Εργασιά Προγραμματισμού Τεχνητή Νοημοσύνη

Κωνσταντίνος Νικολός
Αριθμός Μητρώου: 2019030096

8 Φεβρουαρίου 2025

Περίληψη

Αναφορά της προγραμματιστικής εργασίας του μαθήματος Τεχνητής Νοημοσύνης.

1 Εισαγωγή

Η συγκεκριμένη εργασία αφορά την παραγωγή και χρήση αλγορίθμων αναζήτησης στο περιβάλλον παιχνιδιού $HexThello$. Πρόκειται για ένα παιχνίδι δύο παικτών όπου κάθε παίκτης καλείται να πάρει μία απόφαση στη σειρά του έχοντας ως δεδομένο την τωρινή κατάσταση του παιχνιδιού (ταμπλό). Ειδκότερα χρησιμοποιείται ο αλγόιρθμος $MiniMax$ κατάλληλος για το συγκεκριμένο πρόβλημα με αντίπαλο και ορισμένες παραλλαγές του για την επιλογή επομένης κίνησης όποτε κληθεί από τον παίκτη.

2 Δομή του Κώδικα

2.1 Δένδρο Αναζήτησης

Για να αναπαραστήσουμε το δέντρο αναζήτησης, δημιουργήσαμε μια δομή $treeNode$, η οποία περιέχει όλη την πληροφορία μίας κατάστασης του παιχνιδιού:

- Τρέχουσα θέση του παιχνιδιού στο ταμπλό ($position$).
- Τελευταία κίνηση που οδήγησε στην κατάσταση αυτή ($lastMove$).
- Εκτιμομένη τιμή του κόμβου από την συνάρτηση αξιολόγησης ($valuation$).

- Τα παιδιά του κόμβου, δηλαδή τις επόμενες πιθανές καταστάσεις ($children$).
- Τον αριθμό των διαθέσιμων επόμενων καταστάσεων (παιδιών) ($childCount$).

2.2 Συνάρτηση Αξιολόγησης

Η συνάρτηση αξιολόγησης που χρησιμοποιείται είναι η απλούστερη δυνατή όπως προτάθηκε στην εκφώνηση της άσκησης.

$$V(state) = (\text{Αριθμός δίσκων μας}) - (\text{Αριθμός δίσκων αντιπάλου}) \quad (1)$$

2.3 Απλός $Minimax$

Ο απλός αλγόριθμος $Minimax$ υλοποιήθηκε σύμφωνα με τις διαλέξεις του μαθήματος. Η συνάρτηση $simpleMinimax$ εφαρμόζει αναδρομικά την αναζήτηση στο δέντρο και επιστρέφει την καλύτερη εκτίμηση για μια δεδομένη θέση.

2.4 $Alpha - BetaPruning$

Η μέθοδος $alphaBetaMinimax$ βελτιώνει τον $Minimax$ χρησιμοποιώντας $Alpha - Beta\ pruning$, διατηρώντας δύο παραμέτρους ($alpha$ και $beta$), οι οποίες επιτρέπουν το *κλάδεμα* μη αναγκαίων κόμβων του δέντρου. Ο σκοπός της συγκεκριμένης παραλλαγής είναι η ταχύτερη εύρεση λύσης καθώς υπάρχει περίπτωση να επισκεφθεί λιγότερους κόμβους χωρίς τον κίνδυνο αποκοπής της βέλτιστης λύσης.

2.5 $Alpha - BetaPruningwithMoveOrdering$

Πρόκειται για βελτίωση του αλγορίθμου $alphaBetaMinimaxWithOrdering$ αλλά δεν εφαρμόζει φορωαρδ πρυνινγ. Επί της ουσίας γίνεται κλάδεμα $Alpha - Beta\ pruning$ αφού έχει εφαρμοστεί ταξινόμηση των παιδιών του κόμβού ώστε να γίνεται ταχύτερα και σε μεγαλύτερο βαθμό το κλάδεμα. Όπως και ο προηγούμενος αλγόριθμος δεν κινδυνεύει να αποκόψει τη λύση και θα έβγαζε την ίδια λύση με τον απλό $MiniMax$ αλγόριθμο.

3 Λειτουργία του Κώδικα

Ο χρήστης καλείται με το $option - a$ να δηλώσει ποιον αλγόριθμο θέλει να χρησιμοποιήσει. Όταν έρθει η σειρά του παίκτη, θα γίνει έλεγχος εάν υπάρχει διαθέσιμη κίνηση. Αν δεν υπάρχει, ο αλγόριθμος επιστρέφει την κίνηση $NULL$,

διαφορετικά προχωράει στη δημιουργία του κόμβου ρίζας, δηλαδή της τελευταίας κατάστασης που δόθηκε από τον σερερ, και στην κλήση του κατάλληλου αλγορίθμου.

Κάθε αλγόριθμος ελέγχει τις διαθέσιμες επόμενες καταστάσεις, δημιουργεί τα αντίστοιχα παιδιά και καλεί τον εαυτό του αναδρομικά μέχρι να φτάσει στον κόμβο με την καλύτερη αξιολόγηση *valuation*.

Ο αλγόριθμος επιστρέφει την καλύτερη τιμή που υπολογίστηκε και ο παίκτης αναζητά το αντίστοιχο παιδί (κατάσταση) της ρίζας με τη συγκεκριμένη τιμή. Χρησιμοποιώντας τη μεταβλητή *last_move* του παιδιού βρίσκεται και η αντίστοιχη κίνηση που θα οδηγήσει στον κόμβο με το υψηλότερο *valuation* που υπολόγισε ο αλγόριθμος.

Όλες οι προσθήκες κώδικα έλαβαν μέρος στο αρχείο *client.c* και δεν τροποποιήθηκε κανένα αρχείο επικοινωνίας με το *server* του παιχνιδιού.

4 Ανάλυση Αποτελεσμάτων

Σύμφωνα με τις δοκιμές, οι τρεις αλγόριθμοι αποδίδουν διαφορετικά όσον αφορά το χρόνο εκτέλεσης όπως αναμενόταν. Πιο συγκεκριμένα ο απλός αλγόριθμος $MiniMax$ παρουσιάζεται πιο αργός ενώ υπάρχει σημαντική βελτίωση για τον $Alpha-Beta\ pruning$ και ακόμα πιο σπουδαία στην περίπτωση $preordering$

Δοκιμές με $MAX\_DEPTH = 3$, έδειξαν ότι ο αλγόριθμος $Alpha-BetaPruning$ ήταν περίπου **50% ταχύτερος** από το $SimpleMinimax$ ενώ ο $Alpha-BetameMoveOrdering$ αποδείχθηκε **100% ταχύτερος** από τον $SimpleMinimax$.

Προφανώς όσο το βάθος αυξάνεται τόσο αυξάνεται και η χρονική πολυπλοκότητα με τις διαφορές για $MAX\_DEPTH = 4$, να εξελίσσονται στον $Alpha-Beta$ **κατά 5 φορές ταχύτερο του απλού** $MiniMax$, ενώ ο $Alpha-Beta$ με $MoveOrdering$ είναι **10 φορές ταχύτερος** .

5 Συμπεράσματα

Συμπερασματικά, η χρήση του $Alpha-Beta\ Pruning$ μειώνει δραστικά τον χρόνο εκτέλεσης, χωρίς να επηρεάζει τη βέλτιστη επιλογή κίνησης ενώ η προσθήκη $Ordering$ αυξάνει περαιτέρω την αποδοτικότητα του $pruning$, προσφέροντας έναν ταχύτερο παίκτη για το $HexThello$ που σε περιπτώσεις περιορίσμου του χρόνου απόφασης θα ήταν αποδοτικότερος.

6 Οδηγίες Εκτέλεσης

- $./client - i127.0.0.1 - p6002 - a0$: *Simple MiniMax*
- $./client - i127.0.0.1 - p6002 - a1$: $A - B$ *pruning*
- $./client - i127.0.0.1 - p6002 - a1$: $A - B$ *pruning with ordering*