ΠΛΗ417 – Τεχνητή Νοημοσύνη
2η Προγραμματιστική Εργασία

LAB41744570
Μανώλης Πετράκος ΑΜ 2014030009

Για την υλοποίηση της εργασίας έχουν δημιουργηθεί δύο νέα αρχεία (state.c, state.h) στα οποία υλοποιείται ο πράκτορας. Επίσης, έχουν γίνει αλλαγές στα αρχεία board.c και board.h ώστε η δομή Position να περιέχει περισσότερη πληροφορία. Τέλος, έχουν γίνει αλλαγές στα αρχεία client.c και client.h ώστε να εμφανίζονται στατιστικά και επιδόσεις του πράκτορα.

Επεξήγηση προσέγγισης

Το πρώτο βήμα για την κατασκευή του πράκτορα είναι να καθοριστεί μια δομή ως κατάσταση του πράκτορα και κόμβος του δέντρου αναζήτησης. Δεν έχει δημιουργηθεί κάποια καινούργια δομή, αλλά χρησιμοποιείται η *Position*. Τα δεδομένα της είναι αρκετά για την λειτουργία του minimax αλγόριθμου, αλλά έχει επεκταθεί για να μειωθεί ο υπολογιστικός φόρτος και για καλύτερη διάκριση μεταξύ διαφορετικών καταστάσεων. Πιο συγκεκριμένα, θυμάται για κάθε πλευρά:

- πόσα μυρμήγκια ζουν,
- πόσα μυρμήγκια έχουν πεθάνει σε σχέση με την ρίζα του δέντρου,
- πόσα φαγητά έχουν πάρει σε σχέση με την ρίζα του δέντρου.

Για την ενημέρωση των παραπάνω, δημιουργήθηκε η συνάρτηση *DoMove2*. Αυτή, διαφέρει από την έτοιμη *DoMove* και στον υπολογισμό του σκορ, δίνοντας πάντα πόντο όταν τρώγεται ένα φαγητό. Τότε, το σκορ του *Position* περιγράφει το μέγιστο δυνατό σκορ σε αυτήν τη κατάσταση. Με την χρήση των παραπάνω στατιστικών είναι δυνατόν να ξεχωριστούν οι σίγουροι πόντοι (από μυρμήγκια-βασίλισσες και από την αρχή του δέντρου) και οι πιθανοί πόντοι (από φαγητά).

Για την επιλογή των κινήσεων είναι απαραίτητη μια συνάρτηση καταλληλότητας. Λαμβάνονται υπόψιν οι παρακάτω παράμετροι και για τους δύο παίχτες:

- Ζωντανά μυρμήγκια με βάρος 15,
- σίγουροι πόντοι με βάρος 10,
- πιθανοί πόντοι με βάρος 5,
- δυνατότητα αιχμαλώτισης από τον επόμενο παίχτη με βάρος 10. Αν ο επόμενος παίχτης είναι ο αντίπαλος του πράκτορα, το βάρος γίνεται -10.

Δίνεται μεγαλύτερη βαρύτητα στα ζωντανά μυρμήγκια, καθώς αυτά μπορούν να παράξουν περισσότερους πόντους στο μέλλον και να εμποδίσουν τον αντίπαλο. Αν οι πόντοι είχαν ίσο ή μεγαλύτερο βάρος από τα ζωντανά μυρμήγκια, ο πράκτορας γινόταν greedy και προσπαθούσε να μαζέψει γρήγορα όσους περισσότερους πόντους μπορούσε, βάζοντας τον εαυτό σε δυσκολότερη θέση στο μέλλον. Τέλος, η τελευταία παράμετρος υπάρχει για να μειωθεί το horizon effect[1] που εμφανίζεται από την ατελής αναζήτηση. Συγκεντρωτικά, ο υπολογισμός της αξίας μιας κατάστασης γίνεται ως εξής:

$$value = (\mu\upsilon\rho\mu\acute{\eta}\gamma\kappa\iota\alpha_{\pi\rho\acute{\alpha}\kappa\tau o\rho\alpha} - \mu\upsilon\rho\mu\acute{\eta}\kappa\iota\alpha_{\alpha\nu\tau\acute{\iota}\pi\alpha\lambda o\upsilon}) * 15 + (\pi\acute{o}\nu\tau o\iota^{\sigma\acute{\iota}\gamma o\upsilon\rho o\iota}_{\pi\rho\acute{\alpha}\kappa\tau o\rho\alpha} - \pi\acute{o}\nu\tau o\iota^{\sigma\acute{\iota}\gamma o\upsilon\rho o\iota}_{\alpha\nu\tau\acute{\iota}\pi\alpha\lambda o\upsilon}) * 10 + (\pi\acute{o}\nu\tau o\iota^{\pi\iota\theta\alpha\nu o\acute{\iota}}_{\pi\rho\acute{\alpha}\kappa\tau o\rho\alpha} - \pi\acute{o}\nu\tau o\iota^{\pi\iota\theta\alpha\nu o\acute{\iota}}_{\alpha\nu\tau\acute{\iota}\pi\alpha\lambda o\upsilon}) * 5 + \alpha\iota\chi\mu\alpha\lambda\acute{\omega}\tau\iota\sigma\eta_{\epsilon\pi\acute{o}\mu\epsilon\nu o\varsigma\ \gamma\acute{\upsilon}\rho o\varsigma} * \pm 10$$

Η συνάρτηση καταλληλότητας δεν υπολογίζεται για κάθε κόμβο, αλλά μόνο για τους τερματικούς, οι οποίοι επιλέγονται μέσω της συνάρτησης τερματισμού. Ο βασικότερος παράγοντας της είναι το μέγιστο βάθος του δέντρου, η συνάρτηση σταματάει όλους τους κόμβους που θα το φτάσουν. Πέρα από αυτό, μπορεί να θεωρήσει έναν κόμβο τερματικό για τους παρακάτω λόγους:

- Είναι αδύνατον να γίνει αιχμαλωσία ή να πάρει κάποιος φαγητό τώρα ή στο μέλλον.
- Ο αντίπαλος δεν έχει μυρμηγκιά και δεν υπάρχει διαθέσιμο φαγητό.
- Ο πράκτορας χάνει και είναι αδύνατον να το αλλάξει.
- Ο πράκτορας κερδίζει και είναι αδύνατον να αλλάξει.

Σε γενικές γραμμές, τα παραπάνω κριτήρια στοχεύουν σε καταστάσεις όπου δεν μπορεί να γίνει κάποια ουσιαστική αλλαγή σε αυτές ή σε κάποια απόγονό τους. Τέλος, υπάρχει η επιλογή ο πράκτορας να μην σταματάει σε ‘άστατες’ καταστάσεις, δηλαδή σε αυτές όπου μπορεί να γίνει αιχμαλωσία. Τότε, τα φύλλα του δέντρου συνεχίζουν και πέρα του ορίου βάθους, μέχρι να φτάσουν σε κατάσταση ηρεμίας ή σε ένα δεύτερο όριο βάθους. Σαν αποτέλεσμα, μειώνεται το horizon effect αλλά αυξάνεται ο χρόνος εκτέλεσης.

Με τις παραπάνω δομές και συναρτήσεις μπορεί να υλοποιηθεί ο αλγόριθμος minimax. Αυτό γίνεται στις συναρτήσεις *minimax_decision*, *min_value* και *max_value*. Η αναδρομή ξεκινάει από την πρώτη και καλούνται οι άλλες δύο εναλλάξ. Για κάθε νέα κατάσταση μελετούνται όλοι οι δυνατοί απόγονοι, μέχρι να σταματήσουν από την συνάρτηση τερματισμού. Παρατηρείται ότι το δέντρο αναζήτησης γίνεται πολύ μεγάλο με αποτέλεσμα να μην είναι δυνατόν βαθιά αναζήτηση σε λογικά χρονικά πλαίσια.

Για να μπορέσει ο πράκτορας να εμβαθύνει την αναζήτηση, ο αλγόριθμος επεκτείνεται με a-b pruning, ώστε να μην μελετούνται όλοι οι κόμβοι. Μετά από πειραματισμούς, φαίνεται πως το pruning αποδίδει καλύτερα αν, και για τις δύο μεριές, μελετούνται πρώτες οι πιο προωθημένες κινήσεις. Για αυτό, η αναζήτηση των δυνατών κινήσεων γίνεται με διαφορετικό τρόπο ανάλογα το χρώμα του παίχτη. Πιο συγκεκριμένα, υπάρχουν η συνάρτηση *find_moves_white* όπου μελετάει το board από πάνω προς τα κάτω και η συνάρτηση *find_moves_black* όπου μελετάει το board από κάτω προς τα πάνω. Αφού βρεθούν όλες οι κινήσεις, αυτές δοκιμάζονται ανάποδα.

1 Επειδή γίνεται αναζήτηση σε ένα φραγμένο δέντρο καταστάσεων, ο πράκτορας δεν γνωρίζει τι υπάρχει πέρα από αυτό. Τότε εμφανίζεται το horizon effect και οι επιλογές του μπορεί να μην είναι οι βέλτιστες.

Το pruning μπορεί να βελτιωθεί περαιτέρω εάν μελετούνται πρώτα οι καταστάσεις που είναι πιθανότερο να φτάνουν σε καλύτερα αποτελέσματα. Για αυτό, υλοποιείται η συνάρτηση *do_moves_and_reorder* η οποία δημιουργεί τους κόμβους-παιδιά μιας κατάστασης και τους ταξινομεί βάση μιας ευρετικής συνάρτησης. Δεν χρησιμοποιείται η συνάρτηση καταλληλότητας καθώς είναι υπερβολικά υπολογιστικά ακριβό να καλείται για κάθε κόμβο. Επίσης, η χρήση της είναι πλεονάζων γιατί συγκρίνονται κόμβοι με το ίδιο ιστορικό και οτιδήποτε έχει γίνει πριν από αυτούς, είναι κοινό σε όλους. Η ευρετική συνάρτηση είναι μια απλοποιημένη παραλλαγή της και φαίνεται παρακάτω.

$$heurestic = νεκρά_{άλλου} * 2 + πόντοι_{παίχτη}^{πιθανοί} - αιχμαλώτιση_{επόμενος\,γύρος} * 2$$

Ουσιαστικά περιγράφει την μεταβολή που προκάλεσε η κίνηση, για αυτό περιέχει μόνο τις μεταβλητές που μπορούν να αλλάξουν στον αντίστοιχο γύρο. Σε αντίθεση με την συνάρτηση καταλληλότητας, δείχνει προτίμηση του παίχτη που την έκανε, και όχι του πράκτορα. Για αυτό, και οι δύο αναδρομικές συναρτήσεις μελετάνε τις νέες καταστάσεις από την μεγαλύτερη τιμή προς την μικρότερη.

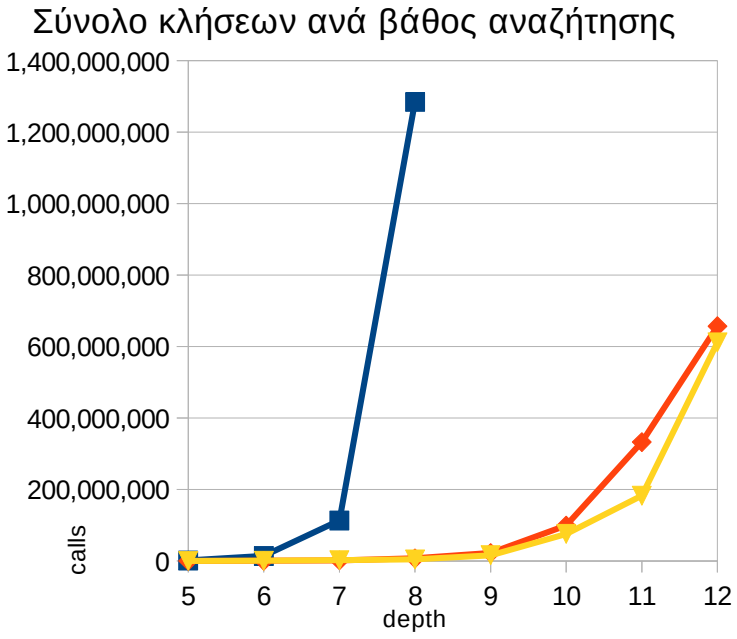
Ο χρόνος εκτέλεσης του αλγορίθμου μπορεί να μειωθεί παραλληλίζοντας την αναζήτηση. Αυτό επιτυγχάνεται χρησιμοποιώντας ένα νήμα για την μελέτη κάθε κίνησης που μπορεί να κάνει ο πράκτορας. Αφού δεν υπάρχει μεταφορά πληροφορίας μεταξύ των καταστάσεων που δημιουργούνται στην συνάρτηση *minimax_decision*, δεν υπάρχουν race conditions και η υλοποίηση είναι αρκετά απλή. Έτσι ο χρόνος εκτέλεσης μπορεί να μειωθεί μέχρι τον χρόνο μελέτης της κίνησης με το μεγαλύτερο υποδέντρο.

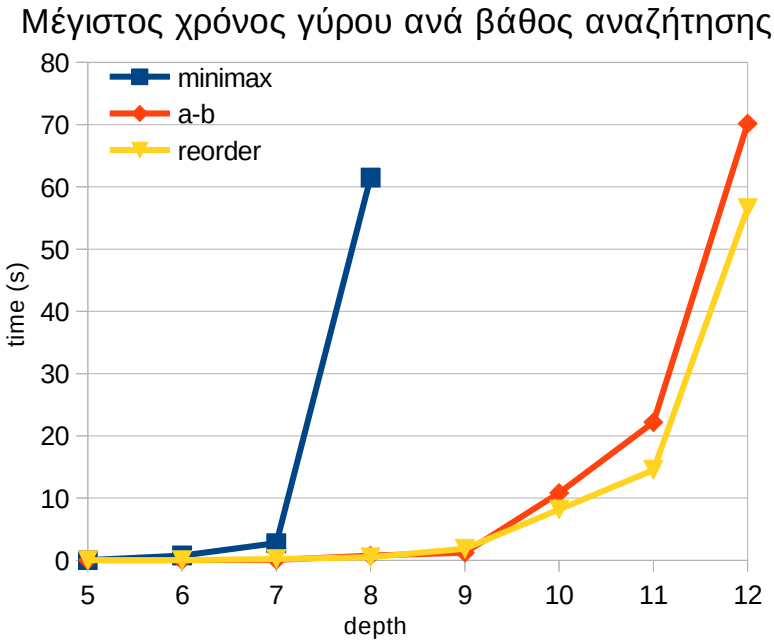
Γενικά, η υλοποίηση του πράκτορα είναι τέτοια ώστε να ελαχιστοποιούνται οι κλήσεις συναρτήσεων καθώς και οι απαραίτητοι υπολογισμοί, με κόστος μεγαλύτερη ανάγκη μνήμης. Όσο πιο γρήγορη είναι η αναζήτηση, τόσο πιο βαθιά μπορεί να γίνει, με αποτέλεσμα ο πράκτορας να παίρνει καλύτερες αποφάσεις. Τέλος, η μείωση των αναδρομών δεν επιφέρει αυτόματα βελτίωση απόδοσης, καθώς οι συναρτήσεις μπορούν να έχουν μεγαλύτερες υπολογιστικές ανάγκες.

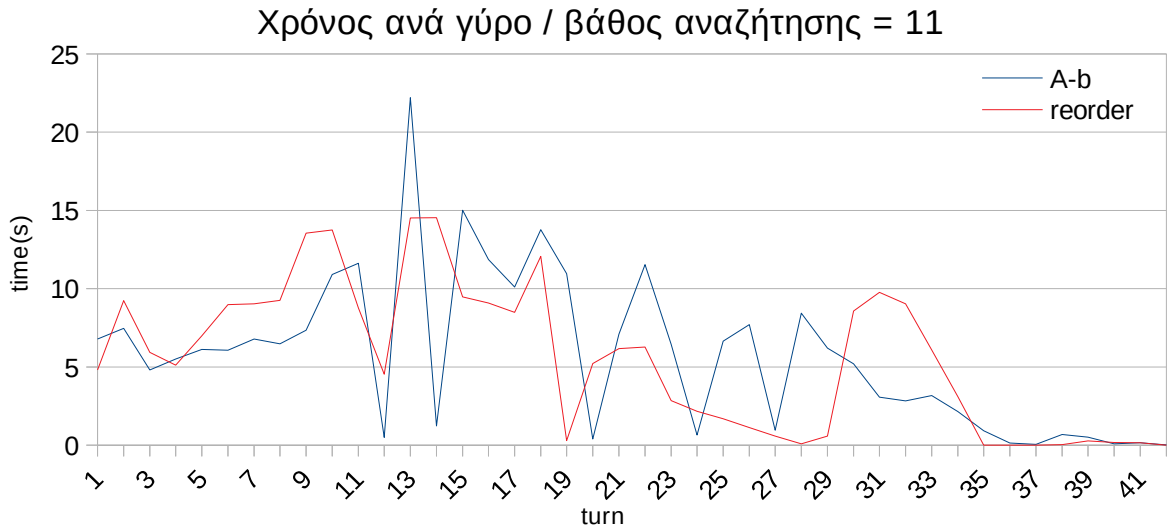
Πειραματική Διαδικασία

Η απόδοση του πράκτορα αποτελείται από δύο παραμέτρους και πρέπει να ποσοτικοποιηθούν για να μελετηθεί σωστά. Η πρώτη είναι πόσο καλές αποφάσεις λαμβάνει. Γενικά, όσο βαθύτερη είναι η αναζήτηση, τόσο καλύτερες θα είναι η αποφάσεις. Άρα η πρώτη παράμετρος μπορεί να αναπαρασταθεί από το βάθος της αναζήτησης. Η δεύτερη παράμετρος είναι η ταχύτητα λήψης αποφάσεων και μπορεί να ποσοτικοποιηθεί ως ο μέγιστος χρόνος λήψης μιας. Όμως, αυτό το μέγεθος επηρεάζεται και από άλλους εξωτερικούς παράγοντες, όπως το μηχάνημα στο οποίο τρέχει ο πράκτορας. Για αυτό, μαζί με χρόνο λήψης μιας απόφασης, παρουσιάζεται και ο συνολικός αριθμός των κλήσεων των αναδρομικών συναρτήσεων του αλγόριθμου, κάτι που επηρεάζεται μόνο από τον πράκτορα και το παιχνίδι.

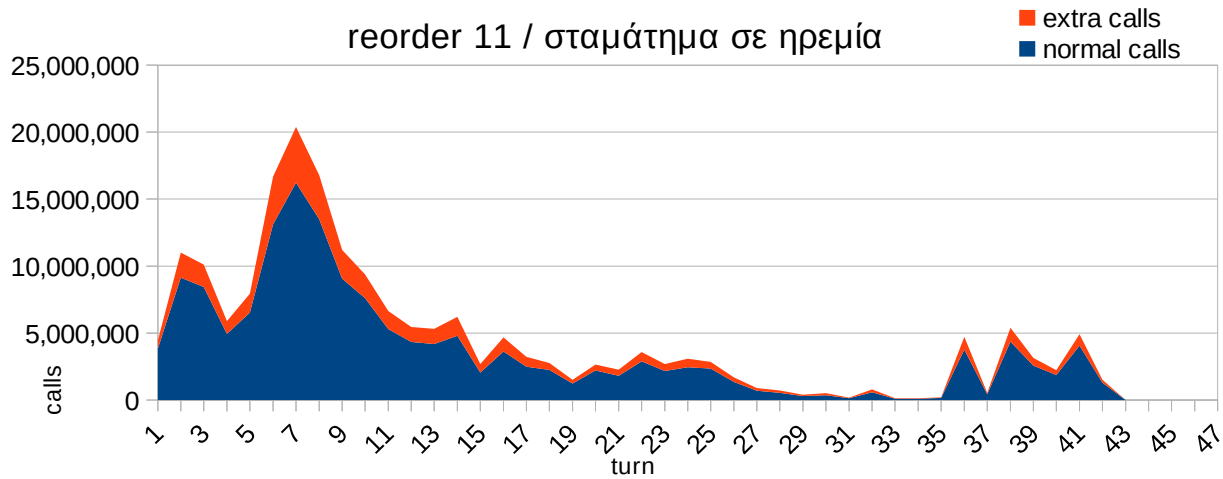
Οι μετρήσεις έγιναν σε παιχνίδια εναντίον του τυχαίου πράκτορα. Για κάθε συνδυασμό βάθους και αλγόριθμου έγιναν μερικές μετρήσεις και κρατήθηκαν τα πιο αντιπροσωπευτικά αποτελέσματα. Δεν χρησιμοποιήθηκαν μέσοι όροι καθώς εξομαλύνουν τα spikes, τα οποία είναι σημαντικό να φανούν.



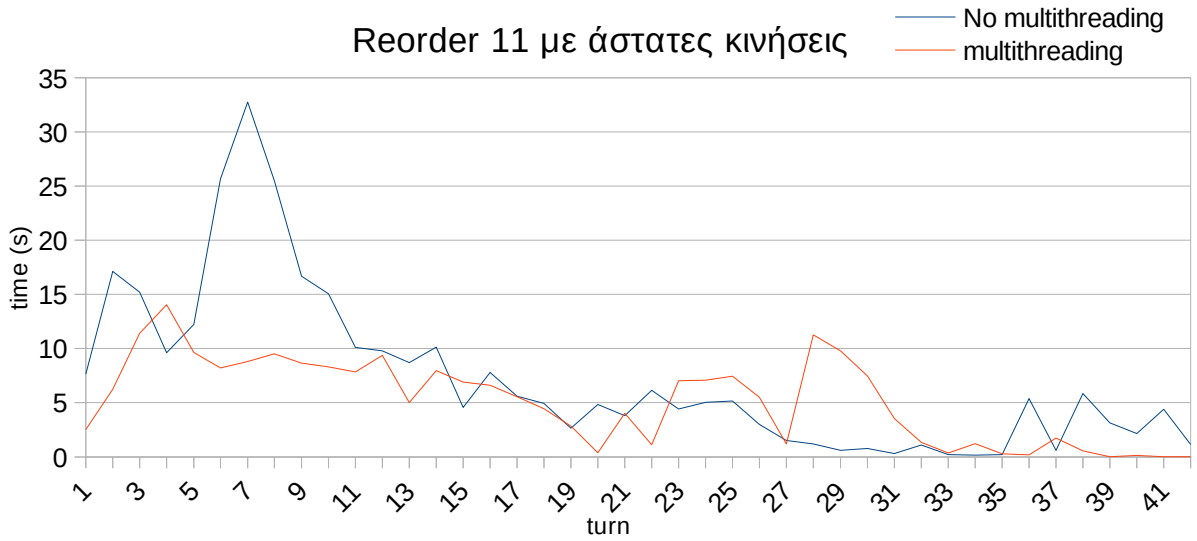




Στα πρώτα δυο διαγράμματα φαίνεται η εξέλιξη της επίδοσης των αλγόριθμων ανάλογα με το βάθος της αναζήτησης. Στο τρίτο παρουσιάζεται ο χρόνος εκτέλεσης κάθε γύρου για το βέλτιστο βάθος. Τα παιχνίδια συνεχίζονται για περίπου ακόμα 50 γύρους, αλλά ο πράκτορας έχει καταλάβει ότι δεν μπορεί να γίνει κάποια ουσιαστική αλλαγή και δεν γίνεται αναζήτηση.



Στο τέταρτο διάγραμμα φαίνονται οι παραπάνω κλήσεις που προκαλούνται από τις ‘άστατες’ καταστάσεις με όριο βάθους για τις extra κλήσεις ίσο με 16.



Στο πέμπτο διάγραμμα παρουσιάζεται η επίδραση του παραλληλισμού της αναζήτησης.

Παρατηρήσεις & Συμπεράσματα

- Από τα πρώτα δύο διαγράμματα φαίνεται ότι το α-β pruning βελτιώνει σημαντικά το βάθος αναζήτησης, κάτι που το καθιστά υποχρεωτικό για την βέλτιστη λειτουργία του πράκτορα.
- Το α-β pruning, εκ φύσεως του, δεν έχει σταθερό χρόνο αναζήτησης και δημιουργούνται υψηλά spikes στον χρόνο εκτέλεσης. Αυτό το φαινόμενο μπορεί να ελαττωθεί χρησιμοποιώντας το reorder, καθώς μειώνει την τυχαιότητα της σειράς εξερεύνησης των κόμβων. Αυτό το καθιστά σημαντικό, ακόμα και αν δεν δημιουργεί μικρή βελτίωση στον συνολικό αριθμό κλήσεων και τον μέσο χρόνο εκτέλεσης.
- Το reorder επηρεάζει την σειρά εξερεύνησης των κόμβων, με αποτέλεσμα οι ισοβάθμιες να επιλύονται διαφορετικά και ο πράκτορας να παίρνει διαφορετικές αποφάσεις. Αυτή η παρενέργεια δεν εμφανίζεται στο απλό α-β pruning.
- Εξαιτίας των κανόνων του παιχνιδιού, οι κινήσεις που μπορούν να γίνουν σε ένα γύρο δεν διαφέρουν πολύ μεταξύ τους, με αποτέλεσμα να είναι δύσκολο να κατασκευαστεί μια ευρετική που να τις ταξινομεί βέλτιστα.
- Στις μετρήσεις παρατηρήθηκε ότι 1εκ. κλήσεις διαρκούν στον minimax ~0.5 sec, στο α-β pruning ~0.9sec και στο reorder ~1.5sec. Αυτό καθιστά την ταξινόμηση στο reorder υποψήφιο σημείο για βελτιστοποίηση.
- Από το τέταρτο διάγραμμα φαίνεται ότι η αντιμετώπιση του horizon effect απαιτεί αύξηση των υπολογισμών. Το παραπάνω κόστος είναι ανεκτό και η αλλαγή κρατήθηκε.
- Ο παραλληλισμός αποδίδει, αλλά εξαρτάται από το μηχάνημα που χρησιμοποιείται. Οι μετρήσεις έγιναν σε ένα διπύρηνο επεξεργαστή άρα το μέγιστο speedup που μπορούν να εμφανίσουν είναι 2. Μπορεί να βελτιωθεί περαιτέρω με την χρήση ενός πιο έξυπνου work scheduling/stealing scheme.
- Ο συνδυασμός που επιλέχθηκε ως βέλτιστος είναι α-β / reorder / σταμάτημα σε ηρεμία / multi threading / βάθος 11.

Μελλοντικές Βελτιώσεις

- Το επόμενο βήμα στην ανάπτυξη του πράκτορα είναι η χρήση transposition table.
- Βελτιστοποίηση ταξινόμησης του reorder.

Πηγές

Minimax, a-b pruning

Move Ordering

Horizon Effect

Parallel Search