ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ 2021-2022

ΓΕΩΡΓΙΟΣ ΚΥΡΙΟΠΟΥΛΟΣ 2019030017

ΓΙΑΝΝΗ ΓΙΟΕΛ 2019030116

ΕΙΣΑΓΩΓΗ

Στα πλαίσια του μαθήματος ζητήθηκε η σχεδίαση και υλοποίηση ενός πράκτορα όπου παίζει το παιχνίδι TUC CHESS. Πιο συγκεκριμένα σε πρώτο στάδιο ζητήθηκε η υλοποίηση του αλγορίθμου minimax μαζί με επεκτάσεις οι οποίες αποσκοπούν στην βελτίωση της αποδοτικότητάς της, ενώ στην συνέχεια ζητήθηκε η υλοποίηση του αλγόριθμού Monte Carlo.

ΜΕΡΟΣ Α

Σε αυτό το στάδιο υλοποιήθηκε ο αλγόριθμος minimax. Η κύρια ιδέα είναι αυτή που αναφέρεται στο σύγγραμμα του μαθήματος όμως είναι προσαρμοσμένος στις ανάγκες του συγκεκριμένου προβλήματος. Πιο συγκεκριμένα, στην δικιά μας υλοποίηση ο αλγόριθμος δεν φτιάχνει το δέντρο αναζήτησης και έπειτα το αναζητεί, αυτό που κάνει είναι ότι μέσω αναδρομικών κλήσεων επεκτείνει τα nodes μέχρι το cut-off από κάτω προς τα πάνω ενώ ταυτόχρονα υπολογίζει ένα evaluation για κάθε παραλλαγή της παρτίδας. Στο τέλος επιστρέφει την κίνηση η οποία ενδέχεται να οδηγήσει στην παραλλαγή με το καλύτερο δυνατό αποτέλεσμα για τον παίκτη.

Για να επιτύχουμε τους παραπάνω στόχους υλοποιήθηκαν μερικές βοηθητικές συναρτήσεις η πιο σημαντική εξ' αυτών ήταν η findScore η οποία βρίσκει για κάθε κίνηση η οποία πρόκειται να παιχτεί το πόσο αυτή θα επηρεάσει το τελικό score του εκάστοτε παίχτη.

Η findScore δεν δίνει βάρος μόνο όταν επρόκειτο να γίνει capture ενός κομματιού αλλά και όταν γίνει promote ή πιαστεί κάποιο δώρο. Πιο συγκεκριμένα όταν αιχμαλωτίζονται πιόνια ή πύργοι ή ο βασιλιάς προσθέτει στο score +1 ή +3 ή +8 αντίστοιχα. Όταν ένα πιόνι πρόκειται να βρεθεί σε θέση για promote προσθέτει +1 ενώ όταν κάποιο κομμάτι πρόκειται να πάρει ένα από τα δώρα στην σκακιέρα προστίθεται στο score +1 με πιθανότητα 9/10. Για να υλοποιηθεί ο παράγοντας της τύχης γράφτηκε μια συνάρτηση παραγωγής τυχαίων αριθμών.

Η συνάρτηση evaluation όπου χρησιμοποιήθηκε ήταν αυτή που περιγράφεται και στην εκφώνηση της άσκηση δηλαδή: Evalution = |whitePieces + whiteScore| - |blackPieces + blackScore|

Ο πράκτορας δηλαδή επιχειρεί να οδηγηθεί σε μια κατάσταση όπου μεγιστοποιείται η συνάρτηση evaluation στην περίπτωση που έχει τα λευκά η ελαχιστοποιείται σε περίπτωση που έχει τα μαύρα.

Για την περεταίρω βελτίωση της ταχύτητας αναζήτησης εφαρμόστηκε ο αλγόριθμος a-b pruning. Ουσιαστικά αυτός αποσκοπεί στο «κλάδεμα» του δένδρου στις περιπτώσεις όπου ο πράκτορας χάνει έτσι ώστε να μειωθεί ο χώρος αναζήτησης και να βελτιωθεί ο τελικός χρόνος εκτέλεσης.

ΜΕΡΟΣ Β

Στο δεύτερο μέρος υλοποιήθηκε ο αλγόριθμος Monte Carlo, η υλοποίηση του έγινε όπως περιγράφεται στην θεωρία σε 4 δηλαδή στάδια: Selection , Expansion, Simulation, Back-Propagation. Συγκεκριμένα ακολουθώντας την δομή της παρακάτω σελίδας:

https://www.baeldung.com/java-monte-carlo-tree-search

Υλοποιήθηκαν οι συναρτήσεις findNextMove, η οποία είναι η κεντρική συνάρτηση του MCTS και επιστρέφει την επόμενη κίνηση, η selectPromisingNode , όπου ανάλογα την UCT τιμή επιλέγουμε το αντίστοιχο node και το επιστρέφουμε , η expandNode , η οποία κάνει expand το node της selectPromisingNode και το προσθέτει στο array με τα παιδιά του node που έχει υλοποιηθεί στην κλάση Node, η simulateRandomPlayout , προσομοιώνει

το παιχνίδι από ένα τυχαίο node που επιλέχθηκε και επιστρέφει ποιος κέρδισε και τέλος η backpropagation όπου γυρνάει προς το rootNode και σταδιακά αλλάζει τα rewards των Nodes και το πόσες φορές επισκέφθηκαν . Εκτός από τις βασικές συναρτήσεις, έχουν υλοποιηθεί και δύο βοηθητικές συναρτήσεις η getMoves() και makeMove1(). Να αναφερθεί ότι ο κώδικας που υλοποιήθηκε για το μέρος του MonteCarlo δεν είναι λειτουργικός καθώς δεν λύθηκαν όλα τα προβλήματα που προέκυψαν.

Καθώς δεν μπορούμε να συγκρίνουμε τις δύο μεθόδους μεταξύ τους για λόγους που προαναφέρθηκαν, θεωρητικά όμως αναμένουμε ότι σε προβλήματα με μεγάλο branching factor, ο αλγόριθμος Monte Carlo είναι πιο αποτελεσματικός από των minimax διότι υπολογίζει όλο το game tree σε αντίθεση με τον Monte Carlo ο οποίος επιλέγει promising Nodes και στη συνέχεια προσομοιώνει το παιχνίδι , χάνωντας ίσως λίγο σε accuracy λόγω τις τυχαιότητας.

Πηγές:

https://www.baeldung.com/java-monte-carlo-tree-search

https://ai-boson.github.io/mcts/

https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/

http://www.incompleteideas.net/609%20dropbox/other%20readings%20and%20resources/MCTS-survey.pdf

https://www.baeldung.com/java-minimax-algorithm

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

https://www.youtube.com/watch?v=-ivz8yJ4l4E

https://www.youtube.com/watch?v=l-hh51ncgDI