Τεχνητή Νοημοσύνη – 1η Εργασία

ΑΝΑΦΟΡΑ ΠΑΡΑΔΟΣΗΣ

(**Όνομα:** Αναστάσιος Παπαπαναγιώτου, **ΑΜ:** 3200143, **Email:** p3200143@aueb.gr)

(**Όνομα:** Φοίβος Παπαθανασίου, **ΑΜ:** 3200138, **Email:** p3200138@aueb.gr)

Περίληψη:

Υλοποίηση παιχνιδιού Othello για έναν παίκτη με αντίπαλο τον υπολογιστή. Για την επιλογή των κινήσεων του υπολογιστή χρησιμοποείται ο αλγόριθμος MiniMax με πριόνισμα α-β. Για την αξιολόγηση καταστάσεων χρησιμοποείται συνάρτηση η οποία αποτελεί γραμμικό άθροισμα συναρτήσεων. Για τον προσδιορισμό των βαρών κατασκευάστηκαν δύο διαφορετικοί αλγόριθμοι (Γενετικός και 1v1 Knockout Elimination Style Tournament) και τέλος επιλέχθηκε το καλύτερο σετ βαρών εκ των δύο.

Τρόπος Χρήσης:

Αρχικά, ο χρήστης επιλέγει εάν θα παίξει πρώτος ή δεύτερος πληκτρολογώντας “yes” ή “no” ανάλογα. Στη συνέχεια, ο παίκτης επιλέγει το μέγιστο βάθος αναζήτησης του αλγορίθμου MiniMax.

Ακολούθως επιλέγει πού θα τοποθετήσει το πούλι προσδιορίζωντας τη γραμμή και ύστερα τη στήλη του πίνακα.(επαναληπτικά)

Αρχεία:

- board.py

  Περιέχει την κλάση **Board** η οποία είναι η υλοποίηση του board του παιχνιδιού, επίσης περιέχει τη βοηθητική κλάση **PossibleMoves** η οποία χρησιμεύει στο να κρατούνται οι δυνατές κινήσεις ενός παίκτη.

  Μεθόδοι Κλάσης Board:

  full() – Επιστρέφει True εάν ο πίνακας είναι γεμάτος.

  printBoard() – Για τη γραφική αναπαράσταση του board state στο console.

  hasLegalMoves(player) – Επιστρέφει True εάν ο player έχει έστω και μία legal κίνηση, διαφορετικά False.

  findMoves(row, col, diskColor) – Κατασκευάζει αντικείμενο possibleMoves και το ενημερώνει με όλα τα ranges στα οποία αλλάζουν χρώμα τα πούλια

όταν το πούλι με χρώμα diskColor τοποθετηθεί στη θέση (row, col) του πίνακα.

makeMove(row, col, diskColor) – Κάνει τις απαραίτητες τροποποιήσεις στο board για να αντικατοπτρίσει την τοποθέτηση μιας μάρκας με χρώμα diskColor στη θέση (row, col) του πίνακα.

outflankHorizontally(row, col, diskColor) – Επιστρέφει το οριζόντιο range που επηρεάζεται από την τοποθέτηση δίσκου χρώματος diskColor στη θέση (row, col) του πίνακα.

outflankVertically(row, col, diskColor) – Αντίστοιχα με παραπάνω αλλά το κάθετο range.

outflankDiagonally(row, col, diskColor) – Προσδιορίζει την/τις διαγωνίους που επηρεάζονται από την τοποθέτηση δίσκου χρώματος diskColor στη θέση (row, col) του πίνακα.

isTerminal() – Επιστρέφει True εάν το board βρίσκεται σε τελική κατάσταση – δηλαδή έαν το παιχνίδι έληξε, διαφορετικά επιστρέφει False.

getChildren() – Επιστρέφει λίστα που περιέχει όλες τις θέσεις του πίνακα οι οποίες είναι κενές.

getCopy() – Επιστρέφει βαθύ αντίγραφο του αντικειμένου πίνακα, απαραίτητη για τον αλγόριθμο MiniMax.

getLegalMoves(diskColor) – Επιστρέφει λίστα που περιέχει όλες τις επιτρεπτές κινήσεις για τον παίκτη με χρώμα diskColor.

Ορισμένες βοηθητικές μεθόδοι παραλείπονται, επίσης οι μεθόδοι της βοηθητικής κλάσης **PossibleMoves** είναι βοηθητικές και παραλείπονται.

- determine_weights.py

  Αλγόριθμος προσδιορισμού βαρών.
  Κατασκευάζονται M = 2^κ λίστες με N = 2^λ σετ βαρών η καθεμία και πραγματοποιείται για κάθε λίστα ένα 1v1 Knockout Elimination Style Τουρνουά όπου κάθε σετ βαρών αντιπροσωπεύει έναν παίκτη. Στη συνέχεια οι νικητές (σετ βαρών) των τουρνουά συνιστούν μία λίστα μεγέθους M με την οποία πραγματοποιείται ένα τελευταίο τουρνουά και εκτυπώνεται ο νικητής. («βέλτιστο» σετ βαρών)

  Συναρτήσεις:

  tournament(weights) – Πραγματοποιείται ένα τουρνουά με τη δοθείσα λίστα από σετ βαρών και επιστρέφεται ο νικητής (σετ βαρών).

battle(blackWeights, whiteWeights, maxDepth =2) – Τρέχει μία παρτίδα Othello μεταξύ δύο υπολογιστών με βάρυ blackWeights και whiteWeights αντίστοιχα. Επιστρέφει το σετ βαρών του νικητή καθώς επίσης 0 αν νίκησε ο άσπρος, 1 αν νίκησε ο μαύρος ή -1 αν ήρθε ισοπαλία (τότε επιστρέφεται τυχαίο σετ βαρών με πιθανότητα 0.5).

generateStartingWeights(numOfTuples, numOfWeights) – Κατασκευάζει και επιστρέφει λίστα που περιέχει numOfTuples σετ βαρών μεγέθους numOfWeights. Τα tuples κατασκευάζονται με τον Τροποποιημένο Αλγόριθμο Kramer[1] ώστε τα βάρυ να έχουν κανονική κατανομή.

generateTuple(numOfWeights, sm) – Κατασκευάζει και επιστρέφει σετ βαρών με χρήση του Τροποποιημένου Αλγορίθμου Kramer[1]. Το άθροισμα των βαρών είναι ίσο με sm.

determine(numOfTuples, numOfContestants, numOfWeights) – Κύρια Συνάρτηση. Πραγματοποιεί numOfTuples τουρνουά με numOfContestants παίκτες ανα τουρνουά και σετ βαρών μεγέθους numOfWeights. Ύστερα πραγματοποιεί τελικό τουρνουά με τους νικητές των τουρνουά και εκτυπώνει τον νικητή (σετ βαρών).

- genetic_determine_weights.py

Γενετικός Αλγόριθμος προσδιορισμού βαρών.
Κάθε χρωμόσωμα αντιπροσωπεύει ένα σετ βαρών (χρωμόσωμα := σετ βαρών) και κάθε βάρος είναι ένα γονίδιο. Ο αρχικός πλυθησμός είναι μία λίστα από χρωμοσώματα.

Για την αξιολόγηση των χρωμοσωμάτων, κάθε χρωμόσωμα πραγματοποιεί battle() (βλ. determine_weights.py) με όλα τα υπόλοιπα ως μαύρος παίκτης, για κάθε νίκη παίρνει έναν πόντο, για κάθε ήττα παίρνει το αντίπαλο χρωμόσωμα έναν πόντο και στην ισοπαλία κανένα χρωμόσωμα δεν παίρνει πόντο. Συνεπώς η πιθανότητα επιλογής κάποιου χρωμοσώματος για αναπαραγωγή εξαρτάται από το πλήθος νικών του χρωμοσώματος.

Επειδή το άθροισμα των βαρών κάθε σετ βαρών πρέπει να ισούται με 1, οι λειτουργίες cross-over και mutation έχουν τροποποιηθεί ανάλογα. (βλ. Συναρτήσεις)

Συναρτήσεις:

runGeneration(…) – Κύρια Συνάρτηση, συνηθισμένη υλοποίηση γενετικού αλγορίθμου. Επιστρέφει το «βέλτιστο» σετ βαρών.

evaluate(evaluationChromosomes, maxDepth) – Πραγματοποιεί τα battles μεταξύ των χρωμοσωμάτων και επιστρέφει λίστα που περιέχει πιθανότητα επιλογής για αναπαραγωγή για το κάθε χρωμόσωμα.

reproduce(x, y) – Πραγματοποιεί cross-over μεταξύ δύο χρωμοσωμάτων. Προκειμένου να διατηρηθεί το άθροισμα των βαρών κάθε σετ βαρών ίσο με 1, το range 0 εώς splitIdx-1 θεωρείται fixed για το tuple x και το range splitIdx εώς len(y)-1 θεωρείται fixed για το tuple y. Συνεπώς, το variable range του tuple x λαμβάνει νέες τιμές με βάση τις αναλογίες των τιμών του ίδιου range του tuple y και αντιστρόφως. Για την αποφυγή ασυνεπειών τα tuple's x,y εναλλάσονται με πιθανότητα 50%.

reproduce2(x, y) – Πραγματοποιεί cross-over μεταξύ δύο χρωμοσωμάτων. Προκειμένου να διατηρηθεί το άθροισμα των βαρών κάθε σετ βαρών ίσο με 1, αφού πραγματοποιηθεί το cross-over τα βάρη των σετ βαρών κανονικοποιούνται με το συνολικό άθροισμα των βαρών τους.

mutate(x, mutationProbability) – Επιλέγει από 0 εώς len(x) βάρυ από το σετ βαρών x και επανακατανέμει τυχαία το συνολικό άθροισμα των επιλεγμένων βαρών.

- player.py

Η κλάση Player είναι η υλοποίηση του υπολογιστή που παίζει εναντίον του παίκτη. Ο υπολογιστής χρησιμοποιεί τον αλγόριθμο MiniMax με α-β πριόνισμα για την επιλογή των κινήσεων του.
Στη συγκεκριμένη υλοποίηση όταν κατασκευάζεται ένα αντικείμενο Player πρέπει να προσδιορισθεί το χρώμα του παίκτη καθώς καλείται η max αρχικά ανεξάρτητα του χρώματος – δηλαδή και στις δύο περιπτώσεις ο παίκτης προσπαθεί να μεγιστοποιήσει τη νίκη του, η συγκεκριμένη προσέγγιση λειτουργεί καθώς το παιχνίδι είναι zero sum (είναι ισοδύναμο με το να ελαχιστοποιεί την ήττα του – δηλαδή να ξεκινάει με κλήση της min).

Μεθόδοι Κλάσης Player:

miniMax(board) – Επιστρέφει τις συντεταγμένες της βέλτιστης κίνησης για τον Player. Πραγματοποιεί την αρχική κλήση της μεθόδου maxValue(...).

maxValue(board, depth, a, b) – Επιλέγει την κίνηση (ενημερώνει τις βέλτιστες συντεταγμένες) που μεγιστοποιεί το όφελος του Player. Πραγματοποιεί και b-πριόνισμα.

minValue(board, depth, a, b) – Επιλέγει την κίνηση (ενημερώνει τις βέλτισες συντεταγμένες) που ελαχιστοποιεί το όφελος του Player. Πραγματοποιεί και a-πριόνισμα.

- utilities.py

Το αρχείο αυτό περιέχει τις συναρτήσεις αξιολόγησης οι οποίες συμμετέχουν στο γραμμικό άθροισμα της συνάρτησης οφέλους του αλγορίθμου miniMax.

Συναρτήσεις:

u(board, diskColor, weights) – Η συνάρτηση οφέλους η οποία αποτελεί γραμμικό άθροισμα των συναρτήσεων αξιολόγησης.

u1(board, diskColor) – Επιστρέφει τη διαφορά στα πούλια χρώματος diskColor από το αρχικό board state μέχρι και το μέγιστου βάθους αναζήτησης (ή terminal state) board state.

u2(board) – Επιστρέφει το άθροισμα όλων των δυνατών κινήσεων του max παίκτη από το αρχικό board state μέχρι το μέγιστο βάθος.

u3(board, diskColor) – Επιστρέφει μία τιμή η οποία αξιολογεί τις θέσεις των πούλιων του παίκτη με χρώμα diskColor.

u4(board) – Επιστρέφει μία τιμή ανάλογα με το πόσες κινήσεις έχει ο min παίκτης από το αρχικό board state μέχρι το μέγιστο βάθος, όσες πιο πολλές κινήσεις έχει – τόσο μικρότερη τιμή επιστρέφει.

u5(board, diskColor) – Εάν το board βρίσκεται σε terminal state, επιστρέφει μία πολύ μεγάλη τιμή εάν ο παίκτης με χρώμα diskColor κερδίζει, επιστρέφει μία πολύ μικρή τιμή εάν ο παίκτης με το άλλο χρώμα κερδίζει, διαφορετικά επιστρέφει 0.

- main.py – Κύρια Συνάρτηση

Παραδείγματα Χρήσης:

Τα παραδείγματα χρήσης βρίσκονται μέσα στον φάκελο **games**.

Πειραματικά Αποτελέσματα:

Τα πειραματικά αποτελέσματα βρίσκονται μέσα στον φάκελο **genetic_weights**.

Πηγές:

[1] http://www.cs.cmu.edu/~nasmith/papers/smith+tromble.tr04.pdf