Τεχνητή Νοημοσύνη
1ο Προγραμματιστικό Θέμα
Αναφορά Εργασίας

Ακ. Έτος 2018-2018

Μέλη ομάδας (αλφαβητικά)

Εμμανουήλ Παναγιώτου (03115079) – el15079@mail.ntua.gr

Μάριος Παπαχρήστου (03115101) – papachristoumarios@gmail.com

"It's going to be interesting to see how society deals with artificial intelligence, but it will definitely be cool". –Colin Angle

1 Σκοπός Εργασίας

Σε αυτή την εργασία υλοποιούμε το βασικό κορμό μιας ευφυούς υπηρεσίας εξυπηρέτησης πελατών ταξί. Συγκεκριμένα,θεωρούμε ότι υπάρχει ένας πελάτης που βρίσκεται σε μια ορισμένη τοποθεσία, ο οποίος διαθέτει κινητό τηλέφωνο με GPS και επιθυμεί να καλέσει ένα ταξί. Η υπηρεσία διαθέτει μια βάση δεδομένων με όλα τα διαθέσιμα ταξί και τη γεωγραφική θέση στην οποία βρίσκονται κάθε χρονική στιγμή, η οποία θεωρητικά θα πρέπει να ανανεώνεται συνεχώς. Η υπηρεσία θα πρέπει να εντοπίζει και να ειδοποιεί το ταξί που μπορεί να μεταβεί πιο γρήγορα στη θέση του πελάτη ώστε να τον εξυπηρετήσει. Ο χάρτης δίνεται απο δειγματοληπτημένα σημεία οδών από την πόλη της Αθήνας και ο τρόπος εύρεσης της βέλτιστης επιλογής γίνεται με τη χρήση του αλγορίθμου A*.

2 Αντικειμενοστραφής Σχεδιασμός

Η παρούσα εργασία αναπτύχθηκε σε γλώσα Java, με γνώμονα τον αντικειμενοστραφή σχεδιασμό. Χρησιμοποιήθηκαν κλάσεις για τα Ταξί (Taxi), τους Πελάτες (Client), τις Κορυφές (Node), τις ακμές (Edge), τους εκτιμητές για τον A* (Estimator) και τον συνολικό γράφο (Graph). Η κλάση (Point) απλά κρατάει μια γραμμή εισόδου του αρχείου csv. Τέλος χρησιμοποιήθηκε μια κλάση (Visual) για την παραγωγή των αρχείων KML ώστε να γίνεται η απεικόνιση των αποτελεσμάτων στον χάρτη. Για τις δομές δεδομένων χρησιμοποιήθηκαν επίσης κλάσεις των standard βιβλιοθηκών της Java.

3 Δημιουργία Γραφήματος

Η δημιουργία του γράφου γίνεται στον constructor της κλάσης (Graph). Στόχος είναι να παράξουμε μια λίστα από αντικείμενα (Node). Κάθε Node αντιστοιχεί (μέσω των atributes x, y) σε μια συντεταγμένη και περιέχει μια λίστα απο Edges (adjacent) που αντιστοιχούν στις ακμές του Node. Αρχικά διατρέχουμε

το αρχείο csv παράγοντας έτσι αντικείμενα Point. Σε κάθε συντεταγμένη υπάρχει περίπτωση (αν είναι διασταύρωση) να αντιστοιχούν πολλά αντικείμενα Point όμως θέλουμε αυτά να αντιστοιχούν σε ένα μοναδικό Node στον γράφο. Έτσι δημιουργείται η ανάγκη για δύο Hashtables (nodesMap, pointsMap) τα οποία αντιστοιχούν , ένα Node σε πολλά Points και ένα Point σε ένα Node αντίστοιχα. Επίσης για να δημιουργούμε ένα Node για κάθε συντεταγμένη κρατάμε ένα Set από όλα τα Nodes. Έτσι όταν πετύχουμε μια συντεταγμένη πολλές φορές, θα δημιουργηθέι ένα μοναδικό αντικείμενο Node για αυτήν. Έχοντας όσα χρειαζόμαστε, διατρέχουμε την λίστα όλων των Points και για κάθε Point βρίσκουμε το Node στο οποίο αυτό αντιστοιχεί (pointsMap). Έπειτα πηγαίνοντας μια θέση πίσω και μία μπροστά στην λίστα με τα Points, γνωρίζουμε ότι αυτά τα δύο είναι γείτονες του Point στο οποίο βρισκόμαστε. Οπότε ελέγχουμε αν βρίσκονται στην ίδια οδό, βρίσκουμε τα αντίστοιχα previous Node και next Node (πάλι μέσω του pointsMap), φτιάχνουμε αντικείμενα Edge από το Node που βρισκόμαστε στο previous και next, και τις προσθέτουμε στην λίστα adjacent του Node που βρισκόμαστε. Συνεχίζοντας αυτή τη διαδικασία για όλα τα δεδομένα εισόδου, φτιάχνουμε την λίστα από Nodes που χρειαζόμαστε. Κάθε Node μέσω της λίστας adjacent δείχνει στους Nodes γείτονες του. Το μήκος κάθε ακμής $\{u, v\} \in E$ τίθεται ίσο με την γεωγραφική απόσταση σε km δηλαδή

$$w(u, v) = \text{geo}(u, v) = 111.18957696 \times \arccos\left[\sin\phi_u \sin\phi_v + \cos\phi_u \cos\phi_v \cos(\theta_u - \theta_v)\right]$$

4 Αλγόριθμος A*

4.1 Περιγραφή Αλγορίθμου

Για την υλοποίηση του A* χρησιμοποιήθηκαν δύο hashtables για να κρατήσουμε τις $f(v)$ και $g(v)$. Επίσης χρησιμοποιήθηκε ένα HashSet για τη διατήρηση του κλειστού συνόλου και μια ουρά προτεραιότητας `PriorityQueue<Estimator>` στην οποία τηρούταν το μέτωπο αναζήτησης (ανοικτό σύνολο). Κάθε φορά, από το μέτωπο αναζήτησης αφαιρούνταν ο κόμβος με την ελάχιστη τιμή

$$f(v) = g(v) + h(v)$$

όπου $h(v)$ η ευριστική συνάρτηση. Αν ο v ήταν στόχος τότε η αναζήτηση τερμάτιζε. Σε κάθε άλλη περίπτωση ο αλγόριθμος έκανε relax στις κορυφές στις οποίες βελτιώνονταν το $g(v)$.

4.2 Ευριστική Συνάρτηση

Δεδομένης της γεωμετρίας της πόλης, η ευριστική συνάρτηση που διαλέξαμε είναι η Νόρμα Ταξί L_1 από τον κοντινότερο στόχο με τύπο

$$h(v) = \min_{g \in G} h_g(v)$$

όπου $G = \{g \in V \mid g \text{ στόχος}\}$ με

$$h_g(v) = \lambda \text{geo}(v, g) \qquad \lambda \leq 1$$

Στη δική μας υλοποίηση θέσαμε $\lambda = 0.9$. Για να επιστρέψει σωστά αποτελέσματα ο A* θα πρέπει η ευριστική συνάρτηση να υποεκτιμά το πραγματικό κόστος προς το στόχο.

4.3 Αποθήκευση Διαδρομών

Για να τηρήσουμε τις διαδρομές χρησιμοποιούμε ένα `Hashtable<Node, ArrayList<Pair>>` και σε κάθε relaxation προσθέτουμε στη λίστα την τιμή που βρίσκουμε (η οποία μικραίνει συνεχώς). Ο πίνακας κατακερματισμού θα χρησιμεύσει στην εκτύπωση των συντομότερων διαδρομών. Ο κατευθυνόμενος υπολειμματικός γράφος H είναι DAG. Κάνοντας μια αναζήτηση από την κορυφή–στόχο στην αρχή

μπορούμε να φτιάξουμε το γράφημα των ισοδύναμων βέλτιστων διαδρομών για τις τιμές των εκτιμήσεων της πραγματικής απόστασης του κάθε κόμβου σε κάθε χρονική στιγμή με τη βέλτιστη (συγκλίνουσα) εκτίμηση. Για τις ισοδύναμες διαδρομές έχουμε επιτρέψει κάποιο tolerance ϵ (της τάξης των 0.001km).

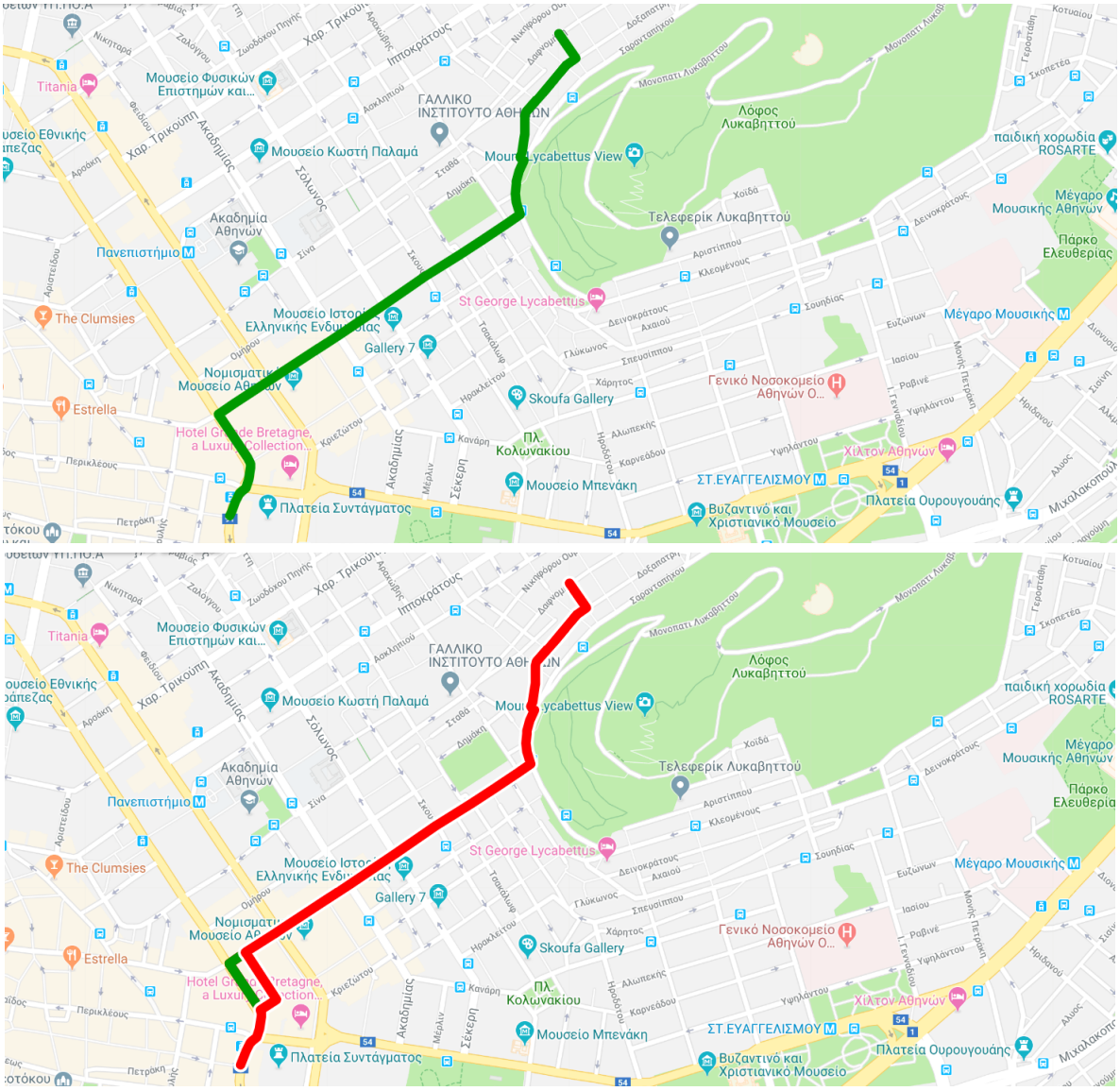
4.4 Αξιολόγηση Αλγορίθμου

Ο αλγόριθμος A* βρίσκει τη βέλτιστη λύση αν η ευριστική συνάρτηση $h(v)$ **υποεκτιμά το πραγματικό κόστος προς τον κοντινότερο στόχο–ταξί**. Ο πίνακας κατακερματισμού για τους γονείς (parent) επιτρέπει σε κάθε έναν κόμβο να έχει γονείς εντός μιας ακρίβειας ϵ από τη βέλτιστη τιμή $g(v)$. Το συγκεκριμένο πρόβλημα πληροί τις προϋποθέσεις για να προτείνει ισοδύναμες ελάχιστες διαδρομές. Θέτοντας $\epsilon = 0$ λαμβάνουμε την πραγματική ελάχιστη διαδρομή. Οι ισοδύναμες διαδρομές βρίσκονται εντός κάποιας τιμής ακρίβειας από την πραγματική βέλτιστη. Στην πράξη, οι χάρτες προτείνουν ”υπο”-βέλτιστες διαδρομές με παρόμοιο ETA.

5 Αποτελέσματα

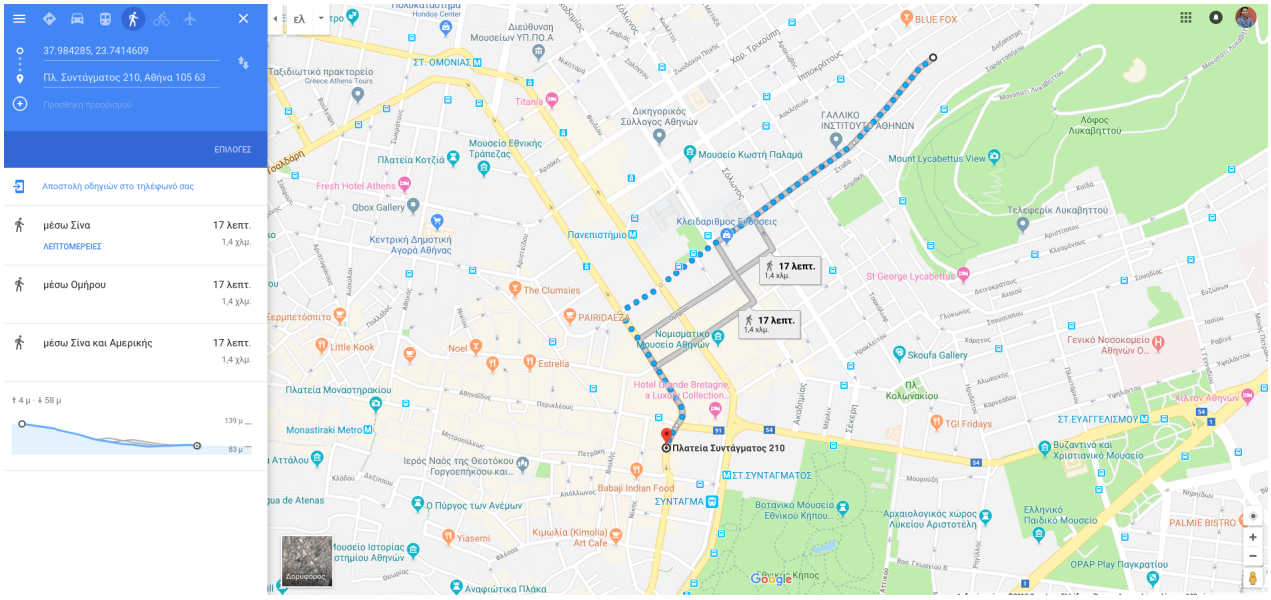
Ο χάρτης για την αποτίμηση των αποτελεσμάτων βρίσκεται εδώ: `https://drive.google.com/open?id=1YxWv5TfQ8e8Q4oBy8NvlUeZeGtOheg1Z&usp=sharing`

Το αποτέλεσμα για το παράδειγμα βρίσκεται παρακάτω



Σχήμα 1: Αποτέλεσμα για το παράδειγμα από το KML. Βέλτιστη διαδρομή (πάνω). Ισοδύναμη διαδρομή με tolerance 0.001km (κάτω)

Από το Google Maps αναζητήσαμε την ίδια διαδρομή και λάβαμε το ίδιο αποτέλεσμα



Σχήμα 2: Αποτέλεσμα για το παράδειγμα από Google Maps

Το μήκος της διαδρομής είναι 1.3696755979043929 km όπως και στην διαδρομή που προτείνει το Google Maps. Τα KML βρίσκονται στη `src/results` ενώ τα δεδομένα στην διαδρομή `resources/data`.

Αναφορές

[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,, 2016.

[2] Στάμου Γ., Τεχνητή Νοημοσύνη, Διαφάνειες