Μάθημα: Τεχνητή Νοημοσύνη

Άσκηση 1: Αναζήτηση σε λαβύρινθο

Ονοματεπώνυμο: Κωνσταντίνος Βαρδάκας

ΑΜ: 522

Email: pcs0522@uoi.gr

Εισαγωγή

Η πρώτη εργασία υλοποιεί δύο αλγορίθμους αναζήτησης σε γλώσσα προγραμματισμού C, τον Uniform Cost Search (UCS) και τον A*, για την επίλυση ενός προβλήματος πλοήγησης ρομπότ σε λαβύρινθο NxN. Ο λαβύρινθος δημιουργείται τυχαία με πιθανότητα p για ελεύθερα κελιά, ενώ τα κελιά A (κάτω αριστερά, συντεταγμένες (N-1, 0)) και B (άνω δεξιά, συντεταγμένες (0, N-1)) είναι πάντα ελεύθερα, όπως και τα κελιά εκκίνησης (S) και στόχου (G). Το ρομπότ μπορεί να κινείται οριζόντια, κατακόρυφα ή διαγώνια με κόστος 1, καθώς και να μεταφέρεται απευθείας μεταξύ των κελιών A και B με κόστος 2. Στόχος είναι η εύρεση της βέλτιστης διαδρομής από το S στο G, με εκτύπωση του λαβυρίνθου, του μονοπατιού, του κόστους και του αριθμού επεκτάσεων για κάθε αλγόριθμο.

Υλοποίηση

Αρχικά, ο λαβύρινθος δημιουργείται ως πίνακας NxN, όπου κάθε κελί είναι ελεύθερο με πιθανότητα p. Τα κελιά S, G, A και B ορίζονται ρητά ως ελεύθερα. Οι συντεταγμένες των κελιών S και G εισάγονται από τον χρήστη.

Υλοποίηση του UCS

Ο αλγόριθμος Uniform Cost Search (UCS) υλοποιείται χρησιμοποιώντας μια ουρά προτεραιότητας, η οποία διαχειρίζεται κόμβους με βάση το συνολικό κόστος g(n)

από το αρχικό κελί S. Κάθε κόμβος περιλαμβάνει τις συντεταγμένες (x, y), το κόστος g(n), και τις συντεταγμένες του γονέα για την ανακατασκευή του μονοπατιού.

Η διαδικασία ξεκινά με την αρχικοποίηση της ουράς προτεραιότητας, στην οποία εισάγεται ο αρχικός κόμβος S με κόστος g(n)=0. Σε κάθε βήμα, επιλέγεται από την ουρά ο κόμβος με το μικρότερο g(n). Αν αυτός ο κόμβος αντιστοιχεί στο τελικό κελί G, τότε το μονοπάτι ανακατασκευάζεται χρησιμοποιώντας τις πληροφορίες για τους γονείς των κόμβων και επιστρέφεται το συνολικό κόστος. Σε διαφορετική περίπτωση, εξετάζονται όλες οι πιθανές μεταβάσεις προς τα οκτώ γειτονικά ελεύθερα κελιά (οριζόντιες, κατακόρυφες και διαγώνιες), κάθε μία με κόστος ίσο με 1. Αν το νέο κόστος μετάβασης σε ένα γειτονικό κελί είναι μικρότερο από αυτό που έχει ήδη καταγραφεί για το κελί αυτό, τότε οι πληροφορίες του κόμβου ενημερώνονται και επανεισάγεται στην ουρά. Επιπλέον, αν ο τρέχων κόμβος είναι το ειδικό κελί A, εξετάζεται η πιθανή ειδική μετάβαση προς το κελί B με κόστος ίσο με 2, και αντίστροφα. Η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο G ή μέχρι να αδειάσει η ουρά, οπότε και δηλώνεται ότι δεν υπάρχει διαθέσιμο μονοπάτι. Καθ' όλη τη διάρκεια της αναζήτησης καταγράφεται ο αριθμός των κόμβων που επεκτάθηκαν.

Η εκτύπωση του μονοπατιού γίνεται με ανακατασκευή από το G προς το S, χρησιμοποιώντας έναν πίνακα για να σημειώνονται τα κελιά του μονοπατιού, ενώ ο λαβύρινθος εμφανίζεται με σύμβολα για το S, G, το μονοπάτι (*), τα ελεύθερα κελιά (κενό) και τα εμπόδια (X).

Υλοποίηση του A*

Ο αλγόριθμος A* υλοποιείται με παρόμοιο τρόπο όπως ο UCS, με τη βασική διαφορά ότι χρησιμοποιεί τη συνάρτηση f(n)=g(n)+h(n), όπου h(n) είναι η ευρετική εκτίμηση του κόστους από τον κόμβο n έως τον στόχο, ώστε να καθοδηγεί πιο αποτελεσματικά την αναζήτηση. Η δομή των κόμβων παραμένει ίδια με εκείνη του UCS, με την προσθήκη της τιμής f(n), η οποία χρησιμοποιείται για την ταξινόμηση στην ουρά προτεραιότητας.

Ομοίως με τον αλγόριθμο UCS, η διαδικασία ξεκινά με την εισαγωγή του αρχικού κελιού S στην ουρά, με τιμή g(n)=0 και έτσι το αρχικό συνολικό κόστος καθορίζεται από την τιμή της ευρετικής συνάρτης του A*, f(n)=h(S). Σε κάθε

επανάληψη, επιλέγεται ο κόμβος με το μικρότερο f(n). Αν ο κόμβος αυτός αντιστοιχεί στο κελί G, τότε το μονοπάτι πάλι ανακατασκευάζεται μέσω των γονέων και επιστρέφεται το συνολικό κόστος. Διαφορετικά, εξετάζονται όλες οι δυνατές κινήσεις προς τα οκτώ γειτονικά ελεύθερα κελιά, με κόστος 1 για κάθε μετακίνηση. Αν η νέα τιμή g(n) για ένα κελί είναι μικρότερη από την ήδη καταγεγραμμένη, τότε επανυπολογίζεται το h(n), ενημερώνεται το f(n), και ο κόμβος προστίθεται εκ νέου στην ουρά. Όπως και στον UCS, αν ο κόμβος αντιστοιχεί στο ειδικό κελί A ή B, εξετάζεται η ειδική κίνηση μεταξύ τους με κόστος 2. Η αναζήτηση συνεχίζεται έως ότου φτάσει στον στόχο ή εξαντληθούν οι κόμβοι, οπότε και δηλώνεται η αποτυχία εύρεσης μονοπατιού. Σε όλη τη διάρκεια της διαδικασίας καταγράφεται ο αριθμός των κόμβων που επεκτάθηκαν.

Ευρετική Συνάρτηση του A*

Η ευρετική συνάρτηση h(n) υπολογίζει το ελάχιστο εκτιμώμενο κόστος από ένα κελί (x, y) στο τελικό κελί (gx, gy). Υπολογίζεται ως το ελάχιστο από τρεις επιλογές:

1. Απευθείας απόσταση: Χρησιμοποιεί την απόσταση Chebyshev, που ορίζεται ως το μέγιστο της απόλυτης διαφοράς των x-συντεταγμένων και των y-συντεταγμένων μεταξύ (x,y) και (gx,gy).

2. Απόσταση μέσω κελιού A: Υπολογίζεται ως η απόσταση Chebyshev από το (x,y) στο A (N-1,0), συν το κόστος μεταφοράς (2), συν η απόσταση Chebyshev από το B (0,N-1) στο (gx,gy).

3. Απόσταση μέσω κελιού B: Υπολογίζεται ως η απόσταση Chebyshev από το (x,y) στο B (0,N-1), συν το κόστος μεταφοράς (2), συν η απόσταση Chebyshev από το A (N-1,0) στο (gx,gy).

   Η τελική ευρετική είναι το ελάχιστο αυτών των τριών τιμών.

Αποδεκτικότητα της Ευρετικής

Η ευρετική συνάρτηση που χρησιμοποιείται είναι αποδεκτή, καθώς δεν υπερεκτιμά ποτέ το πραγματικό κόστος της βέλτιστης διαδρομής. Συγκεκριμένα, η

απόσταση Chebyshev θεωρεί ότι το ρομπότ μπορεί να κινείται διαγώνια και ότι δεν υπάρχουν εμπόδια. Δεδομένου ότι κάθε κίνηση (οριζόντια, κατακόρυφη ή διαγώνια) έχει κόστος 1, η απόσταση Chebyshev παρέχει το ελάχιστο δυνατό κόστος σε έναν λαβύρινθο χωρίς εμπόδια και συνεπώς είναι πάντα μικρότερη ή ίση από το πραγματικό κόστος της διαδρομής. Επιπλέον, όταν η ευρετική περιλαμβάνει την ειδική μεταφορά μέσω του κελιού A, υπολογίζεται η απόσταση Chebyshev από το τρέχον κελί έως το A, προστίθεται το σταθερό κόστος της μεταφοράς (2), και έπειτα προστίθεται η απόσταση από το B έως τον στόχο G. Το ίδιο ισχύει και για την αντίστροφη διαδρομή μέσω B.

Εφόσον τελικά επιλέγεται το ελάχιστο μεταξύ αυτών των τριών εκτιμήσεων, είναι αδύνατη η υπερεκτίμηση του συνολικού κόστους. Επιπλέον, η ευρετική συνάρτηση είναι συνεπής (ή μονοτονική), καθώς ικανοποιεί την ιδιότητα ότι για κάθε κόμβο n και κάθε διάδοχό του n′, η τιμή της ευρετικής στο n, δηλαδή h(n), δεν υπερβαίνει το άθροισμα του κόστους μετάβασης από το n στο n′, δηλαδή c(n,n′), και της ευρετικής στο n′, δηλαδή h(n′). Μαθηματικά, αυτό εκφράζεται ως:

$$h(n) \leq c(n, n') + h(n')$$

Έτσι, αυτή η ευρετική συνάρτηση, επιτρέπει στον αλγόριθμο A* να βρει μια βέλτιστη λύση καθώς είναι αποδεκτή. Επίσης, κάθε κόμβος επεκτείνεται το πολύ μία φορά και έτσι ο αλγόριθμος καθίσταται αποδοτικός λόγω της συνέπειας της συνάρτησης.

Αποτελέσματα και Συμπεράσματα

Για την αξιολόγηση των αλγορίθμων, εκτελέστηκαν δοκιμές με διάφορους λαβυρίνθους. Ενδεικτικά αποτελέσματα για 5 λαβυρίνθους με N=10, p=0.7 (οι διαφορές των 5 λαβυρίνθων προκύπτει από τη τυχαιότητα των κελιών να είναι ελεύθερα ή μπλοκαρισμένα), S=(9,3), και G=(2,8) (με πρόσβαση στα σημεία A και B):

κόστος μονοπατιού					
αριθμός πειράματος	1	2	3	4	5
UCS	7	7	8	8	7
A*	7	7	8	8	7

Και οι δύο αλγόριθμοι εντοπίζουν το βέλτιστο μονοπάτι κάθε λαβυρίνθου. Στους συγκεκριμένους λαβύρινθους απαιτούσε την αξιοποίηση των κελιών A και B.

αριθμός επεκτάσεων						
αριθμός πειράματος	1	2	3	4	5	mean
UCS	55	52	63	59	61	58.0
A*	14	14	22	15	14	15.8
Διαφορά	41	38	41	44	47	42.2

Ο αλγόριθμος A* είναι σημαντικά πιο αποδοτικός από τον UCS, καθώς απαιτεί πολύ μικρότερο αριθμό επεκτάσεων για να εντοπίσει τη βέλτιστη λύση. Αυτή η συμπεριφορά είναι αναμενόμενη, αφού ο A* αξιοποιεί και την ευρετική συνάρτηση που παρέχει επιπλέον πληροφορία για το εκτιμώμενο κόστος από κάθε κελί προς τον στόχο. Με βάση αυτή την προσέγγιση, η προτεραιότητα των κελιών στην ουρά αλλάζει, δίνοντας προβάδισμα σε εκείνα που εκτιμάται ότι οδηγούν ταχύτερα και οικονομικότερα στον στόχο. Παρόμοια αποτελέσματα λήφθηκαν και για πειράματα με άλλες τιμές των υπερπαραμέτρων (N, p, S, G).