάνυσμα βαρών. Αυτό το διάνυσμα είναι το αποτέλεσμα στο οποίο συγκλίνει σταδιακά η μάθηση. Σημειώνεται ότι ο αντίπαλος παραμένει σταθερός κατά τη διάρκεια της μάθησης.

Κεφάλαιο 5. Υλοποίηση

Στην εργασία αυτή, για τον πράκτορα και το γραφικό περιβάλλον, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java [21] [22]. Υλοποιήθηκε στην πλατφόρμα Eclipse IDE με εγκατεστημένο το Java JRE 1.8. Το λειτουργικό σύστημα στο οποίο έγινε η υλοποίηση είναι Windows 7. Το πρόγραμμα έτρεξε σε σύστημα 64bit με 4GB RAM. Οι ελάχιστες απαιτήσεις, ωστόσο, είναι χαμηλότερες. Για την εργασία αυτή δεν προϋπήρχε σχετικός κώδικας, συνεπώς υλοποιήθηκαν όλα 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 σκακιέρας, όριο βάθους επέκτασης, ρυθμός μάθησης, κ.α.), έτσι ώστε όλο σύστημα να είναι πλήρως παραμετροποιήσιμο. Σε αυτήν την κλάση αρχικοποιείται και η σκακιέρα του παιχνιδιού ανάλογα με τις διαστάσεις που έχουν δοθεί.

Την προετοιμασία του παιχνιδιού αναλαμβάνει η συνάρτηση **`game_run`**, στην οποία ορίζεται ο τύπος του κάθε παίκτη (Random, User, MiniMax, Pruning, Monte Carlo), πόσα παιχνίδια θα παιχθούν, ποιος θα παίξει πρώτος κάθε παιχνίδι, τι διαδικασία θα εκτελεστεί (παιχνίδι ή μάθηση) και, τέλος, είναι υπεύθυνη για να τρέξει το παιχνίδι και να παίξουν οι παίκτες. Δηλαδή να καλεί τον κάθε παίκτη να παίξει όταν έρθει η σειρά του, να ελέγχει αν το παιχνίδι έχει τελειώσει, όπως επίσης να κρατάει και το score των παιχνιδιών αν παιχθεί παραπάνω από ένα παιχνίδι.

5.2 MiniMax, α-β Pruning και Monte Carlo

Αφού υλοποιήθηκε το περιβάλλον του παιχνιδιού, συνέχεια δόθηκε με την εφαρμογή των αλγορίθμων αναζήτησης MiniMax και MiniMax με α-β Pruning, όπως επίσης και η προσθήκη Monte Carlo στο παιχνίδι. Στην κλάση **`game_run`**, δημιουργήθηκαν οι αντίστοιχες συναρτήσεις που εκτελούν τους παραπάνω αλγόριθμους, όπως επίσης και οι συναρτήσεις **`MaxMove()`** και **`MinMove()`**. Η **`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 συνάρτηση **ABPruning()** και **MC()** (Monte Carlo), διότι αποτελούν βελτιώσεις του MiniMax.

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 **evaluation_function()** 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, 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 **evaluation** δημιουργήθηκε η συνάρτηση **weights_update()**, η οποία υλοποιεί την ανανέωση των βαρών σύμφωνα με τον αλγόριθμο TD-Learning.

Η συνάρτηση αυτή καλείται από την κλάση **game_run** και τη συνάρτηση μετά από κάθε κίνηση του πράκτορά μας, αν και μόνο αν έχει οριστεί ότι πρόκειται για διαδικασία μάθησης. Η κλάση **game_run** υπολογίζει και παρέχει στην κλήση της **weights_update()** τα απαραίτητα ορίσματα για την ανανέωση.

Το διάνυσμα βαρών όπως και τα χαρακτηριστικά, δεν το παρέχει η κλάση **game_run** . Όπως αναφέρθηκε στην προηγούμενη ενότητα, τα διανύσματα αυτό ορίζονται στην κλάση **evaluation**. Για το λόγο αυτό δημιουργήθηκαν οι συναρτήσεις **getWeights** και **getFeatures**, οι οποίες μας επιστρέφουν τα διανύσματα βαρών και χαρακτηριστικών 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 παιχθεί από κάποιον χρήστη ενάντια στον υπολογιστή ή μεταξύ δύο χρηστών, ήταν απαραίτητη η ύπαρξη γραφικού περιβάλλοντος. Το γραφικό περιβάλλον που δημιουργήθηκε αποτελείται από μία οθόνη στην οποία βλέπουμε το παιχνίδι. 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`** [23]. Μέσα στην κλάση αυτή δηλώνονται όλα τα γραφικά στοιχεία που χρησιμοποιούνται για την απεικόνιση του παιχνιδιού και γίνεται η σωστή διάταξη της σκακιέρας. Επιπλέον, δημιουργείται η συνάρτηση **`move()`**, η οποία υλοποίει την κίνηση ώστε στην συνέχεια η συνάρτηση **`draw()`** να μπορέσει να δείξει γραφικά την κίνηση που μόλις εκτελέστηκε.

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 ανάλυσή τους.

6.1 Πειραματική Διαδικασία

Εκτελέστηκαν αρκετά πειράματα για να εξαχθούν όσο το δυνατό καλύτεροι παίκτες. Σε κάθε πείραμα που εκτελείται, το διάνυσμα βαρών του πράκτορά μας αρχικοποιείται σε μία από τις εξής επιλογές:

- **Zeros** (όλα τα βάρη αρχικοποιούνται με μηδενικά)
- **Ones** (όλα τα βάρη αρχικοποιούνται με μονάδες)
- **Mp** (όλα τα βάρη αρχικοποιούνται με μια δική μας εμπειρική αρχικοποίηση)

Στην διάρκεια της μάθησης το διάνυσμα βαρών του πράκτορα που μαθαίνει τροποποιείται, ωστόσο η αρχικοποίηση είναι πολύ σημαντική γι' αυτό και εξετάστηκαν διαφορετικές αρχικοποιήσεις. Οι αντίπαλοι που αντιμετώπισε ο πράκτορας μας προκειμένου να εκπαιδευθεί ήταν οι παρακάτω:

- **Random** (επιλέγει εντελώς τυχαία κίνηση)
- **AbMp** (χρησιμοποιεί α-β Pruning και τα βάρη Mp)
- **McMp** (χρησιμοποιεί α-β Pruning με Monte Carlo και τα βάρη Mp)

Ο πράκτορας που μαθαίνει χρησιμοποιεί πάντα α-β Pruning με Monte Carlo ωστόσο με μεταβαλλόμενα βάρη τα οποία τροποποιούνται από τον κανόνα μάθησης.

Τα πειράματα μάθησης δεν έγιναν για συγκεκριμένο αριθμό παιχνιδιών, αλλά για συγκεκριμένο αριθμό κινήσεων, συγκεκριμένα για 100.000 κινήσεις από συνεχόμενα παιχνίδια. Τα παιχνίδια αυτά πραγματοποιήθηκαν σε σκακιέρα διαστάσεων 8×8 και ο πράκτοράς μας είχε όριο βάθους επέκτασης D=2. Μετά από κάθε παιχνίδι οι παίκτες αντάλλασαν πλευρά με σκοπό την πληρότητα της μάθησης και από τις δύο πλευρές της σκακιέρας. Ο ρυθμός μάθησης ξεκινάει από την τιμή α=0,1 και υποδιπλασιάζεται κάθε 5.000 κινήσεις. Η τελική του τιμή στο τέλος της μάθησης είναι $\alpha=1{,}9\times10^{-7}$.

Κάθε 50 κινήσεις αποθηκεύεται το τρέχον διάνυσμα βαρών του πράκτορά μας, το οποίο ονομάζεται στιγμιότυπο. Τέλος, για να παρακολουθήσουμε τη σύγκλιση του διανύσματος βαρών, κάθε 50 κινήσεις πάλι υπολογίζεται και αποθηκεύεται η Ευκλείδεια

απόσταση (L2 norm) του τρέχοντος διανύσματος βαρών και του προηγούμενου διανύσματος βαρών πριν από 50 κινήσεις.

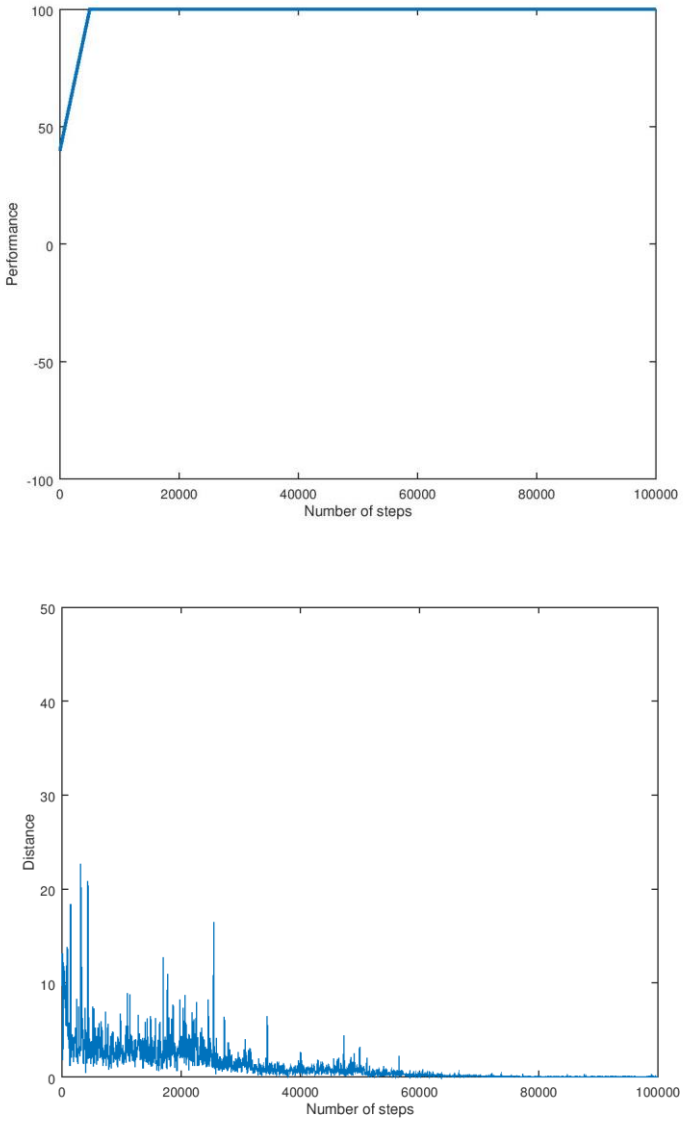
Βάσει του παραπάνω σχεδιασμού ο πράκτορας επικεντρώνεται σε συγκεκριμένους τρόπους επιλογής κινήσεων, οι οποίοι ενδέχεται να μην καλύπτουν όλο το εύρος των πιθανών κινήσεων. Κρίθηκε, λοιπόν, αναγκαίο να προστεθεί και η δυνατότητα εξερεύνησης εντός της διαδικασίας μάθησης. Οπότε, με πιθανότητα e=0.15 ο παίκτης μας δεν ακολουθούσε την κίνηση που του υπεδείκνυε η αναζήτηση, αλλά πραγματοποιούσε κάποια τυχαία κίνηση. Έτσι, ο παίκτης μας περνούσε και από καταστάσεις που ίσως να μην αντιμετώπιζε ποτέ χωρίς εξερεύνηση, με αποτέλεσμα 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 5.000 κινήσεις, το αποθηκευμένο στιγμιότυπο του πράκτορά μας από τη διαδικασία της μάθησης καλείται να αντιμετωπίσει τον αντίπαλο σε 50 παιχνίδια (25 από κάθε πλευρά) με σταθερό διάνυσμα βαρών (χωρίς ενημέρωση βαρών) και χωρίς εξερεύνηση. Για κάθε νίκη βαθμολογείται με +2 πόντους και για κάθε ήττα με -2 πόντους. Άρα, το ανώτατο όριο απόδοσης του παίκτη είναι το +100 και το κατώτατο το -100. Η συγκομιδή των πόντων του πράκτορά μας είναι μία ένδειξη της απόδοσής του.

6.2 Αποτελέσματα Πειραμάτων

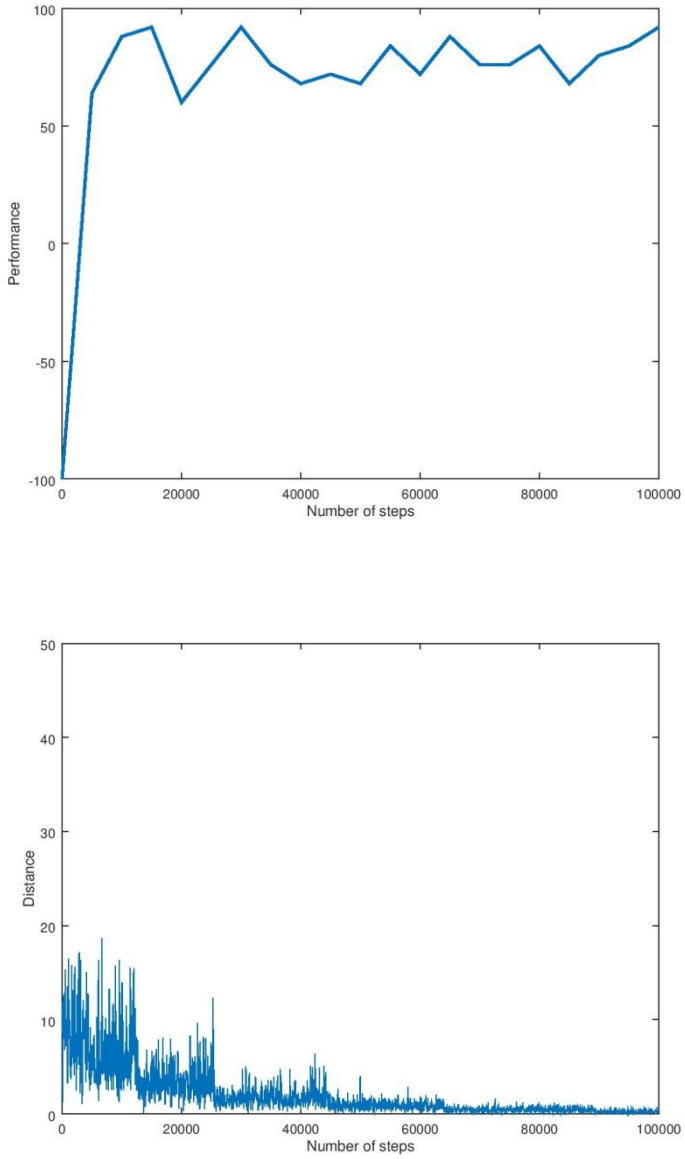
Στην παρούσα ενότητα για κάθε πείραμα, παρουσιάζονται δύο διαγράμματα: (α) η απόδοση του πράκτορά μας ως προς το πλήθος των κινήσεων, και (β) η σύγκλιση του διανύσματος βαρών ως προς το πλήθος των κινήσεων. Κάθε πείραμα κωδικοποιείται με την ονομασία **XxxxVsYyyy**, όπου **Xxxx** η αρχικοποίηση των βαρών του πράκτορά μας και **Yyyy** ο αντίπαλός του κατά την διάρκεια της μάθησης.

Στο πρώτο πείραμα, **ZerosVsRandom**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών μηδενικά και αντιμετωπίζει τον τυχαίο αντίπαλο. Τα αποτελέσματα φαίνονται στο Σχήμα 13. Όπως αναμενόταν, πολύ γρήγορα ο πράκτοράς μας μαθαίνει εύκολα να κερδίζει τον τυχαίο αντίπαλο και να φτάσει στην μέγιστη απόδοση. Επίσης, η σύγκλιση των βαρών επιτυγχάνεται σχετικά γρήγορα περίπου στις 40.000 κινήσεις.



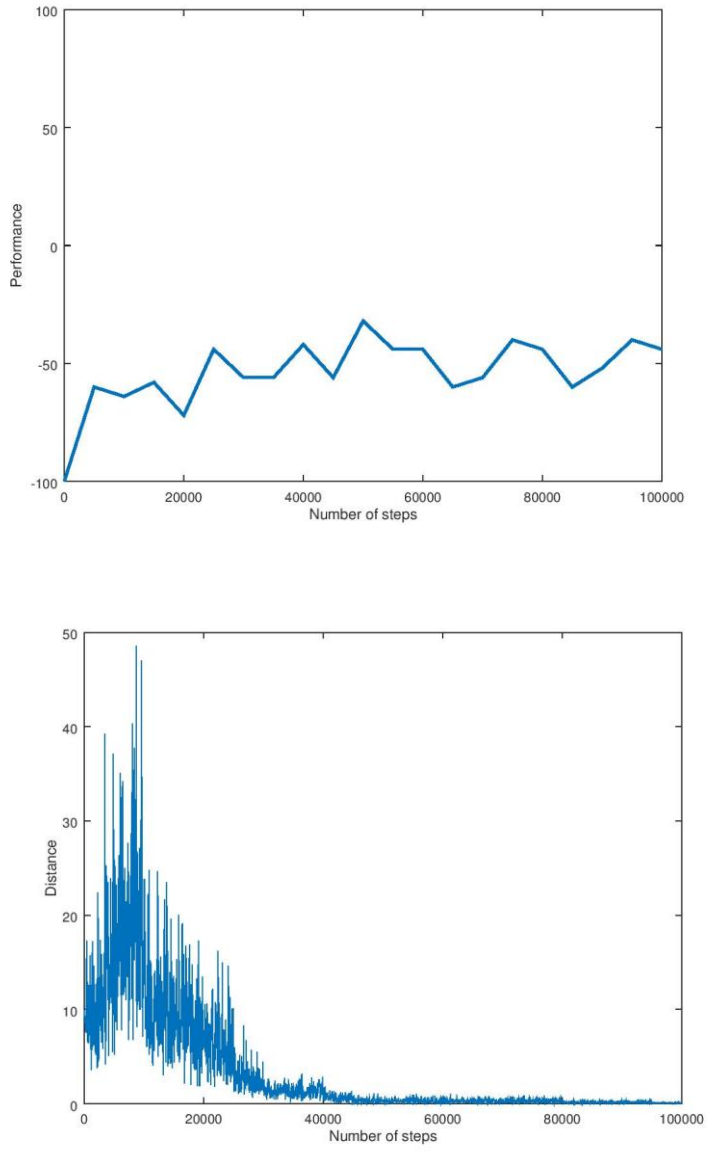
Σχήμα 13 ZerosVsRandom-Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο δεύτερο πείραμα, **ZerosVsAbMp**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών μηδενικά και αντιμετωπίζει αντίπαλο που χρησιμοποιεί α-β Pruning με βάρη Mp. Τα αποτελέσματα φαίνονται στο Σχήμα 14. Όπως περιμέναμε, λόγω κακής αρχικοποίησης ο πράκτοράς μας χάνει όλα τα παιχνίδια στην αρχή μιας και παίζει με ένα καλό αντίπαλο. Όμως αρκετά γρήγορα μαθαίνει και η απόδοσή του αυξάνεται ραγδαία και φθάνει κοντά στο μέγιστο πριν τις 20.000 επαναλήψεις. Επίσης, η σύγκλιση των βαρών δεν επιτυγχάνεται γρήγορα και αυτό οφείλεται στο γεγονός ότι ο αντίπαλός μας είναι καλός. Η σύγκλιση επιτυγχάνεται περίπου στις 65.000 κινήσεις.



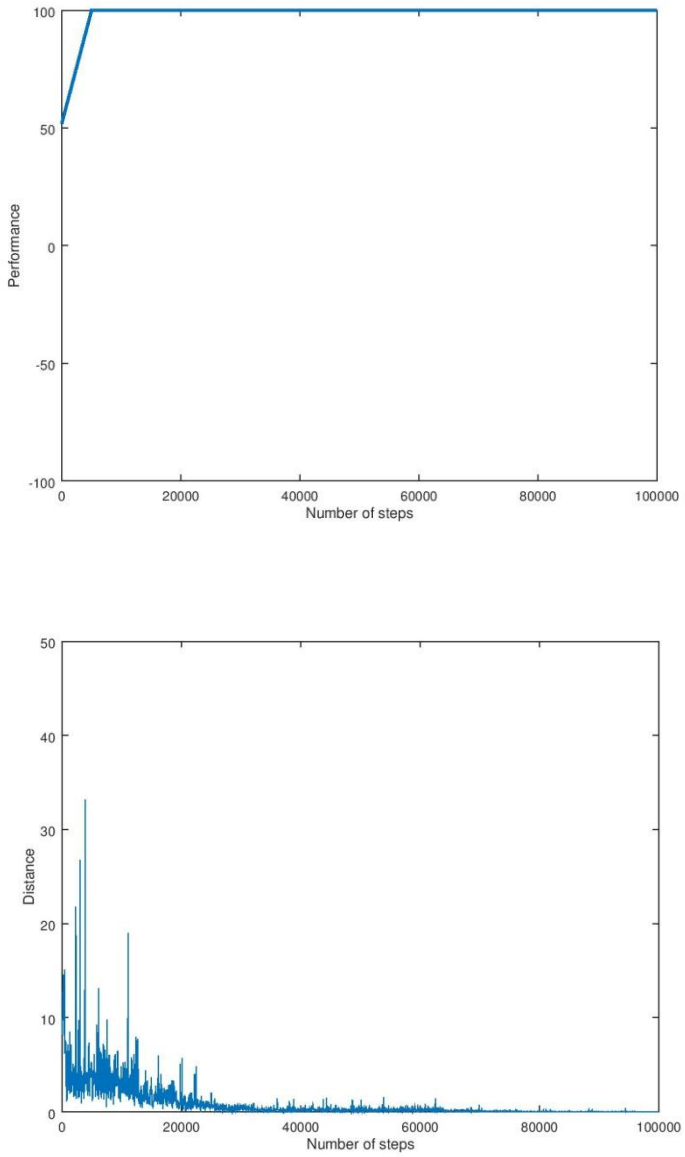
Σχήμα 14 ZerosVsAbMp- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο τρίτο πείραμα, **ZerosVsMcMp**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών μηδενικά και αντιμετωπίζει αντίπαλο που χρησιμοποιεί Mc με βάρη Mp. Τα αποτελέσματα φαίνονται στο Σχήμα 15. Τα αποτελέσματα εδώ δεν είναι τόσο καλά. Εκτός του ότι τα αρχικά βάρη του πράκτορά μας δεν είναι καλά, ο αντίπαλος επιπλέον σε αυτό το πείραμα είναι αρκετά καλός. Οπότε, επειδή το α φθίνει σχετικά γρήγορα και ο δύσκολος αντίπαλος δεν αφήνει πολλά περιθώρια νίκης, δεν προλαβαίνει ο πράκτοράς μας να μάθει να παίζει ανταγωνιστικά εναντίον του και αυτό φαίνεται στην απόδοση. Η σύγκλιση των βαρών επιτυγχάνεται περίπου στις 30.000 κινήσεις.



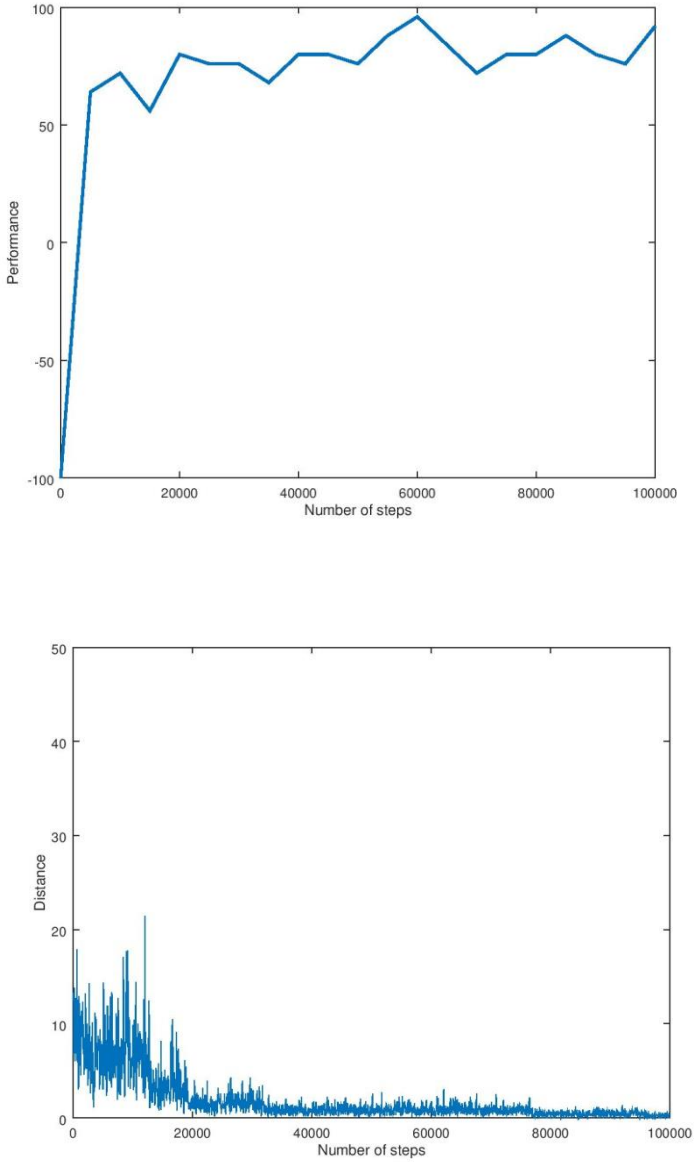
Σχήμα 15 ZerosVsMcMp- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο τέταρτο πείραμα, **OnesVsRandom**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών μονάδες και αντιμετωπίζει τον τυχαίο αντίπαλο. Τα αποτελέσματα φαίνονται στο Σχήμα 16. Όπως αναμενόταν, ξεκινάει καλύτερα από ότι στην αρχικοποίηση με τα μηδενικά και όπως είναι λογικό φτάνει αρκετά γρήγορα στην μέγιστη απόδοση. Επίσης, η σύγκλιση των βαρών επιτυγχάνεται αρκετά γρήγορα, λίγο μετά τις 20.000 κινήσεις.



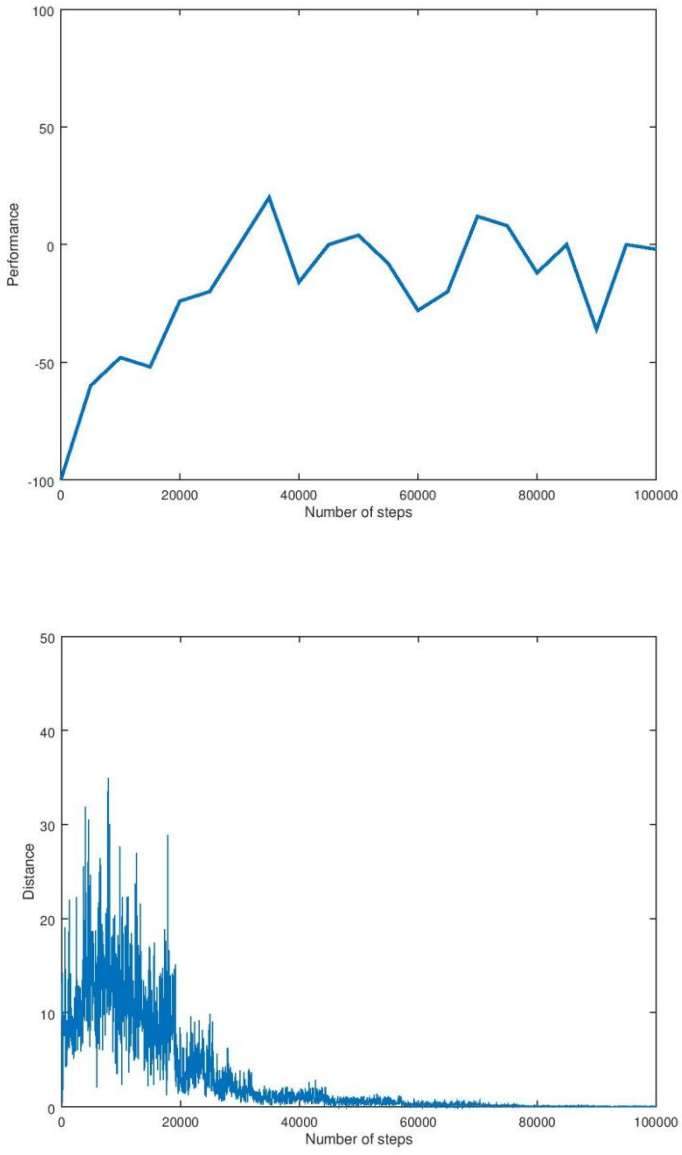
Σχήμα 16 OnesVsRandom- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο πέμπτο πείραμα, **OnesVsAbMp**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών μονάδες και αντιμετωπίζει αντίπαλο που χρησιμοποιεί α-β Pruning με βάρη Mp. Τα αποτελέσματα φαίνονται στο Σχήμα 17. Όπως περιμέναμε, λόγω της όχι και τόσο καλής αρχικοποίησης ο πράκτορας μας χάνει όλα τα παιχνίδια στην αρχή μιας και παίζει απέναντι σε έναν καλό αντίπαλο. Όμως αρκετά γρήγορα μαθαίνει και η απόδοσή του αυξάνεται ραγδαία και φθάνει κοντά στο μέγιστο στις 60.000 επαναλήψεις. Επίσης, η σύγκλιση των βαρών δεν επιτυγχάνεται γρήγορα και αυτό οφείλεται στο γεγονός ότι ο αντίπαλός μας είναι καλός. Η σύγκλιση επιτυγχάνεται περίπου στις 80.000 κινήσεις.



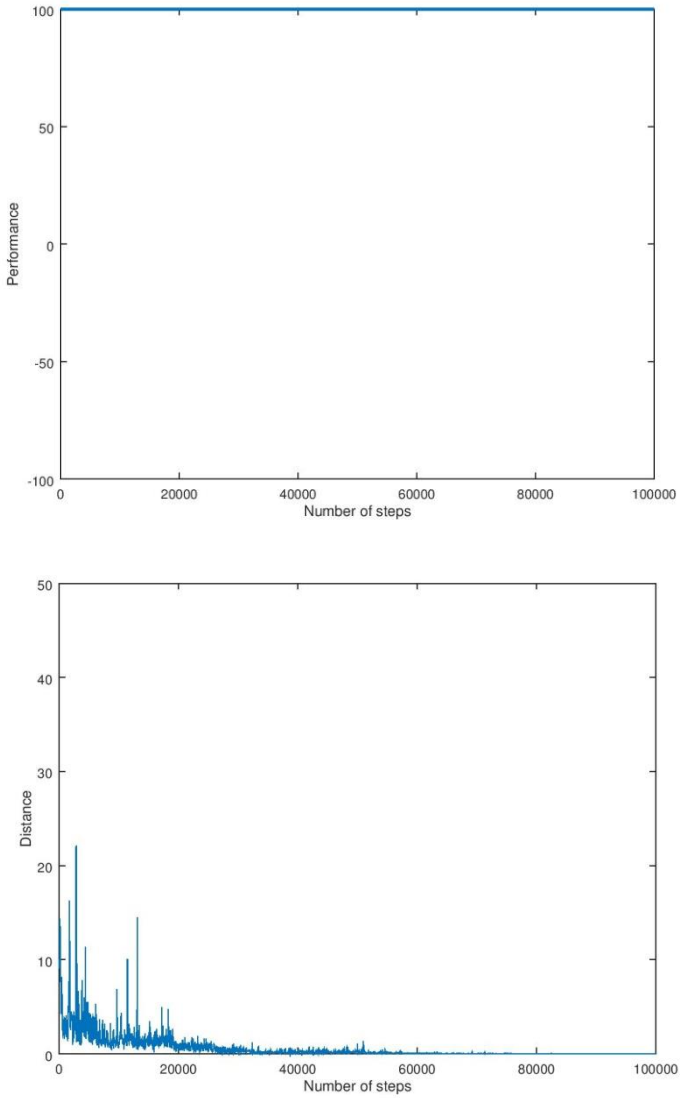
Σχήμα 17 OnesVsAbMp- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο έκτο πείραμα, **OnesVsMcMp**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών μονάδες και αντιμετωπίζει αντίπαλο που χρησιμοποιεί Mc με βάρη Mp. Τα αποτελέσματα φαίνονται στο Σχήμα 18. Τα αποτελέσματα εδώ είναι κάπως καλύτερα σε σχέση με το αντίστοιχο πείραμα με μηδενικά αρχικά βάρη. Αυτό οφείλεται στο ότι η αρχικοποίησή μας με μονάδες είναι λίγο καλύτερη. Όμως το πρόβλημα παραμένει, δηλαδή ο αντίπαλός μας είναι αρκετά καλός, οπότε επειδή το α μηδενίζεται αρκετά γρήγορα δεν προλαβαίνει ο πράκτορας μας να μάθει να παίζει καλύτερα από τον αντίπαλο, αλλά μόνο ισότιμα εναντίον του (ισόπαλο σκορ παιχνιδιών στο τέλος της μάθησης). Η σύγκλιση των βαρών επιτυγχάνεται περίπου στις 50.000 κινήσεις.



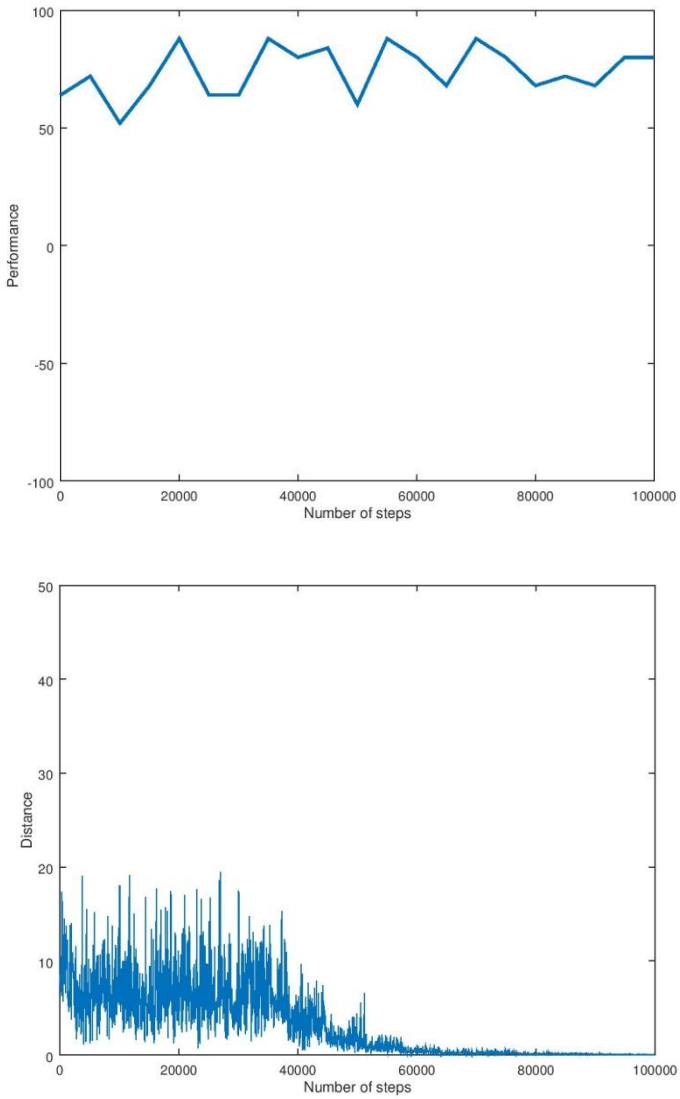
Σχήμα 18 OnesVsMcMp- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο έβδομο πείραμα, **MpVsRandom**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών το MP και αντιμετωπίζει τον τυχαίο αντίπαλο. Τα αποτελέσματα φαίνονται στο Σχήμα 19. Όπως αναμενόταν, ξεκινάει ήδη με πολύ καλά βάρη, οπότε επιτυγχάνει ήδη από την αρχη την μέγιστη απόδοση. Η μάθηση ουσιαστικά δεν συνεισφέρει τίποτα, γι' αυτό και η σύγκλιση των βαρών επιτυγχάνεται αρκετά γρήγορα περίπου στις 20.000 κινήσεις.



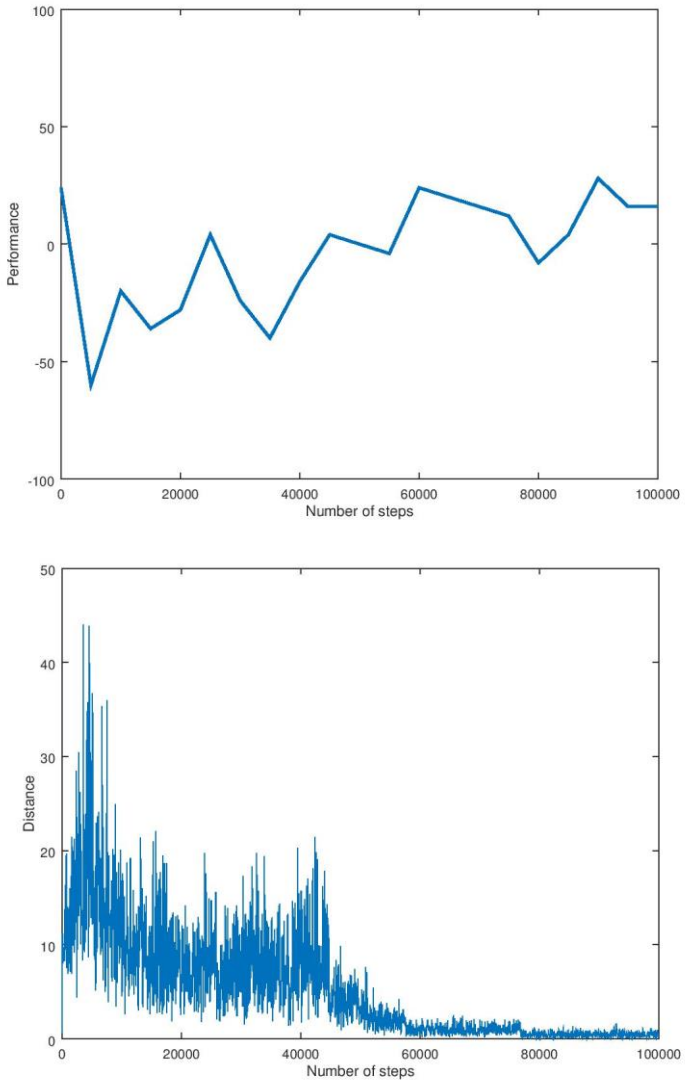
Σχήμα 19 MpVsRandom- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο όγδοο πείραμα, **MpVsAbMp**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών MP και αντιμετωπίζει αντίπαλο που χρησιμοποιεί α-β Pruning με βάρη Mp. Τα αποτελέσματα φαίνονται στο Σχήμα 20. Όπως ήταν αναμενόμενο, έχουμε αρκετά καλή απόδοση από την αρχή με τον πράκτορά μας να κερδίζει περισσότερα παιχνίδια λόγω του ότι είναι καλύτερος ο Mc από τον α-β Pruning παρόλο που τα αρχικά βάρη τους είναι ίδια. Ωστόσο, βλέπουμε ότι ο πράκτορας σχεδόν αμέσως αρχίζει και βελτιώνεται. Η σύγκλιση των βαρών βέβαια αργεί να γίνει και επιτυγχάνεται στις 60.000 κινήσεις.



Σχήμα 20 MpVsAbMp- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

Στο ένατο πείραμα, **MpVsMcMp**, ο πράκτοράς μας έχει ως αρχικό διάνυσμα βαρών MP και αντιμετωπίζει αντίπαλο που χρησιμοποιεί Mc με βάρη Mp. Τα αποτελέσματα φαίνονται στο Σχήμα 21. Όπως ήταν αναμενόμενο, αρχικά τα παιχνίδια ήταν περίπου μοιρασμένα. Αυτό οφείλεται στι ότι παίζουν και οι δύο με τον ίδιο τρόπο. Στην συνέχεια, βλέπουμε ότι ο πράκτοράς μας πέφτει σε απόδοση και ο λόγος είναι πιθανότητα ο παράγοντας της εξερεύνησης. Όμως βλέπουμε ότι στην συνέχεια έχει μια σχετικά ανοδική πορεία. Παρόλα αυτά δεν βλέπουμε μεγάλη βελτίωση, γιατί ο McMp είναι ένας αρκετά δυνατός αντίπαλος και λόγω του ότι το α μειώνεται αρκετά γρήγορα ο πράκτορας μας δεν προλαβαίνει να μάθει να τον αντιμετωπίζει αρκετά εύκολα. Όπως φαίνεται και από το διάγραμμα σύγκλισης αργεί να συμβεί σύγκλιση, αφού επιτυγχάνεται στις 60.000 κινήσεις.



Σχήμα 21 MpVsMcMp- Διαγράμματα απόδοσης και σύγκλισης διανύσματος βαρών

6.3 Αξιολόγηση Αποτελεσμάτων

Για καλύτερη αξιολόγηση των αποτελεσμάτων μας διεξάγαμε ένα τουρνουά του παιχνιδιού Amazons στο οποίο συμμετείχαν 5 αντιπροσωπευτικοί πράκτορες οι οποίοι ανταγωνίστηκαν μεταξύ τους. Περιορίσαμε τους παίκτες σε 5, διότι ήταν χρονοβόρα η διαδικασία διεξαγωγής όλων των αγώνων. Οι 5 πράκτορες που επιλέχθηκαν ήταν: (α) ο τυχαίος παίκτης Random, (β) ο παίκτης MP, (γ) ο παίκτης MpVsMcMp, (δ) ο παίκτης OnesVsAbMp και (δ) ο MpVsRandom. Οι δύο πρώτοι είναι προκατασκευασμένοι παίκτες, ενώ οι τρεις τελευταίοι έχουν προκύψει από τις αντίστοιχες διαδικασίες μάθησης. Κατά την διάρκεια του τουρνουά τα βάρη διατηρούνται σταθερά για όλους τους παίκτες. Στον παρακάτω πίνακα φαίνονται οι τιμές των βαρών για τους 4 παίκτες (πλην του τυχαίου) του τουρνουά.

w_i	Πράκτορας	MP	MpVsMcMp	OnesVsAbMp	MpVsRandom
w_1		5,000	-12,017	57,049	18,065
w_2		-1,200	-273,423	-72,989	-42,819
w_3		-4,500	-2,651	-1,086	-7,108
w_4		0,500	17,907	-3,502	14,267
w_5		-4,000	-106,914	-40,455	10,799
w_6		2,000	71,908	138,272	66,463
w_7		3,000	3,418	3,321	-0,290
w_8		-1,000	-4,294	-10,978	0,550
w_9		-7,000	-181,415	-159,730	-5,554
w_{10}		4,000	198,125	201,602	97,551
w_{11}		1,500	-16,121	13,319	15,994
w_{12}		-0,500	-66,384	-20,559	-2,409

Κάθε παίκτης που συμμετείχε στο τουρνουά έπαιξε με κάθε αντίπαλο 100 παιχνίδια όπου στα μισά έπαιξε πρώτος και στα μισά δεύτερος. Η πολλαπλότητα των παιχνιδιών κρίθηκε απαραίτητη λόγω της τυχαιότητας επιλογής κίνησης σε περιπτώσεις ισοτιμίας. Συνολικά διεξήχθησαν 5 παίκτες × 4 αντίπαλοι × 100 παιχνίδια = 2.000 παιχνίδια. Για κάθε νίκη ο παίκτης έπαιρνε ένα βαθμό, ενώ για κάθε ήττα έχανε ένα βαθμό. Οπότε, αν κάποιος παίκτης κέρδιζε όλα τα παιχνίδια απέναντι σε όλους τους αντιπάλους, θα συγκέντρωνε +400 βαθμούς, ενώ αν κάποιος παίκτης έχανε όλα τα παιχνίδια θα συγκέντρωνε -400 βαθμούς. Τα αποτελέσματα φαίνονται στον παρακάτω πινάκα:

Πράκτορας	Νίκες	Ήττες	Βαθμοί
OnesVsAbMp	317	83	+234
MpVsMcMp	257	143	+114
MP	253	147	+106
MpVsRandom	244	156	+88
Random	0	400	-400

Όπως βλέπουμε και από τα αποτελέσματα οι παίκτες είχαν μεγάλες διαφορές στην βαθμολογία. Ξεχωρίζει ο παίκτης που στην μάθηση ξεκίνησε με τα βάρη του αρχικοποιημένα σε μονάδες και έπαιζε απέναντι στον πράκτορα AbMp. Στην συνέχεια ακολουθούν με μικρή διαφορά μεταξύ τους στην βαθμολογία ο παίκτης που είχε τα αρχικά βάρη του MP και έμαθε απέναντι στον παίκτη McMp και ο προκατασκευασμένος παίκτης MP. Προτελευταίος τερμάτισε ο παίκτης που είχε αρχικά τα βάρη του MP και έμαθε απέναντι στον Random, ενώ τελευταίος χωρίς να κερδίσει ούτε ένα παιχνίδι ήταν φυσικά ο Random παίκτης, αφού όλοι οι παίκτες ήταν αρκετά «έξυπνοι» ώστε να τον κερδίσουν. Παρατηρούμε βέβαια ότι στην μάθηση παίζει μεγαλύτερη σημασία απέναντι σε ποιον πράκτορα μαθαίνει ο πράκτοράς μας, παρά τα αρχικά του βάρη. Βλέπουμε ότι στο τουρνουά τα πήγε καλύτερα ο MP που δεν συμμετείχε στην μάθηση παρά ο παίκτης με αρχικά βάρη του Mp αλλά με μάθηση απέναντι στον Random. Επίσης, προκαλεί εντύπωση ότι ο παίκτης που ξεκίνησε με μέτρια βάρη (μονάδες) και έπαιζε απέναντι σε ένα σχετικά καλό παίκτη (όχι τον καλύτερο) είχε την καλύτερη επίδοση στο τουρνουά. Για τον Random το αποτέλεσμα ήταν αναμενόμενο μιας και είναι αρκετά δύσκολο παιχνίδι για κάποιον που παίζει στην τύχη, χωρίς ούτε καν μια απλή στρατηγική.

Αξίζει να παρατηρήσει κανείς τα βάρη του καλύτερου παίκτη του τουρνουά, OnesVsAbMp. Δίνει μεγάλο θετικό βάρος στα χαρακτηριστικά F_1, F_6, F_{10} δηλαδή στις διαθέσιμες κινήσεις του, στις Αμαζόνες του αντιπάλου με μία μόνο κατεύθυνση κίνησης και στις παγιδευμένες Αμαζόνες του αντιπάλου. Στα αντίστοιχα χαρακτηριστικά F_2, F_5, F_9 δηλαδή στις διαθέσιμες κινήσεις του αντιπάλου, στις Αμαζόνες του πράκτορα με μία μόνο κατεύθυνση κίνησης και στις παγιδευμένες Αμαζόνες του πράκτορα δίνει τα μεγαλύτερα αρνητικά βάρη.

Κεφάλαιο 7. Συμπεράσματα

7.1 Συζήτηση

Το επιτραπέζιο παιχνίδι Amazons είναι ένα αρκετά ενδιαφέρον παιχνίδι. Η μεγάλη του δυσκολία έγκειται στο γεγονός ότι έχει μεγάλο αριθμό διαθέσιμων κινήσεων, οπότε χρειάζεται μεγάλη προσοχή για την κάθε κίνηση και για την πρόβλεψη της κίνησης του αντιπάλου για να οδηγηθεί σε αξιόπιστη στρατηγική.

Στην παρούσα εργασία σχεδιάσαμε και αναπτύξαμε έναν αυτόνομο πράκτορα με δυνατότητες μάθησης για το παιχνίδι Amazons. Ο πράκτορας μας μπορεί να εξετάσει έως και 3 κινήσεις μπροστά, όμως ο χρόνος αναζήτησης ανά κίνηση είναι σχετικά μεγάλος. Η εναλλακτική λύση ήταν να εξετάζει 2 κινήσεις μπροστά και να χρησιμοποιεί τεχνικές Monte Carlo για να αντισταθμίσει αυτή την μείωση.

Για την δημιουργία του πράκτορά μας χρησιμοποιήσαμε τον αλγόριθμο MiniMax με α-β Pruning (με συμμετοχή και του Monte Carlo Tree Search) και τον αλγόριθμο μάθησης TD-Learning. Οι παραπάνω αλγόριθμοι οδήγησαν στην δημιουργία αξιόλογων παικτών που αποτελούν δύσκολους αντιπάλους για έναν μέσο παίκτη, όπως επιβεβαιώνεται από δοκιμαστικές παρτίδες με ανθρώπους που είχαν κάποια εμπειρία με το παιχνίδι.

7.2 Μελλοντικές Επεκτάσεις

Η εργασία αυτή έχει περιθώρια βελτίωσης ως προς την επιλογή αλγορίθμου αναζήτησης. Λόγω του μεγάλου αριθμού κινήσεων, ίσως μπορούν να χρησιμοποιηθούν άλλες τεχνικές αναζήτησης, ώστε να μην είναι τόσο χρονοβόρα η επιλογή κίνησης και κατά συνέπεια θα μπορούσε να επιτευχθεί μεγαλύτερο βάθος επέκτασης το οποίο θα είχε αισθητή βελτίωση στον πράκτορά μας.

Επίσης, στην διαδικασία μάθησης μπορούν να γίνουν περισσότερα πειράματα και με διαφορετικούς, πιθανότατα μεταβαλλόμενους, αντιπάλους, διαφορετικές ρυθμίσεις σε εξερεύνηση και ρυθμό μάθησης και ίσως και σε διαφορετικού μεγέθους σκακιέρες. Ακόμα θα μπορούσε να γίνει αναθεώρηση των χαρακτηριστικών της συνάρτησης αξιολόγησης για να γίνεται καλύτερη αναπαράσταση των καταστάσεων. Επίσης, θα μπορούσε να χρησιμοποιηθεί κάποιος προηγμένος αλγόριθμος μάθησης που επεξεργάζεται μαζικά δεδομένα εκπαίδευσης. Έτσι θα βελτιωνόταν ακόμα περισσότερο ο πράκτοράς μας.

Μια σημαντική προσθήκη η οποία θα προσέδιδε μεγάλη βελτίωση στο πράκτορά μας θα ήταν η εισαγωγή opening book και endgame database, οι οποίες είναι τεχνικές που

χρησιμοποιούνται αρκετά σε παιχνίδια με μεγάλο αριθμό κινήσεων, όπως το σκάκι και το Checkers με μεγάλη επιτυχία.

Βιβλιογραφία

[1] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2003.

[2] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας και Η. Σακελλαρίου, Τεχνητή Νοημοσύνη, Θεσσαλονίκη: Εκδόσεις Πανεπιστημίου Μακεδονίας, 2011.

[3] D. E. Knuth and R. W. Moore, "An Analysis of Alpha-Beta Pruning," *Artificial Intelligence,* vol. 6, no. 4, pp. 293-326, 1975.

[4] B. Bouzy, "Associating domain-dependent knowledge and Monte Carlo approaches within a Go program," *Information Sciences, Heuristic Search and Computer Game Playing IV,* vol. 175, no. 4, pp. 247-257, 2005.

[5] B. Bouzy, "Move Pruning Techniques for Monte-Carlo Go," in *11th International Conference on Advances in Computer Game (ACG)*, Taipei, Taiwan, 2005.

[6] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 4, no. 1, pp. 1-43, March 2012.

[7] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 1998.

[8] L. P. Kaelbling, M. L. Littman and A. G. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research,* vol. 4, no. 1, pp. 237-285, 1996.

[9] Σ. Βαγενάς, «Συστηματική Αναζήτηση και Ενισχυτική Μάθηση για το Επιτραπέζιο Παιχνίδι Turning Points,» Διπλωματική Εργασία, Σχολή ΗΜΜΥ, Πολυτεχνείο Κρήτης, Χανιά, 2016.

[10] "The Game of Amazons - Invader," [Online]. Available: http://www.csun.edu/~lorentz/amazon.htm#Introduction.

[11] "Game of the Amazons," [Online]. Available: https://en.wikipedia.org/wiki/Game_of_the_Amazons#cite_note-chessvariants-1.

[12] T. Tegos, "Shooting the last arrow," Master's Thesis, University of Alberta, Alberta, Canada, 2002.

[13] M. Müller, "Solving 5x5 Amazons," in *The 6th Game Programming Workshop (GPW 2001)*, Hakone, Japan, 2001.

[14] J. Song and M. Mueller, "An enchanced solver for the game of Amazons," *Transactions on Computational Intelligence and AI in Games,* vol. 7, no. 1, pp. 16-27, 2013.

[15] J. Lieberum, "An evaluation function for the game of amazons," *Theoretical Computer Science,* vol. 349, no. 2, pp. 230-244, 2005.

[16] J. Kloetzer, H. Iida and B. Bouzy, "The Monte-Carlo Approach in Amazons," in *Computer Games Workshop*, Amsterdam, The Netherlands, 2007.

[17] "ICGA Tournaments," [Online]. Available: https://www.game-ai-forum.org/icga-tournaments/game.php?id=15.

[18] "Invader," [Online]. Available: http://www.csun.edu/~lorentz/amazon.htm#Download.

[19] "Arrow2," [Online]. Available: https://webdocs.cs.ualberta.ca/~mmueller/amazons/arrow.html.

[20] "Gamazons," [Online]. Available: http://www.yorgalily.org/gamazons/.

[21] W. J. Savitch, Absolute Java, Pearson, 2012.

[22] Oracle, "Java Documentation," 2017. [Online]. Available: docs.oracle.com/en/java/.

[23] "Oracle Help Center-JFrame," [Online]. Available: https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html.