Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τεχνητή Νοημοσύνη

Ανδρουλιδάκης Ιωάννης (03111030)

Βασιλάκης Γεώργιος (03111061)

27/12/2015

Ερώτηση 1

Παρουσίαση προβλήματος και σκιαγράφηση υλοποίησης

Το πρόβλημα αφορά την εύρεση λύσης σε χώρο καταστάσεων.Για το λόγο αυτό, χρησιμοποιήσαμε τον αλγόριθμο A* προκειμένου 2 ρομπότ να βρούν το δρόμο τους προς ένα στόχο σ' ένα χάρτη 2 διαστάσεων από μια συγκεκριμένη αρχική θέση, περνώντας από συγκεκριμένα σημεία προηγουμένως.
Ο χώρος καστάσεων δίνεται ως ένα πλέγμα με) στις θέσεις που επιτρέπεται η κίνηση και X στις θέσεις που υπάρχουν εμπόδια , στον οποίο εφαρμόσαμε τον αλγόριθμο A*.
Τα 2 ρομπότ ξεκινάνε από τυχαίες θέσεις στο επίπεδο και πρέπει να συναντηθούν σ' ένα συγκεκριμένο σημείο το οποίο δίνεται σαν είσοδο από το χρήστη. Ωστόσο, πριν συναντηθούν στο τελικό σημειο-στόχο, θα πρέπει πρώτα να περάσουν από κάποια checkpoints .Απ τα checkpoints αυτά θεωρούμε ότι τα ρομπότ θα περάσουν σειριακά, ένα ένα από τα checkpoints.Οπότε, πιθανά collisions θα έχουμε όταν το ένα ρομπότ έχει ολοκληρώσει τη μετάβασή του προς κάποιο checkpoint ή προς τον τελικό στόχο.Επίσης, Θεωρούμε ότι τα 2 ρομπότ κινούνται με την ίδια ταχύτητα και οι κινήσεις που μπορούν να κάνουν είναι βόρεια,νότια,δυτικά και ανατολικά, όχι όμως διαγώνια.
Στο πρόβλημά μας θα πρέπει να παρατηρήσουμε ότι υπάρχουν και εμπόδια τα οποία τα 2 ρομπότ θα πρέπει να αποφύγουν, καθώς και πιθανές συγκρούσεις μεταξύ των ρομπότ τις οποίες θα αναλύσουμε παρακάτω.
Ο αλγόριθμος A* που χρησιμοποιήσαμε μας εγγυάται ότι θα μας δώσει το βέλτιστο μονοπάτι προς τα σημεία που θέλουμε κάθε φορά , χρησιμοποιώντας, πάντα , την κατάλληλη ευριστική.
Η γλώσσα στην οποία υλοποιήσαμε είναι η C++ , καθώς είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού, η οποία μας παρέχει αρκετές δεδομένων που διευκολύνουν την υλοποίηση του A* και τη γενικότερη υλοποίηση του αλγορίθμου.

Ερώτημα 2

Επιλογή δομών δεδομένων

Στην υλοποίησή μας χρησιμοποιήσαμε τις παρακάτω δομές δεδομένων:

- 2-dimensional Vector από Nodes τα οποία αποτελούν τους κόμβους του δέντρου που αναπαριστά το πρόβλημα.
- Pair για την αναπαράσταση των συντεταγμένων που βρίσκεται το κάθε ρομπότ κάποια χρονική στιγμή.
- Μια ουρά προτεραιότητας στην οποία εισάγουμε κάθε φορά τις καταστάσεις και επιλέγουμε εκείνη με το μικρότερο κόστος. Η συγκεκριμένη ουρά χρησιμοποιήθηκε για τη διευκόλυνση εύρεσης του ελαχίστου
- Set από Nodes για την εύρεση των γειτονικών κόμβων κάθε κόμβου
- Vectors από Nodes για το path που επιστρέφει ο A*

Ερώτημα 3

Βασικές Συναρτήσεις & Τελεστές

Ανάγνωση Εισόδου

Για την ανάγνωση των δεδομένων της εισόδου χρησιμοποιήσαμε τα streams της C++. Διαβάσαμε τις διαστάσεις του πλέγματος, τις αρχικές θέσεις των ρομπότ, τον τελικό προορισμό, το πλήθος των chekckpoints και τις συντεταγμένες των checkpoints. Στη συνέχεια δημιουργήσαμε το αντικείμενο graph, θέσαμε μέσω των κατάλληλων μεθόδων τα παραπάνω δεδομένα στο graph και τέλος διαβάσαμε και καθορίσαμε στο map τα επιτρεπτά και μη σημεία του χάρτη.

Ευριστικές Συναρτήσεις

Χρησιμοποιήθηκαν 2 συναρτήσεις:

- Η απόσταση manhattan ως υποεκτιμήτρια.
- Η απόσταση squared ως υπερεκτιμήτρια.

Κυρίως σώμα του A*

Εντός της κλασης Graph, ορίσαμε τη μέθοδο executeAstar() η οποία παίρνει σαν ορίσματα αρχική και τελική θέση και επιστρέφει ένα vector από Nodes, δηλαδή το ζητούμενο path από την αφετηρία στον προορισμό. Ως κριτήριο επιλογής του επόμενου κόμβου από την priority queue χρησιμοπιήσαμε τη μεταβλητή Score στην οποία αποθηκεύουμε το άθροισμα των :

- g = γνωστό κόστος από αφετηρία μέχρι τον τρέχοντα κόμβο
- h = ευριστική εκτίμηση του κόστους απ τον τρέχοντα κόμβο στον προορισμό.

Χώρος Καταστάσεων

Σε κάθε βήμα του αλγορίθμου η συνάρτηση getNeighbours μας επιστρέφει το σύνολο των γειτόνων του τρέχοντος κόμβου οι οποίοι προστίθενται στην ουρά προτεραιότητας openNode, ώστε στη συνέχεια να επιλεγεί ο κόμβος εκείνος με το μικρότερο κόστος .

Ερώτημα 4

Ευριστικές μέθοδοι

Ως admissible heuristic χρησιμοποιήθηκε η συνάρτηση απόστασης Manhattan. Ο λόγος που είναι admissible είναι διότι ένα robot χρειάζετα τόσα βήματα όσα δείχνει η απόστασης Manhattan για να φτάσει στο στόχο στην καλύτερη περίπτωση, αν δεν συναντήσει εμπόδια. Διαφορετικά χρειάζεται περισσότερα βήματα. Ως non-admissible heuristic χρησιμοποιήθηκε η συνάρτηση $x^2 + y^2$. Αυτή η συνάρτηση είναι non-admissible heuristic. Ο λόγος είναι ότι υπερεκτιμά την απόσταση προς το στόχο. Για παράδειγμα, αν δεν υπάρχουν καθόλου εμπόδια ανάμεσα στη θέση που βρίσκεται το robot και στο στόχο, τότε ο υπερεκτιμητής θα δώσει ως αποτέλεσμα μεγαλύτερο απ’ ό,τι η πραγματική απόσταση που θα διένυε η βέλτιστη λύση. Γι’ αυτό το λόγο, ενδέχεται ο υπερεκτιμητής να οδηγηθεί σε υποβέλτιστα αποτελέσματα, αν και σε λιγότερο πλήθος βημάτων.

Ερώτημα 5

Resolve Collision

Στην υλοποίησή μας θεωρούμε ότι το μόνο πιθανό Collision που μπορεί να υπάρξει είναι στον τελικό στόχο, καθώς οι κινήσεις των 2 ρομπότ εκτελούνται σειριακά και σ' αυτήν την περίπτωση το υλοποιούμε έτσι ώστε το ρομπότ που κινείται να γνωρίζει τη θέση του σταματημένου ρομπότ και να το θεωρεί σαν εμπόδιο ώστε να αποφύγει τη σύγκρουση.

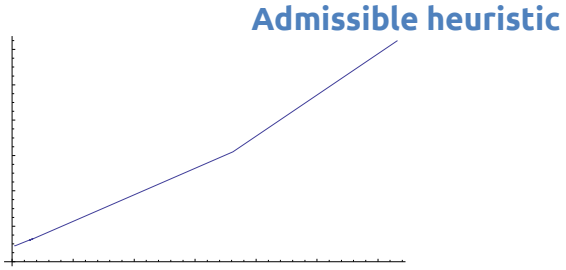
Ωστόσο, αν τα ρομπότ τρέχανε παράλληλα, ο αλγόριθμος που μπορεί να χρησιμοποιηθεί για την αποφυγή συγκρούσεων μπορεί να τρέξει μετά την εφαρμογή του A* και για τα δύο robot και να κάνει το robot A (που επιλέγεται ώστε να παραχωρεί προτεραιότητα) απλώς να περιμένει αν το robot B βρίσκεται στο δρόμο του μία δεδομένη χρονική στιγμή και είναι ο ακόλουθος:

```
Resolve-Collision( PathA, PathB ):
        NewPathA = []
        j ≔ 0
        for t ≔ 0 to |PathA| - 2:
                if PathB[ i ] = PathA[ i ]:
                        NewPathA[ j ] ≔ WAIT
                else:
                        NewPathA[ j ] ≔ PathA[ i ]
        return ( NewPathA, PathB )
```

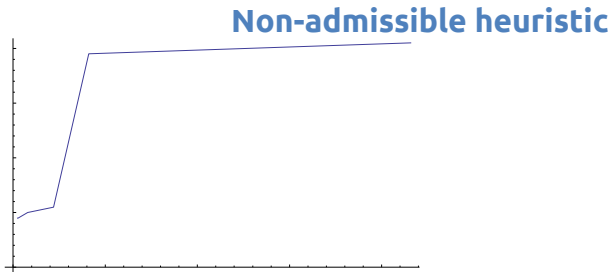
Η υλοποίηση της WAIT μπορεί να γίνει βάζοντας το ρομπότ να πηγαινοέρχεται ανάμεσα στα δύο τετραγωνάκια που επισκέφθηκε πιο πρόσφατα για ένα γύρο αν είμαστε υποχρεωμένοι να κινούμαστε. Στη συνέχεια, όποιο από τα ρομπότ A ή B φτάνει τελευταίο (δηλαδή έχει λόγου χάρη |PathA| < |PathB|), μπορεί να αφαιρέσει το τελευταίο βήμα από τη διαδρομή του ώστε να καταλήξει σε κουτάκι γειτονικό του στόχου

Ερώτημα 6

Παρατίθενται τα στατιστικά αποτελέσματα της εκτέλεσης 6 διαφορετικών αρχείων εισόδου.



Χρόνος εκτέλεσης συναρτήσει του αριθμού των κόμβων



Χρόνος εκτέλεσης συναρτήσει του αριθμού των κόμβων

Και στις δύο περιπτώσεις ο χρόνος εκτέλεσης είναι αμελητέος για τις μικρές δοκιμαστικές περιπτώσεις και πολύ μικρός για τις μεγαλύτερες. Για την μέτρηση χρησιμοποιήθηκε η κλήση time του UNIX, χωρίς να συμπεριλαμβάνονται οι χρόνοι εκτύπωσης των αποτελεσμάτων και οπτικοποίησης.

Ερώτημα 7

Παρακάτω φαίνεται το text output του προγράμματος σε πιθανή περίπτωση εντοπισμού σύγκρουσης για ένα παράδειγμα κίνησης:

```
Robot A path:
(1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (4, 5), (5, 5), (6, 5), (7, 5)
Robot B path:
(4, 1), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (4, 5), (5, 5), (6, 5), (7, 5)
First potential collision at ( 3, 5 ).
Avoiding collision by making Robot A wait.
```