ΠΛΗ 417 – ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

ΧΑΙΤΙΔΗΣ ΣΑΒΒΑΣ	2014030014
ΜΠΟΛΙΩΤΗΣ ΜΑΝΟΥΣΟΣ	2014030030
ΜΙΧΑΗΛ ΑΛΕΞΑΝΔΡΟΣ	2014030077
ΤΟΡΑΚΗΣ ΙΩΑΝΝΗΣ	2014030205

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΑΣΚΗΣΗΣ

Η άσκηση που μας ανατέθηκε είναι ίσως το πιο κλασσικό πρόβλημα τεχνητής νοημοσύνης, το πρόβλημα των n βασιλισσών. Το τελευταίο ορίζεται ως την τοποθέτηση n βασιλισσών σε μία σκακιέρα, σε θέσεις τέτοιες ώστε να απειλούνται το δυνατόν λιγότερες βασίλισσες. Υπάρχουν πολλές στρατηγικές λύσεις οι οποίες αντιμετωπίζουν το συγκεκριμένο πρόβλημα,άλλες κουνώντας βασίλισσες και άλλες τοποθετώντας κατάλληλα τις βασίλισσες εξαρχής. Οι δικοί μας αλγόριθμοι υλοποίησης ακολουθούν την πρώτη στρατηγική.

ΥΛΟΠΟΙΗΣΗ

Καθώς γνωρίζουμε ότι η πλειοψηφία των λύσεων θα γίνει σε περιβάλλον java/python, αποφασίσαμε να πρωτοτυπίσουμε και να δουλέψουμε σε μία low level γλώσσα, την C. Ακολουθούν οι απαντήσεις στα ζητούμενα καθώς και τα αποτελέσματα και τα εξαγόμενα συμπεράσματα.

ΜΕΡΟΣ Α

Ζητούμενο 1

Ξεκινώντας, υλοποιώντας το ζητούμενο 1, φτιάξαμε έναν πίνακα n x n, ο οποίος δημιουργείται δυναμικά παίρνοντας ως είσοδο το μέγεθος n, και παίζει τον ρόλο της σκακιέρας μας. Έπειτα τοποθετήσαμε n βασίλισσες σε τυχαίες θέσεις και υπολογίσαμε ποιες βρίσκονται υπό απειλή. Η υλοποίηση του τελευταίου έγινε μέσω μίας συνάρτησης, η οποία παίρνει σαν όρισμα τον δισδιάστατο πίνακα (σκακιέρα) και επιστρέφει τον αριθμό των απειλούμενων βασιλισσών. Η διαδικασία που ακολουθήσαμε για το τελευταίο, είναι η ακόλουθη. Διατρέχουμε σειριακά τον πίνακα μέχρι να βρούμε την πρώτη βασίλισσα. Εκεί σταματάμε και διασχίζουμε τις γραμμές,τις στήλες και τις διαγωνίους της τρέχουσας θέσης της βασίλισσας,αυξάνοντας σε κάθε περίπτωση έναν counter.

Ζητούμενο 2

Στη συνέχεια αναπτύξαμε τα μέτρα απόδοσης με τα οποία αξιολογείται ο πράκτορας μας. Ο συγκεκριμένος πράκτορας είναι ένας software agent, ο οποίος είναι utility based, καθώς είναι βασισμένος στην χρησιμότητα,ενεργώντας κάθε φορά ανάλογα με το πόσο απέχει από τον στόχο του, ακολουθώντας διάφορα paths. Τα μέτρα απόδοσης του συνοψίζονται στα παρακάτω.

Performance

• Να φτάσει με μία κατάσταση όπου απειλούνται το δυνατόν λιγότερες βασίλισσες.

• Να έχει χαμηλή χρονική πολυπλοκότητα, η οποία εξαρτάται από τα iterations που χρησιμοποιούμε.

• Να έχει τη βέλτιστη χωρική πολυπλοκότητα, δηλαδή να καταλαμβάνει λίγο χώρο στον δίσκο και να έχει μικρές απαιτήσεις ram.

Environment

Σαν περιβάλλον του πράκτορα ορίζεται η σκακιέρα, οι ήδη τοποθετημένες βασίλισσες, καθώς και οι θέσεις που βρίσκονται υπό απειλή.

Actuators

Οι επενεργητές του παρόντος software agent είναι η εγγραφή στον δίσκο , η πρόσβαση στην μνήμη ram και η διεπαφή με τον χρήστη, δηλαδή η εκτύπωση στην οθόνη.

Sensors

Οι αισθητήρες του συγκεκριμένου πράκτορα είναι το διάβασμα εισόδου από το πληκτρολόγιο και το ποντίκι, όπως και το διάβασμα δεδομένων από αρχεία.

ΜΕΡΟΣ Β

Ζητούμενο 1

Αποφασίσαμε να υλοποιήσουμε γεννητικό αλγόριθμο για το ζητούμενο της τοπικής αναζήτησης, καθώς είναι ένας αλγόριθμος ο οποίος βρίσκει αποτελέσματα για μεγάλα n σε εξαιρετικά μικρούς χρόνους. Η ιδέα του genetic algorithm είναι ότι υπάρχει μία μάσκα (επιθυμητό αποτέλεσμα), με την οποία συγκρίνεται η εκάστοτε τωρινή κατάσταση. Στόχος είναι να καταλήξουμε στην ίδια κατάσταση με την μάσκα και η διαδικασία που ακολουθείται είναι η εξής. Αρχικά,γίνεται τυχαία αναπαραγωγή διαδόχων, επιλέγεται δηλαδή μία τυχαία αρχική κατάσταση. Στην συνέχεια ,ανάλογα με την καταλληλότητα (fitness) με την μάσκα, επιβιώνουν οι καλύτεροι απόγονοι,αυτοί δηλαδή με το καλύτερο h(n). Ακολουθεί το crossover, όπου από τους δύο πλέον κατάλληλους γονείς δημιουργείται μία νέα κατάσταση. Την διαδικασία συμπληρώνει το mutation (μετάλλαξη), κατά την οποία αλλάζουμε τυχαία ένα bit της συμβολοσειράς. Πλέον έχουμε την νέα καταλληλότερη γενιά, η οποία θα περάσει από τα ίδια στάδια. Η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί λύση του προβλήματος, να καταλήξουμε ακριβώς στην μάσκα δηλαδή (optimum), στην οποία και εκτυπώνεται η σκακιέρα με q για τις θέσεις των βασιλισσών. Σε περίπτωση που υπερβούμε το όριο των 1000 γενιών, τότε σταματάμε, επιστρέφοντας την σκακιέρα με Q για τις βασίλισσες που βρίσκονται ακόμα υπό απειλή και q για τις ασφαλείς.

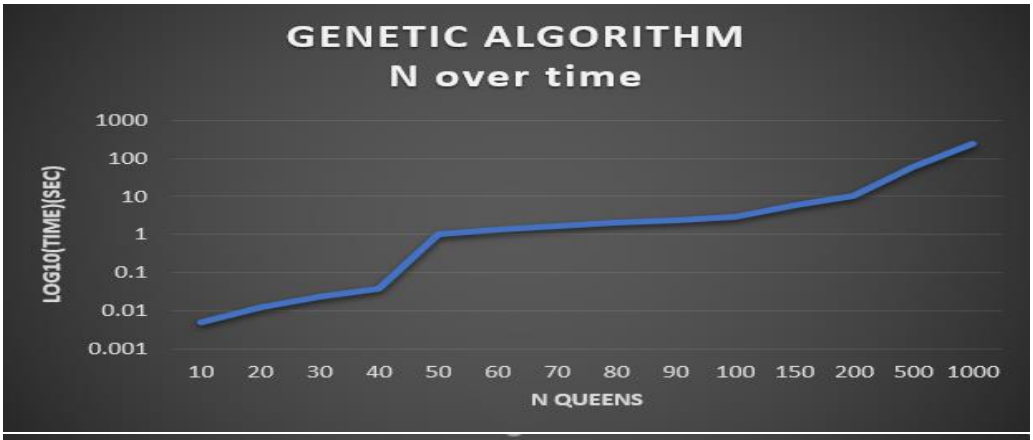
Ζητούμενο 2

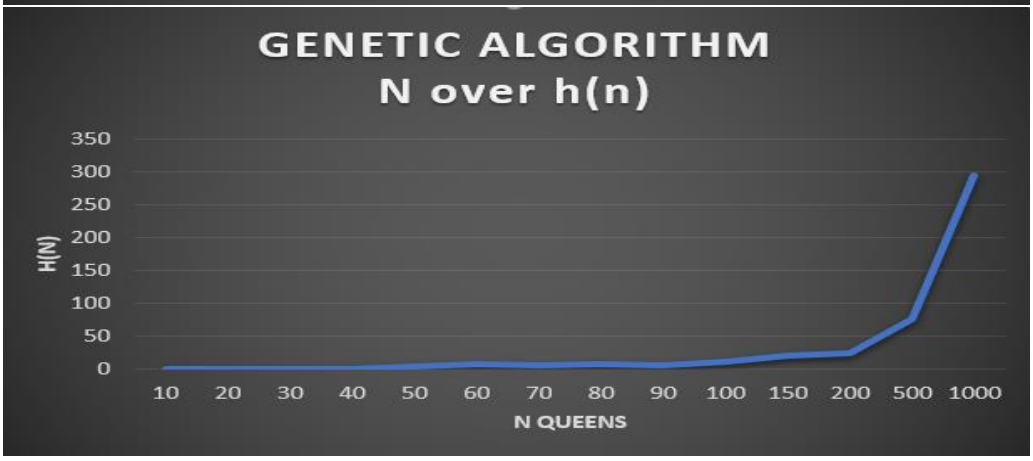
Επιλέξαμε να υλοποιήσουμε την online αναζήτηση με τον αλγόριθμο αναρρίχησης λόφων (online hill climbing), καθώς είναι ένας αλγόριθμος που βρίσκει καλές λύσεις σε πολύ μεγάλους/άπειρους χρόνους και απαιτεί λίγη μνήμη. Η ιδέα του online hill climbing είναι ότι ακολουθούμε τα βήματα ενός local hill climbing,έχοντας παράλληλα πληροφορία στην μνήμη. Αρχικά ξεκινάμε από μία random αρχικοποίηση της σκακιέρας,έχοντας τις βασίλισσες σε τυχαίες θέσεις, και βρίσκουμε αυτές που είναι υπό απειλή (με την συνάρτηση queens in danger). Στη συνέχεια ξεκινάμε από μία random τιμή και διατρέχουμε την σκακιέρα ψάχνοντας για queens in danger. Αν το h(n) που μας επιστρέφει η νέα κλήση και τοποθέτηση βασιλισσών είναι μικρότερο από το προηγούμενο,τότε μετακινούμε τις βασίλισσες στις νέες συντεταγμένες. Η διαδικασία ολοκληρώνεται όταν δεν μπορεί να βρεθεί μικρότερο h(n), άρα έχουμε καταλήξει σε τοπικό ελάχιστο. Αποθηκεύουμε στην μνήμη το βέλτιστο h(n) που καταλήξαμε έτσι ώστε να έχουμε μία καλύτερη εκτίμηση h(s) για την επόμενη κλήση. Συλλέγοντας ένα πλήθος τιμών h(n), ο αλγόριθμος σε κάθε βήμα ενημερώνεται με εμπειρίες και μπορεί να ξέρει εκ των προτέρων αν τον βολεύει η random επανεκκίνηση που θα κάνει ώστε να πετύχει h(n) κοντά στο βέλτιστο.

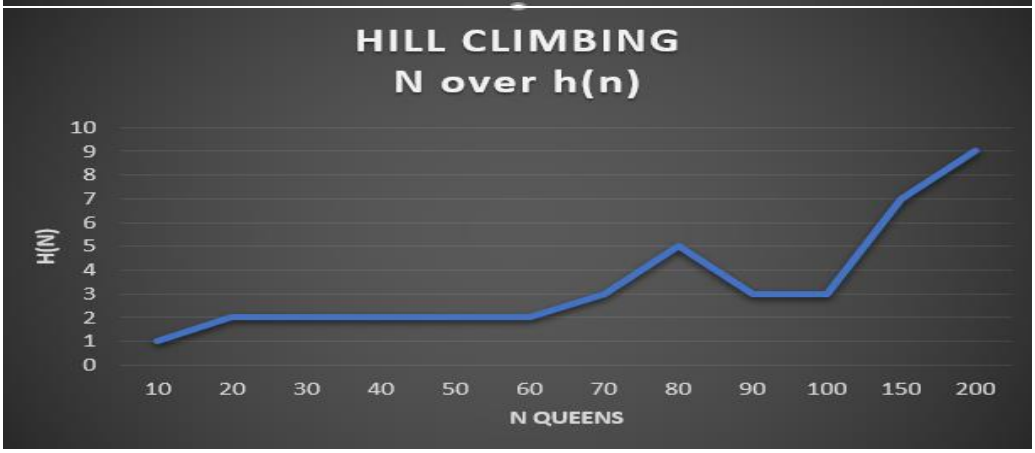
ΜΕΡΟΣ Γ

Στην συνέχεια δοκιμάσαμε τους αλγορίθμους μας, τρέχοντας τους πολλές φορές και για διάφορα n. Τα αποτελέσματα που παρατηρήσαμε παρατίθενται παρακάτω.

Genetic Algorithm			Hill Climbing		
N	time	h(n)	N	time(sec)	h(n)
10	0.005	0	10	0.015	1
20	0.012	0	20	0.04	2
30	0.023	0	30	0.125	2
40	0.039	0	40	0.39	2
50	1.035	3	50	1.18	2
60	1.31	7	60	2.58	2
70	1.7	5	70	5.96	3
80	2.06	7	80	10.83	5
90	2.43	6	90	17.62	3
100	2.94	11	100	31.74	3
150	6	20	150	220.2	7
200	10.14	24	200	992.3	9
500	59.56	76	500	1.00E+04	0
1000	245.87	295	1000	1.00E+06	0







Από τους παραπάνω πίνακες και τα διαγράμματα, έχουμε να εξάγουμε ενδιαφέροντα συμπεράσματα για την συμπεριφορά του κάθε αλγόριθμου.

Ξεκινώντας, παρατηρούμε ότι ο genetic algorithm πετυχαίνει εξαιρετικά μικρούς χρόνους ακόμα και για μεγάλα n, και αυτό οφείλεται στο ότι έχει μικρή χρονική πολυπλοκότητα O(g(nm + nm + n)) όπου g τα generations, n το μέγεθος του πληθυσμού (queens) και m το μέγεθος των individuals. Παρ'όλα αυτά, με την αύξηση του n παρατηρείται και η αστοχία του αλγορίθμου να βρεί βέλτιστες λύσεις, καθώς αυξάνεται σημαντικά και η h(n). Σε γενικές γραμμές είναι ένας πολύ αποδοτικός αλγόριθμος και προτιμάται για χαμηλές τιμές του n.

Όσον αφορά τον online hill climbing, είναι ένας πολύ πιο αργός αλγόριθμος, καθώς η χρονική του πολυπλοκότητα αυξάνεται αισθητά $O(n^2)$ λόγω των πολλών iterations που χρησιμοποιεί. Έτσι παρατηρείται αργή λύση του προβλήματος,μέχρι και αδυναμία αντιμετώπισης όσο αυξάνεται το n. Ωστόσο, τα αποτελέσματα που πετυχαίνει είναι πολύ πιο κοντα στο βέλτιστο,συγκριτικά με τον genetic algorithm, καθώς βρίσκει μικρότερο h(n) για μεγάλα n και μπορεί να αποτελέσει μία πιο αργή αλλά πολύ πιο αξιόπιστη λύση.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Ολοκληρώνοντας την εργασία μας,εφαρμόσαμε τις θεωρητικές μας γνώσεις στην πράξη και είχαμε την ευκαιρία να παρέμβουμε και να συμβάλουμε σε ήδη υπάρχοντες αλγορίθμους. Παρατηρήσαμε ότι τα εξαγόμενα αποτελέσματα συνάδουν πλήρως με το θεωρητικό υπόβαθρο και εντοπίσαμε τις ουσιαστικές διαφορές μεταξύ local και online αναζήτησης. Θεωρούμε ότι η επιλογή των αλγορίθμων ήταν σωστή, καθώς παρατηρήθηκαν τα αναμενόμενα αποτελέσματα. Τέλος, η επιλογή του περιβάλλοντος προγραμματισμού έγινε σκόπιμα,θέλοντας να επιχειρήσουμε να κάνουμε κάτι διαφορετικό, και ο έξτρα εισαγόμενος βαθμός δυσκολίας αντιμετωπίστηκε επιτυχώς.

ΑΝΑΦΟΡΕΣ

Ο κώδικας μας είναι μία μίξη δικής μας δουλειάς και τροποποιημένου κώδικα από τις παρακάτω πηγές. Πιο συγκεκριμένα, αντλήσαμε από τις πηγές κώδικα για τους local και online search algorithms, ενώ δική μας δουλειά είναι η αρχική συνάρτηση επιστροφής των απειλούμενων βασιλισσών, καθώς και ο εμπλουτισμός και η τροποποίηση των ήδη υλοποιημένων αλγορίθμων.

https://github.com/ivopascal/nQueens

Balint Hompot & Ivo de Jong University of Groningen | RUG · Institute of Artificial Intelligence and Cognitive Engineering