Ημερομηνία: 27/03/2021

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ: ΗΜΜΥ

Τεχνητή Νοημοσύνη
ΠΛΗ311

1η Προγραμματιστική Άσκηση

Ομάδα Εργασίας LAB31146722 :

Περάκης Εμμανουήλ **ΑΜ: 2017030099**
Γεωργακάς Ιωάννης-Ιάσων **ΑΜ: 2017030021**

Εισαγωγή και Περιγραφή του Προβλήματος:

Στην παρούσα προγραμματιστική άσκηση, τίθεται προς λύση το πρόβλημα εύρεσης της βέλτιστης διαδρομής με το οποίο έρχεται αντιμέτωπος ένας εργαζόμενος σε μία μεγαλούπολη, προσπαθώντας να μεταβεί από τον τόπο διαμονής στον τόπο δουλειάς του, κατά τη διάρκεια περίπου 3 μηνών (80 εργάσιμων ημερών). Το πρόβλημα αυτό, στα πλαίσια αυτής της προγραμματιστικής άσκησης, λύνεται με δύο διαφορετικές μεθόδους αναζήτησης, μία offline το προηγούμενο βράδυ και μία online το πρωΐ, στα μέρη Α και Β αντιστοίχα. Κατά την offline αναζήτηση χρησιμοποιούνται δύο διαφορετικοί αλγόριθμοι για την εύρεση του πλάνου που θα ακολουθηθεί κατά τη μετακίνηση του την επόμενη μέρα. Ο πρώτος αλγόριθμος offline αναζήτησης είναι ο Uniform Cost Search (UCS), ο οποίος καταττάσσεται στους αλγορίθμους απληροφόρητης αναζήτησης. Ο δεύτερος offline αλγόριθμος είναι ο Iterative Deepening A* (IDA*), που είναι πληροφορημένης αναζήτησης, καθώς χρησιμοποιεί πληροφορία που συλλεγει από την προηγούμενη μέρα για την ακρίβεια της πρόβλεψης και εξισορροπεί κατάλληλα τα βάρη. Τέλος, ο online αλγόριθμος που χρησιμοποιείται είναι ο Learning Real Time A* (LRTA*). Περισσότερες λεπτομέρειες για αυτόν στο μέρος Β.

Υλοποίηση και Ανάλυση Αποτελεσμάτων:

Για την προσομοίωση του προβλήματος απαιτούταν αρχικά το διάβασμα, από το αρχείο που δίνεται, της γενικής δομής του γράφου (multigraph) δηλαδή των διασυνδέσεων μεταξύ των κόμβων αλλά και τα «φυσιολογικά» κόστη διάνυσης κάθε ακμής-οδού. Ύστερα για κάθε μέρα διαβάζονται οι ταμπέλες κίνησης για κάθε δρόμο και τα βάρη των οδών τροποποιούνται ως εξής:

- Low traffic: predicted_weight = 0.9*normal_weight
- Normal traffic: predicted_weight = 1*normal_weight
- Heavy traffic: predicted_weight = 1.25*normal_weight

Οι πληροφορίες αυτές αποθηκεύονται σε ένα στιγμιότυπο της κλάσης Day και ανακτούνται κάθε βράδυ για τη δημιουργία του πλάνου στους δύο offline αλγορίθμους.

Για τη διαπίστωση της ορθής λειτουργίας των δύο αλγορίθμων χρησιμοποιήθηκε βοθητικός γράφος με λιγότερες ακμές και κορυφές σε σύγκριση με τους γράφους της πόλης που δίνονται. Η δομή αυτού του γράφου φαίνεται στο αρχείο testGraph (Δεν αντιπροσωπεύει σε καμία περίπτωση την πραγματική τοποθεσία των πόλεων).

Μέρος Α:

Το συγκεκριμένο μέρος της άσκησης χωρίζεται σε δύο υπομέρη, στα οποία αναλύονται οι δύο διαφορετικοί offline αλγόριθμοι.

i) Iterative Deepening A* (IDA*):

Ο Iterative Deepening A* (A* με Επαναληπτική Εμβάθυνση) είναι ένας βέλτιστος αλγόριθμος διάσχισης γράφων και δέντρων (ειδική περίπτωση γράφου) που εγγυάται την εύρεση του βέλτιστου μονοπατιού από έναν αρχικό κόμβο (σπίτι εργαζομένου) σε ένα κόμβο προορισμού (τόπος δουλειάς), αν και εφόσον η ευριστική συνάρτηση h(n) είναι παραδεκτή (admissible) και συνεπής (consistent). Ο αλγόριθμος αυτός δανείζεται τη λογική του Iterative Deepening DFS με τη διαφορά ότι κάθε φορά το νέο όριο (bound ή fringe) υπολογίζεται με βάση τη συνάρτηση f(n) = g(n) + h(n), που χρησιμοποιεί και ο A*, και όχι με βάση το βάθος που βρίσκεται κάθε φορά. Το αποτέλεσμα αυτού του τεχνάσματος είναι η χαμηλότερη χρήση μνήμης σε σχέση με τον A* όπου σε κάθε χρονική στιγμή είναι αποθηκευμένοι στη μνήμη όλοι οι κόμβοι των οποίων δεν έχει βρεθεί ακόμα το βέλτιστο μονοπάτι από τον αρχικό κόμβο.

Η απόδοση του αλγορίθμου επηρεάζεται σε μεγάλο βαθμό από την ευριστική συνάρτηση που επιλέγεται. Διακρίνονται οι εξής περιπτώσεις ως προς τη συμπεριφορά του αλγορίθμου A*, και κατ' επέκταση του IDA*, για διάφορα h(n):

- h(n) = 0 : Η συνάρτηση f(n) ταυτίζεται με τη g(n) και ο αλγόριθμος μετατρέπεται στον Dijkstra.

- h(n) πάντα χαμηλότερη από το ακριβές κόστος μετάβασης από τον κόμβο n στον τελικό κόμβο: Ο αλγόριθμος θα βρει εγγυημένα το βελτιστο μονοπάτι προς τον τελικό κόμβο, παρότι ο αριθμός των κόμβων που θα ελέγξει δε θα είναι ο ελάχιστος

- h(n) πάντα ίση με το ακριβές κόστος μετάβασης από τον κόμβο n στον τελικό κόμβο: Ο αλγόριθμος ακολουθεί μόνο το βέλτιστο μονοπάτι, ελέγχοντας το ελάχιστο πλήθος κόμβων

- h(n) δεν είναι πάντα χαμηλότερη από το ακριβές κόστος μετάβασης από τον κόμβο n στον τελικό κόμβο: Εδώ ο αλγόριθμος δεν εγγυάται την εύρεση του βέλτιστου μονοπατιού
- h(n) >> g(n): Ο αλγόριθμος μετατρέπεται στον Greedy Best First Search

Στην προκείμενη περίπτωση η ευριστικές συναρτήσεις όλων των κόμβων υπολογίζονται πριν την εκτέλεση του αλγορίθμου για τις 80 νύχτες. Θεωρώντας προσωρινά πως όλοι οι δρόμοι του γράφου έχουν χαμηλή κίνηση (low traffic), υπολογίζονται οι ελάχιστες αποστάσεις όλων των κόμβων από το στόχο. Με αυτόν τον τρόπο βεβαιώνεται πως κάθε εκτίμηση της απόστασης θα είναι μικρότερη ή ίση από την πραγματική απόσταση καθώς το καλύτερο δυνατό σενάριο είναι όλοι οι δρόμοι να έχουν χαμηλή κίνηση (admissible heuristic). Ο αλγόριθμος UCS που υλοποιείται, χρησιμοποιείται επίσης και για την εύρεση αυτών των ελάχιστων αποστάσεων για κάθε κόμβο.

Η συνάρτηση g(n) είναι σε κάθε πιθανή στιγμή ίση με την απόσταση του κόμβου από τον αρχικό κόμβο.

Η λογική του αλγορίθμου έχει ως εξής:

Σε κάθε επανάληψη διάσχυσε με βάση τον αλγόριθμο DFS το γράφο έως ότου βρεθεί ο στόχος ή ξεπεραστεί το όριο που έχει τεθεί για το συγκεκριμένο iteration. Το νέο όριο επιλέγεται να είναι το ελάχιστο f(n) = g(n) + h(n) από όλα τα f(n) που ξεπερνούν την τιμή του προηγούμενου ορίου. Ο αλγόριθμος τερματίζει όταν βρεθεί ο στόχος.

Το προβλεπόμενο κόστος διάσχισης μίας οδού υπολογίζεται με βάση μία πιθανοτική κατανομή η οποία δίνεται να είναι αρχικά:

$$\begin{matrix} p_1 = P(\omega_1 | \pi_1) = 0.6 \\ p_2 = P(\omega_2 | \pi_1) = 0.2 \\ p_3 = P(\omega_3 | \pi_1) = 0.2 \end{matrix} \quad , \quad \begin{matrix} \omega_i : actual\ traffic\ label \\ \pi_i : predicted\ traffic\ label \end{matrix}$$

Κάθε μέρα, με βάση το predicted label που δίνεται για κάθε οδό, το βάρος προκύπτει να είναι το εξής:

$$predicted\ weight = (p_1 \cdot \pi_1 + p_2 \cdot \pi_2 + p_3 \cdot \pi_3) \cdot normal\_weight$$

$$\pi.\chi. \quad \pi_1 = low = 0.9$$
$$predicted\ weight = (0.9 \cdot 0.6 + 1 \cdot 0.2 + 1.25 \cdot 0.2) \cdot normal\_weight = 0.99 \cdot normal\_weight$$

ii) Uniform Cost Search (UCS):

Ο παρών αλγόριθμος είναι αλγόριθμος απληροφόρητης αναζήτησης, δηλαδή δεν έχει παραπάνω πληροφορία από αυτή που του δίνεται κατά τον ορισμό του προβλήματος. Αυτό σημαίνει πως δε λαμβάνεται υπόψιν κάποια ευριστική συνάρτηση κατά τη δημιουργία του πλάνου το βράδυ. Ο UCS εφαρμόζει την ίδια λογική με τον αλγόριθμο του Dijkstra με τη διαφορά ότι δε βρίσκει τη βέλτιστη διαδρομή κάθε κόμβου από τον αρχικό κόμβο, αλλά προσπαθεί να φτάσει σε έναν προκαθορισμένο κόμβο στόχο βρίσκοντας το ελαφρύτατο μονοπάτι.

Η πολυπλοκότητα του αλγορίθμου είναι η εξής:

Χρονική Πολυπλοκότητα: $O(b^{1+floor(C^*/\varepsilon)})$
Χωρική Πολυπλοκότητα: $O(b^{1+floor(C^*/\varepsilon)})$

Μέρος Β:

Στο συγκεκριμένο μέρος εξετάζεται ένας αλγόριθμος onloine αναζήτησης, ο Learning Real Time A*.

Learning Real Time A* (LRTA*):

Ο αλγόριθμος LRTA*, όπως και όλοι οι αλγόριθμοι online αναζήτησης, δεν έχουν στη διάθεση τους καμία πληροφορία για τα βάρη των δρόμων που θα ακολουθήσουν και των ευριστικών συναρτήσεων που χαρακτηρίζουν τους προορισμούς των δρόμων αυτών. Στην ουσία, ο LRTA* ανανεώνει την εκτίμώμενη απόσταση από τον στοχο του προηγούμενου κόμβου που διέσχισε καθώς μόνο μετά την διάσχιση του κάθε δρόμου γίνεται γνωστό το βάρος του δρόμου, η τιμή της εκτιμώμενης απόστασης βελτιώνεται κερδίζοντας εμπειρία ο πράκτορας από την αλληλεπίδραση του με το περιβάλλον. Στη χειρότερη περίπτωση μπορεί να εξερευνήσει ένα περιβάλλον με n καταστάσεις σε $O(n^2)$ βήματα.

	UCS	IDA*	LRTA*
Visited Nodes Number	366	1677	835800 ns
Execution Time	2579400 ns	835800 ns	51163 ns
Mean Predicted Cost	111.43877	111.43877	-
Mean Real Cost	115.81813	115.81813	668.3551

Πίνακας 1: Σύγκριση Αλγορίθμων

Όπως φαίνεται από τον Πίνακα 1 ο αλγόριθμος UCS με τον αλγόριθμο IDA* βρίσκουν το ίδιο βέλτιστο μονοπάτι, όπως ήταν αναμενόμενο, οπότε το μέσο πραγματικό κόστος και το μέσο προβλεπόμενο κόστος ταυτίζεται και στους δύο αλγορίθμους offline αναζήτησης. Τα δύο κόστη δεν παρουσιάζουν διαφορές καθώς δε λαμβάνεται υπόψιν η ευριστική συνάρτηση κάθε κόμβου κατά τη μέτρηση της απόστασης από τον αρχικό κόμβο στο στόχο. Ο IDA* παρατηρείται να εκτελείται στο μισό περίπου χρόνο από αυτόν του UCS αλλά ο αριθμός κόμβων που επισκέπτεται είναι αρκετά μεγαλύτερος.

Ο LRTA* έχει κατά πολύ μειωμένο χρόνο εκτέλεσης και αριθμό επισκοπτόμενων κόμβων, αφού δεν βρίσκει κανένα πλάνο προπαρσκευαστηκά αλλά εκτελείται σε πραγματικό χρόνο κατά τη διάρκεια της μέρας.

Συμπεράσματα:

Μέσα απ ό την υλοποίηση και ανάλυση, των διαφορετικών μεθόδων αναζήτησης της βέλτιστης διαδρομής σε μία μεγαλούπολη (multigrph), έρχονται στην επιφάνεια τα προτερήματα και τα μειωνεκτήματα κάθε μίας από τις μεθόδους. Όταν δεν παρέχονται οι απαραίτητες πληροφορίες για το περιβάλλον που ο πράκτορας καλείται να διασχίσει, οι αλγόριθμοι online αναζήτησης είναι μονόδρομος. Οι offline αλγόριθμοι από την άλλη αποδεικνύονται πολυ αποτελεσματικοί μέσα σε περιβάλλοντα οπου υπάρχει γνώση της πληροφορίας εκ των προτέρων.

Documentation:

[1] http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html
[2] Artificial Intelligence A Modern Approach Third Edition, Stuart Russell, Peter Norvig
[3]https://en.wikipedia.org/wiki/Iterative_deepening_A*