ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολη ΗΜ&ΜΥ
Τεχνητή Νοημοσύνη
Άσκηση 1
Ακ. Έτος 2012-13

Παπαδημητρίου Κων/νος | ΑΜ: 03108769
Ημερομηνία: 10/04/2013

1. Περιγραφή έργου

Το πρόβλημα που επιλύουμε είναι ένα πρόβλημα αναζήτησης. Συγκεκριμένα με δεδομένα τον χάρτη ενός δωματίου και τις θέσεις 2 ρομπότ, προσπαθούμε να βρούμε τη διαδρομή που πρέπει να ακολουθήσει το ένα ρομπότ ώστε να συναντήσει το άλλο. Το δωμάτιο, εκτός από τα 2 ρομπότ, έχει και διάφορα εμπόδια, στα οποία δεν μπορεί να βρεθεί κάποιο ρομπότ. Το ρομπότ που σχεδιάζει τη διαδρομή του (robot_1), κινείται με ταχύτητα τριπλάσια της ταχύτητας του άλλου (robot_2), το οποίο κινείται τυχαία στο χώρο. (Δηλαδή το robot_1 κινείται με ταχύτητα 3 θέσεις ανα κίνηση, ενώ το robot_2 με ταχύτητα 1 θέση ανα κίνηση.) Τέλος τα 2 ρομπότ δεν μπορούν να κινηθούν διαγώνια, αλλά μόνο μπροστά, πίσω, δεξιά ή αριστερά.

Έτσι κάθε φορά που κινείται το robot_2, το robot_1 πρέπει να σχεδιάσει ξανά τη διαδρομή που πρέπει να ακολουθήσει με στόχο τη νέα θέση του robot_2.

Για τη λύση αυτού του προβλήματος, υλοποιούμε τον αλγόριθμο A* και χρησιμοποιούμε ευριστικές συναρτήσεις (είτε admissible είτε non-admissible). Μέχρι να συναντηθούν τα 2 ρομπότ, το πρώτο επιλέγει τυχαία μια θέση στην οποία μπορεί να πάει και το αφού ανανεωθεί η θέση του, το δεύτερο εκτελεί τον αλγόριθμο αναζήτησης και κινείται 3 θέσεις.

Για την υλοποίηση της λύσης χρησιμοποιήθηκε η γλώσσα C.

2. Αρχείο εισόδου

Το αρχείο εισόδου, θα είναι λίγο διαφορετικό από αυτό που δίνεται στην εκφώνηση της άσκησης. Συγκεκριμένα στη πρώτη γραμμή βρίσκονται 2 ακέραιοι, με τις διαστάσεις του δωματίου (X, Y) . Στις επόμενες 2 βρίσκονται οι συντεταγμένες των θέσεων των 2 ρομπότ $(x_1 y_1)(x_2 y_2)$. Στις επόμενες Y γραμές βρίσκονται X χαρακτήρες, που είναι είτε 'X' είτε 'O', ανάλογα με το αν υπάρχει ή όχι εμπόδιο στην αντίστοιχη θέση. Αξίζει να σημειωθεί πως ενώ η αρίθμηση στον άξονα X είναι αύξουσα από αριστερά προς τα δεξιά, στον άξονα Y είναι αύξουσα από πάνω προς τα κάτω.

Για να μη δημιουργηθούν προβλήματα, επισυνάπτεται και ένας testcase_generator, ο οποίος δημιουργεί τέτοια αρχεία εισόδου. Αυτό που κάνει, είναι να δημιουργεί έναν τυχαίων διαστάσεων πίνακα, να τον γεμίζει με 'O', και στη συνέχεια εκτελώντας προκαθορισμένο αριθμό επαναλήψεων γεμίζει διάφορες θέσεις με 'X'. Τελικά οι θέσεις των 2 ρομπότ γίνονται 'O'.

3. Δομές για την υλοποίηση.

- Περιγραφή του χώρου.

Ο χώρος, διαβάζεται από το αρχείο εισόδου και για τη περιγραφή του χρησιμοποιούμε τη δομή *map_node* η οποία περιέχει πληροφορίες που θα χρησιμοποιηθούν από τον A*, πληροφορίες για το αν υπάρχει ή όχι εμπόδιο στη δεδομένη θέση και για το αν βρέθηκε κάποιο robot στη θέση αυτή (θα χρησιμοποιηθεί για την εκτύπωση του αποτελέσματος).

- Δομές για τη κίνηση

Για τον αλγόριθμο αναζήτησης χρησιμοποιούμε μια νέα δομή, τη *search_node*,

η οποία περιέχει πληροφορίες σχετικά με τη θέση του χώρου στην οποία αναφέρεται, την (πραγματική) απόσταση από την τρέχουσα αρχική θέση και την (προβλεπόμενη) απόσταση από τον στόχο. Για την προβλεπόμενη απόσταση εκτελείται κάποια ευριστική συνάρτηση, την επιλογή της οποίας κάνει ο χρήστης. Υποστηρίζεται η απόσταση Manhattan $(|x_1-x_2|+|y_1-y_2|)$ καθώς και η Ευκλείδεια απόσταση (υψωμένη στο τετράγωνο) $((x_1-x_2)^2+(y_1-y_2)^2)$. Τέλος περιέχει και ένα index το οποίο χρησιμοποιήται για την ανανέωση της ουράς προτεραιότητας καθώς και δείκτες για τους εύρεση των γειτόνων μίας θέσης όπως και την εύρεση της θέσης "γονέα" (από που ήρθε το robot).

Ας αναλύσουμε λοιπόν τώρα λίγο περισσότερο την *map_node*, η οποία περιέχει έναν δείκτη στο αντίστοιχο *search_node* (αν υπάρχει τη δεδομένη χρονική στιγμή), και 2 ακεραίους που αναφέρουν το αν το *search_node* αυτό είναι up to date, αν είναι ανοιχτό (στην ουρά ποτεραιότητας) και αν είναι κλειστό (έχει ήδη βγει από την ουρά). Για να γίνουν αυτοί οι έλεγχοι, κοιτάμε αν ο κάθε ακέραιος είναι ίσος με τον αριθμό των κινήσεων που έχουν γίνει. Δηλαδή, την n-οστή φορά που εκτελείται ο A*, αν ένα *map_node* έχει opened = n, σημαίνει πως έχει *search_node* που είναι up to date και αυτό βρίσκεται στην ουρά προτεραιότητας. Ομοίως ελέγχουμε και το αν είναι closed.

- Ουρά προτεραιότητας

Για την ουρά προτεραιότητας χρησιμοποιούμε σωρό. Έγινε προσπάθεια "βιβλιοθηκοποίησής" του ώστε να χρησιμοποιηθεί και σε άλλες ασκήσεις και projects, αλλά λόγω έλλειψης του απαιτούμενου χρόνου αναβλήθηκε! Τώρα απλά παρέχονται οι βασικές του λειτουργίες, για την υλοποίηση του συγκεκριμένου έργου.

- Τελική διαδρομή

Για την τελική διαδρομή των αλγορίθμων μετακίνησης χρησιμοποιούμε μια απλή δομή που περιέχει την θέση του robot και ένα δείκτη στην επόμενη θέση. Είναι δουλειά του κυρίου προγράμματος που καλεί τον καθεμία συνάρτηση μετακίνησης να κρατήσει τις θέσεις που πρέπει (για το robot_2, μέχρι 3 για το robot_1).

4. Λεπτομέρειες υλοποίησης

Για την υλοποίηση των μετακινήσεων έχουμε δύο συναρτήσεις. Η move_1() είναι για το robot_1 και η move_2() για το robot_2. Η move_1() υλοποιεί τον αλγόριθμο A* και δέχεται σαν όρισμα την περιγραφή του χώρου, καθώς και την ευριστική συνάρτηση που θα χρησιμοποιηθεί. Με δεδομένες λοιπόν τις θέσεις των δύο robot (global μεταβλητές) βρίσκει μια (την ελάχιστη για admissible heuristic) διαδρομή από το robot_1 στο robot_2. Για την υλοποίηση του αλγορίθμου χρησιμοποιήθηκαν πληροφορίες από το www.wikipedia.org.
Η move_2() βρίσκει τους γείτονες του robot_2 και επιλέγει ψευδοτυχαία (με χρήση srand() time() και rand()) έναν γείτονα και επιστρέφει δείκτη σε αυτόν.
Είναι δουλειά του κύριου προγράμματος να ανανεώσει τη θέση του κάθε robot.

Επίσης υλοποιούνται κάποιες συναρτήσεις, για την εύρεση των γειτόνων, καθώς και για την δημιουργία της διαδρομής που βρέθηκε με τον A*. Οι συναρτήσεις

είναι η getChildren() και η reconstruct_path(). Η getChildren() με τη σειρά της χρησιμοποιεί την makeChildren(), η οποία δέχεται τις συντεταγμένες μιας θέσης στο χώρο, ελέγχει αν υπάρχει εμπόδιο στη θέση αυτή. Αν δεν υπάρχει δημιουργεί (αν χρειαστεί) ένα *search_node* για τη θέση αυτή και επιστρέφει δείκτη σε αυτό. Διαφορετικά επιστρέφει *NULL*. Η getChildren() με τη σειρά της δημιουργεί μια σειρά από *searh_node*s και επιστρέφει ένα δείκτη σε αυτά.
Η reconstruct_path(), χρησιμοποιεί τους δείκτες *came_from* για να βρει τη διαδρομή που ακολουθήθηκε μέχρι το στόχο.

5. *Κύριο πρόγραμμα και εκτύπωση εξόδου*

Το κύριο πρόγραμμα, ενώ διαβάσει το αρχείο εισόδου, αναλαμβάνει να εκτελέσει όσες κινήσεις χρειαστούν, ώστε τα 2 ρομπότ να συναντηθούν. Μια κίνηση περιλαμβάνει την εύρεση της επόμενης θέσης του robot_2 και ανανέωση αυτής· την εκτέλεση του αλγορίθμου A* ώστε να βρεθεί μια διαδρομή (ελάχιστη αν γίνει χρήση της admissible heuristic) με αρχή τη θέση του robot_1 και τέρμα τη νέα θέση του robot_2 και την μετακίνηση του robot_1 (το πολύ) 3 θέσεις με βάση τη διαδρομή αυτή. Επειδή η move_2() χρησιμοποιεί ψευδοτυχαίους αριθμούς, κατα την εκτέλεση του προγράμματος παρατηρείται κάποια επανάληψη των κινήσεων από το robot_2. Έτσι στο αρχείο main.h μπορούν να μεταβληθούν οι ταχύτητες αλλάζοντας τα *SPEED_1* και *SPEED_2*. Στην πράξη λοιπόν σε κάθε κίνηση πραγματοποιούνται *SPEED_2* επαναλήψεις της κλήσης της move_2() και της ανανέωσης της θέσης του robot_2 και στη συνέχεια γίνεται μετακίνηση του robot_1 (το πολύ) *SPEED_1* θέσεις σύμφωνα με τη διαδρομή που επέστρεψε ο A*.

Όσο για την εκτύπωση της εξόδου, με την πραγματοποίηση της μετακίνησης ενός ρομπότ, αυτό αναφέρει τις θέσεις από τις οποίες πέρασε (όπως και αυτή, στην οποία κατέληξε). Όταν τελικά τα 2 ρομπότ συναντηθούν, εκτυπώνεται και ο χώρος όπως αυτός λήφθηκε από το αρχείο εισόδου, ενώ φαίνονται με διαφορετικά χρώματα οι διαδρομές που ακολούθησαν τα 2 ρομπότ. (Μπλε για το robot_1 και κόκκινο για το robot_2).

6. *Τελικά*

Μαζί με τη παρούσα αναφορά, επισυνάπτονται και ο κώδικας της εφαρμογής (συμπληρωματικά ένας testcase_generator) μαζί με 10 διαφορετικά testcases (Αυτό που δίνεται με την εκφώνηση, τροποποιημένο σύμφωνα με την παράγραφο 2, όπως και 9 που δημιουργήθηκαν με τον testcase_generator). Επίσης υπάρχει ένα αρχείο txt (stats.txt) που περιέχει πολύ σύντομα στατιστικά στοιχεία (τις διαστάσεις και τον αριθμό των κινήσεων που απαιτήθηκαν) από 4 διαφορετικές εκτελέσεις του κάθε αρχείου (2 για κάθε ευριστική)

Μία πρόχειρη (και προφανής) παρατήρηση όσον αφορά τα στατιστικά αποτελέσματα είναι πως όταν έγινε χρήση της non-admissible heuristic χρειαστήκαν περισσότερες κινήσεις ώστε να συναντηθούν τα 2 ρομπότ.

7. *References*

Διαφάνειες μαθήματος. (*www.mycourses.gr*) - Wikipedia. (*www.wikipedia.org*)

Ζητώ συγγνώμη αν υπάρχουν ορθογραφικά λάθη!