Πολυτεχνείο Κρήτης

ΠΛΗ311
Τεχνητή Νοημοσύνη

1η Προγραμματιστική Εργασία

Σωφρονίου Παύλος…...……....ΑΜ:2015030129
Ζαφειρακόπουλος Αλέξανδρος.....ΑΜ:2015030127

Για οδηγίες εκτέλεσης του προγράμματος βλέπε παράρτημα 2

Σύνοψη εργασίας

Στην παρούσα προγραμματιστική εργασία ζητήθηκε να λυθεί το πρόβλημα εύρεσης βέλτιστης διαδρομής που αντιμετωπίζει ένας εργαζόμενος καθημερινά σε μια μεγαλούπολη δεδομένου ορισμένων περιορισμών που προσαρμόζουν το πρόβλημα πιο κοντά στην πραγματικότητα. Το πρόβλημα διαιρέθηκε σε δύο μέρη, το μέρος Α στο οποίο πρέπει να βρεθεί μία λύση *offline* με την χρήση ενός αλγορίθμου απληροφόρητης αναζήτησης (στην περίπτωση μας επιλέχθηκε ο *breadth first search*) καθώς και με την χρήση του αλγορίθμου *IDA** (πληροφορημένη αναζήτηση). Το μέρος Β αφορά την υλοποίηση του *online* αλγορίθμου *LRTA** για να επιχειρήσει να λύσει το πρόβλημα. Αξίζει να σημειωθεί πως η γλώσσα προγραμματισμού που επιλέχθηκε για την υλοποίηση είναι η python 3.8.0 .

Κομμάτι υλοποίησης Μέρους Α

Γράφος

Αρχικά για την εισαγωγή του γράφου σε μία δομή δεδομένων επιλέχθηκε το dictionary (ευρετήριο) της python (βλέπε παράρτημα 1.1). Πιο συγκεκριμένα, υλοποιήθηκε μια δομή τριών εμφωλευμένων ευρετηρίων όπου το κλειδί του πρώτου dictionary είναι ο κάθε κόμβος του γράφου με value ένα δεύτερο dictionary όπου αντικατοπτρίζει τα παιδιά-γείτονες του. Τα παιδιά του στην συνέχεια είναι αυτά κλειδιά του δεύτερου dictionary με value ένα τρίτο dictionary που αποτελείται από τους πιθανώς πολλαπλούς δρόμους και τα κόστη που συνδέουν τον πατέρα με το κάθε παιδί του. Παρακάτω επισυνάπτεται μία αναπαράσταση του τι περιγράφηκε για καλύτερη κατανόηση.

```
{ 'NodeA': {  'NodeB': { 'Road0': '44' },
              'NodeC': { 'Road1': '9', 'Road2': '47' },
              'NodeD': { 'Road3': '19', 'Road4': '73' },
              'NodeE': { 'Road5': '77' },
              'NodeF': { 'Road6': '15', 'Road7': '55' },
              'NodeG': { 'Road8': '28'}
            },
 'NodeB': { ….
            ….
            }
}
```

Όπως γίνεται αντιληπτό κάθε κόμβος (NodeA) είναι πιθανό να έχει πολλά παιδιά (NodeB, NodeC, NodeD, ...) και κάθε παιδί (π.χ. NodeC) είναι πιθανό να συνδέεται με τον πατέρα με παραπάνω από έναν δρόμο (Road1, Road2) με διαφορετικά κόστη (9, 47).

Κατά την εκτέλεση του προγράμματος για κάθε βράδυ δεν επιλέχθηκε να διαμορφωθεί/κλαδεφθεί ο γράφος καθώς θεωρήθηκε χρονοβόρο και ίσως ανούσιο στο τελικό αποτέλεσμα των δύο αλγορίθμων. Όπως θα γίνει όμως προφανές παρακάτω πάρθηκαν οι κατάλληλες αποφάσεις για την επιλογή ενός δρόμου στην περίπτωση όπου πολλαπλές ακμές συνδέουν δύο κόμβους καθώς η επιλογή αυτή δεν γίνεται τυχαία.

Απληροφόρητη αναζήτηση (Breadth First Search)

Για την απληροφόρητη αναζήτηση επιλέχθηκε ο αλγόριθμος breadth first search κυρίως για την ευκολία στην υλοποίηση του η οποία βασίστηκε στους ψευδοκωδικες που παρέχει το βιβλίο (βλέπε παράρτημα 1.3) και η βικιπαίδεια (βλέπε παράρτημα 1.2). Συνοπτικά ο αλγόριθμος breadth first search ξεκινάει από έναν δοσμένο κόμβο του γράφου και εξερευνά όλους τους γειτονικούς κόμβους/παιδιά στο βάθος του εν λόγω κόμβου προτού προχωρήσει στους κόμβους στο επόμενο επίπεδο βάθους.

Το path που επιστρέφει ο αλγόριθμος *breadth first search* χρησιμοποιείται για τον υπολογισμό του κόστους μονοπατιού με βάση το predicted traffic και το actual traffic της κάθε μέρας. Όσον αφορά την επιλογή των τιμών για τα κόστη του μονοπατιού επιλέγονται οι δρόμοι με το ελάχιστο κόστος που συνδέουν δύο κόμβους, εφόσον υπάρχει παραπάνω από ένας δρόμος. Αυτό επιλέχθηκε με το σκεπτικό πως το ελάχιστο κόστος μεταξύ δύο κόμβων θα αναπαραστούσε την ιδανική περίπτωση για τον *BFS* ώστε να ήταν πιο “ανταγωνιστικός” με τον αλγοριθμο *IDA** (πράγμα που αποδείχθηκε ανούσιο καθώς ο *IDA** είχε μακράν καλύτερο αποτέλεσμα από τον *BFS*).

Πληροφορημένη αναζήτηση (IDA*)

Για την πληροφορημένη αναζήτηση ζητήθηκε η υλοποίηση του αλγορίθμου *IDA** η οποία βασίστηκε στην κατανόηση των εννοιών που εισήγαγαν οι αλγόριθμοι *Iterative Deepening DFS* (βλέπε παράρτημα 1.4) και *A** (βλέπε παράρτημα 1.5). Ο κώδικας διαμορφώθηκε με γνώμονα τον ψευδοκώδικα που παρέχει βικιπαίδεια για τον αλγόριθμο IDA* (βλέπε παράρτημα 1.6).

Συνοπτικά ο αλγόριθμος *IDA** εκτελεί αναδρομικά μια depth first search επιλέγοντας ποια κλαδιά θα απορρίψει όταν το συνολικό τους κόστος $f(n) = g(n) + h(n)$ ξεπεράσει ένα συγκεκριμένο όριο. Το όριο αυτό ξεκινά από μία εκτίμηση του κόστους για τον αρχικό κόμβο και σε κάθε αναδρομή αυξάνεται, με το καινούργιο όριο να γίνεται το ελάχιστο κόστος των κλαδιών που απορρίφθηκαν από το προηγούμενο όριο.

Όσον αφορά την συνάρτηση αξιολόγησης $f(n)$ πρακτικά εκφράζει μία εκτίμηση για το κόστος του φθηνότερου μονοπατιού για την λύση μέσω του δοθέντα κόμβου n, όπου αποτελείται από δύο επιμέρους συναρτήσεις την $g(n)$ και την $h(n)$. Η συνάρτηση $g(n)$ αναφέρεται στο πραγματικό κόστος του να φτάσουμε από τον αρχικό μας κόμβο στον δοθέντα κόμβο n και η

ευρετική συνάρτηση $h(n)$ υπολογίζει μία εκτίμηση για το κόστος του ελάχιστου μονοπατιού από τον δοθέντα κόμβο n στον τερματικό.

Για την υλοποίηση της ευρετικής συνάρτησης $h(n)$ η αρχική υπόθεση βασίστηκε στην εκτίμηση του κόστους από έναν οποιοδήποτε κόμβο n στον τερματικό μέσο της manhattan distance σε ένα καρτεσιανό επίπεδο συντεταγμένων (βλέπε παράρτημα 1.7). Η υπόθεση αυτή όμως τελικά αποδείχθηκε αρκετά χρονοβόρα και πολύπλοκη στην υλοποίηση για τον δοσμένο γράφο καθώς δεν γινόταν επιτυχώς η αναπαράσταση των κόμβων σε ένα καρτεσιανό σύστημα αφού ήταν γνωστό ότι ο καθένας ήταν πιθανό να έχει παραπάνω από 8 παιδιά. Για αυτό τον λόγο, επιλέχθηκε, αρχικά, ως ευρετική συνάρτηση να χρησιμοποιηθεί ο μέσος όρος του αθροίσματος των κοστών όλων των κόμβων του γράφου. Με αυτό τον τρόπο γίνεται μία εκτίμηση που αν και όχι έξυπνη επέτρεψε την εκτέλεση του προγράμματος. Αν και αυτό αποδείχθηκε εύκολα υλοποιήσιμο, στην πράξη οδήγησε σε ένα αρκετά αργό execution time του αλγορίθμου *IDA** και ειδικότερα στο αρχείο εισόδου *sampleGraph2.txt* στο οποίο για κάθε μέρα χρειαζόταν ≃5 λεπτά για την εξαγωγή αποτελεσμάτων, πράγμα που σήμαινε πως για την εκτέλεση και των 80 ημερών χρειαζόταν 6 ώρες!.

Συνεχίζοντας, για την ευρετική συνάρτηση λήφθηκαν υπόψη κάποιες *look-ahead* ευρετικές συναρτήσεις, οι οποίες υπολογίζουν το ελάχιστο κόστος μεταξύ των παιδιών σε επίπεδα που βρίσκονται παρακάτω από τον κόμβο n. Οι συλλογισμοί αυτοί ωστόσο, έχουν τα δικά τους προβλήματα, καθώς το ελάχιστο κόστος που βρίσκεται παρακάτω μπορεί να είναι τοπικό ελάχιστο και να παραβιάζει το κριτήριο της παραδεκτότητας, μεγαλώνοντας έτσι αχρείαστα τον αριθμό των *iterative depening* επαναλήψεων.

Τελικά, ως ευρετική συνάρτηση, υλοποιήθηκε μία συνάρτηση η οποία υπολογίζει την ελάχιστη απόσταση από τον κόμβο n μέχρι τον τερματικό κόμβο χωρίς να συμπεριλαμβάνει τα κόστη που προστίθενται από την κίνηση στους δρόμους της κάθε ημέρας. Φυσικά, το κόστος που υπολογίζει η συνάρτηση αυτή για ένα δοθέντα κόμβο, είναι μόνο μία εκτίμηση, πλησιάζει όμως αρκετά το πραγματικό με απόκλιση που εξαρτάται από την κίνηση (μεγαλύτερη για *heavy*, αρκετά μικρότερη για *low*, και καμία απόκλιση για *normal*). Επειδή οι διαδρομές στον γράφο μεταξύ των ημερών είναι πάντα οι ίδιες, μετά την κατασκευή του γράφου, υπολογίζονται τα κόστη της ευρετικής συνάρτησης για κάθε κόμβο και αποθηκεύονται σε ένα ευρετήριο για την χρήση τους στον αλγόριθμο *IDA**. Με την χρήση αυτής της ευρετικής, ο αριθμός των κόμβων στο προτεινόμενο μονοπάτι μειώθηκε κατά (περίπου) 3 - 4 κόμβους και παράλληλα ο χρόνος ολοκλήρωσης της αναζήτησης μειώθηκε εξαιρετικά (ιδιαίτερα στο αρχείο εισόδου *sampleGraph2.txt)*, και συνολικά όλη η προσομοίωση διαρκεί μόνο μερικά δευτερόλεπτα.

Όσον αφορά την λογική της συνάρτησης $g(n)$ όπως και στον αλγόριθμο *breadth first search* επιλέγεται το ελάχιστο κόστος μεταξύ των δρόμων, (αν υπάρχουν παραπάνω από ένας), για δύο κόμβους με την βασική διαφορά πως το κόστος διαμορφώνεται από την πιθανοτική κατανομή που υπάρχει για το traffic την κάθε μέρα. Πιο συγκεκριμένα, το κόστος που παρέχει η $g(n)$ εξαρτάται από τις πιθανότητες που χαρακτηρίζουν τα predictions μία δεδομένη μέρα για κάθε δρόμο και επομένως η κατανομή καθορίζει με ποιόν συντελεστή θα πολλαπλασιαστεί το *step cost* μεταξύ δύο κόμβων.

Ανάλυση εκτίμησης πιθανοτήτων και αναθεώρηση τους από μέρα σε μέρα

Οι αρχικές εκτιμήσεις των πιθανοτήτων που ανατέθηκαν για τα *predictions* ακολουθούν την κατανομή που δόθηκε στην εκφώνηση, δηλαδή p1 = 0.2 (*low traffic*), p2 = 0.2 (*normal traffic*) και p3 = 0.6 (*heavy traffic*). Οι πιθανότητες αυτές αναθεωρούνται κάθε μέρα με βάση το *actual traffic* της συγκεκριμένης μέρας καθώς και με το *actual traffic* όλων των ημερών που προηγήθηκαν.

Πιο αναλυτικά, αθροίζονται οι εμφανίσεις του κάθε είδους traffic (low, normal, heavy) για μία συγκεκριμένη μέρα και διαιρούνται με τον συνολικό αριθμό των δρόμων στο γράφο, αυτό έχει ως αποτέλεσμα κάθε μέρα να υπολογίζεται μια πιθανοτική κατανομή για κάθε είδος traffic. Στην συνέχεια αυτές οι κατανομές αθροίζονται με τις αντίστοιχες τους και διαιρούνται με τον αριθμό των ημερών που έχουν διατρέξει. Αυτό έχει ως αποτέλεσμα με το πέρασμα των ημερών να υπάρχει μία σύγκλιση προς την αληθινή κατανομή από την οποία έχουν ληφθεί τα δεδομένα για το actualTrafficPredictions.

Αξίζει να σημειωθεί πως η αρχική προσπάθεια για την αναθεώρηση των πιθανοτήτων βασιζόταν στην σύγκριση των predictions με το ActualTrafficPerDay,για μία δεδομένη μέρα, με σκοπό τον υπολογισμό της επιτυχίας (hits) του κάθε είδους prediction σε σχέση με το τι έγινε στην πραγματικότητα. Δηλαδή, για κάθε δρόμο, γινόταν ο έλεγχος εάν την συγκεκριμένη μέρα η πρόβλεψη ήταν αληθής ή όχι και ούτω καθεξής υπολογιζόταν το ποσοστό επιτυχίας για κάθε είδος traffic όπου με βάση αυτό γινόταν η αναθεώρηση των πιθανοτήτων. Το λάθος με αυτή την προσέγγιση ήταν πως δεν λαμβάνοταν υπόψη οι εκάστοτε πιθανότητες για κάθε είδος κίνησης με την έννοια πως αν παρατηρούταν ότι π.χ. η πιθανότητα του prediction “low” έπρεπε να αυξηθεί τότε αυτό σήμαινε πως κάποια άλλη πιθανότητα (είτε normal είτε heavy, πιθανώς και οι δύο) θα έπρεπε να μειωθούν.

Συμπεράσματα

Συγκρίνοντας την απόδοση των δύο αλγορίθμων για τα τρία αρχεία εισόδου και σε διάστημα 80 ημερών εξάγεται εύκολα το συμπέρασμα πως ο αλγόριθμος *IDA** είναι καλύτερος από τον *breadth first search* όσον αφορά τον αριθμό των κόμβων του μονοπατιού που θα παράγει ο καθένας και κατ’ επέκταση το κόστος ανά ημέρα αλλά και το μέσο πραγματικό κόστος ανά ημέρα. Το αποτέλεσμα αυτό είναι λογικό καθώς ο αλγόριθμος *IDA** ώντας πληροφορημένης αναζήτησης μπορεί να λαμβάνει καλύτερες αποφάσεις από τον *BFS* ειδικότερα εάν η ευρετική του συνάρτηση είναι παραδεκτή. Στο πεδίο στο οποίο ο *IDA** υστερεί συγκριτικά με τον *BFS* είναι στο *execution time* πράγμα απολύτως λογικό, καθώς ο *IDA** αν ‘χτυπήσει’ το όριο του και δεν έχει βρει την λύση θα πρέπει να ξαναρχίσει από τον αρχικό κόμβο ψάχνοντας βαθύτερα.

Όσον αφορά την ορθότητα επιλογής και αναθεώρησης των πιθανοτήτων από μέρα σε μέρα παρατηρείται πως το *predicted cost* του αλγορίθμου *IDA** καταφερνει να βρίσκεται κοντά στο πραγματικό κόστος για κάθε μέρα (μετά από αρκετές εκτελέσεις του προγράμματος δεν παρατηρήθηκε ποτέ μια απόκλιση >10 τουλάχιστον για τα 3 αρχεία εισόδου που παρείχε η εκφώνηση). Γίνεται κατανοητό πως επειδή υπάρχει αυτό το κομμάτι της αβεβαιότητας το

predicted cost δεν θα είναι ποτέ ίδιο με το πραγματικό κόστος συνεχόμενα για όλη την διάρκεια εκτέλεσης και πως έχοντας μόνο στην διάθεση μας 80 ημέρες για την αναθεώρηση των πιθανοτήτων δεν μπορούμε ποτέ να φτάσουμε στην πραγματική πιθανοτική κατανομή. Για τον αλγόριθμο *breadth first search*, αν και απληροφόρητης αναζήτησης, η απόκλιση του από το πραγματικό κόστος δεν παρατηρήθηκε να ήταν ποτέ >100, είναι όμως σημαντικά μεγαλύτερη από το αποτέλεσμα της *IDA** (μιας τάξης μεγέθους) που για το δεδομένο πρόβλημα είναι σημαντικό να ελαχιστοποιηθεί η απόκλιση μεταξύ κόστους εκτίμησης και πραγματικού κόστους όσο το μεγαλύτερο δυνατό.

Κομμάτι υλοποίησης Μέρους Β

Online search (LRTA*)

Η αναζήτηση *LRTA**, είναι μία συνάρτηση αναζήτησης η οποία εφαρμόζεται για online προβλήματα, όπου δηλαδή τα κόστη των μονοπατιών δεν είναι γνωστά εξαρχής, και η επιλογή του μονοπατιού γίνεται με βάση την ελάχιστη ευρετική τιμή σε κάθε κόμβο που είναι ήδη γνωστή. Μετά την μετάβαση από τον ένα κόμβο στον άλλο, η τιμή που αντιστοιχεί στο step-cost γίνεται γνωστή, και με βάση αυτή, γίνεται μία αναθεώρηση για την ευρετική τιμή του κόμβου από τον οποίο ξεκίνησε η μετάβαση (βλέπε παράρτημα 1.8).

Η υλοποίηση του αλγορίθμου δεν είναι ολοκληρωμένη, καθώς αντιμετωπίστηκε πρόβλημα κατά την επιλογή των suggested actions. Πιο συγκεκριμένα, επειδή μεταξύ δύο κόμβων μπορεί να υπάρχουν πολλαπλά μονοπάτια, ο αλγόριθμος πρέπει κάπως να μπορεί να επιλέγει μεταξύ όλων των προτεινόμενων action. Δεν υπάρχει κάποιο κριτήριο για την επιλογή ενός από τους πολλαπλούς δρόμους, καθώς το κόστος αυτού δεν είναι γνωστό. Εάν η επιλογή ήταν τυχαία, τότε η τιμή της ευρετικής συνάρτησης για έναν δεδομένο κόμβο, θα έπαυε να αντικατοπτρίζει την πραγματική εκτίμηση καθώς αυτή εξαρτάται από τον δρόμο που επιλέγεται. Ο κώδικας βρίσκεται σε ημιτελή κατάσταση (σε σχόλια για να μην επηρεάζεται η εκτέλεση του πρώτου μέρους), και βασίστηκε στον ψευδοκώδικα που διδάχτηκε στο μάθημα (βλέπε παράρτημα 1.8) με μία παραλλαγή αυτού για να υποστηρίζονται οι δομές δεδομένων που χρησιμοποιήθηκαν στην εργασία.

Παράρτημα 1

1.1 Python Dictionaries
1.2: Breadth-first search
1.3: Artificial Intelligence A Modern Approach 3rd Edition σελ. 82 κεφ. 3.4.1
1.4: Artificial Intelligence A Modern Approach 3rd Edition σελ. 88 κεφ. 3.4.5
1.5: Artificial Intelligence A Modern Approach 3rd Edition σελ. 93 κεφ. 3.5.2
1.6: Iterative deepening A*
1.7: Taxicab geometry
1.8: Artificial Intelligence A Modern Approach 3rd Edition σελ. 152 κεφ. 4.5.3

Παράρτημα 2

Για την εκτέλεση του προγράμματος απαιτείται η έκδοση της python 3.8.0 . Η παρακάτω οδηγίες επισυνάπτονται και σε αρχείο README.txt στο συμπιεσμένο αρχείο.

1. Navigate to the project directory .../AI-project1
2. In the terminal type: python main.py
3. From the menu navigate which file to run the program with
4. Done!

```
PS C:\Users\Alex\Desktop\Σχολη\ΠΛΗ311-Τεχνητη> cd .\AI-project1\
PS C:\Users\Alex\Desktop\Σχολη\ΠΛΗ311-Τεχνητη\AI-project1> python main.py
Choose what to do
  1) Run with sampleGraph1
  2) Run with sampleGraph2
  3) Run with sampleGraph3
  4) Run with custom file name
  5) Exit
Please enter your choice:
```

Παράρτημα 3

Παρακάτω επισυνάπτεται ένα δοκιμαστικό αρχείο εισόδου που χρησιμοποιήθηκε για την σχεδίαση και τον έλεγχο ορθότητας του προγράμματος χωρίς τα predictions (test_graph.txt).

<Source>A</Source>
<Destination>L</Destination>
<Roads>
Road1; A; B; 6
Road2; A; C; 3
Road3; A; D; 5
Road4; B; E; 3
Road5; B; F; 2
Road6; C; G; 1
Road7; D; H; 1
Road8; F; I; 4
Road9; F; J; 11
Road10; F; G; 1
Road11; G; K; 20
Road12; H; K; 10
Road13; I; L; 7
Road14; J; L; 6
Road15; K; L; 8
Road16; I; L; 9
Road17; G; H; 9
Road18; C; H; 3
Road19; C; D; 5
Road20; B; F; 1
</Roads>