ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνητή Νοημοσύνη

Αναφορά 2ου Θέματος

ΟΜΑΔΑ

Γεώργιος-Ορέστης Χαρδούβελης, ΑΜ: 03115100

Ιάσων Λάζαρος Παπαγεωργίου, ΑΜ: 03114034

Η συγκεκριμένη άσκηση αποτελεί επέκταση της υπηρεσίας εξυπηρέτησης πελατών που δημιουργήθηκε στο 1ο Θέμα. Τα αρχεία εισόδου που μας δίνονται είναι τα εξής:

1. **client.csv:** περιέχει τις συντεταγμένες της θέσης του πελάτη όταν καλεί το ταξί, καθώς και τις συντεταγμένες προορισμού του. Επίσης, είναι αποθηκευμένη η χρονική στιγμή κατά την οποία έγινε η κλήση, η ύπαρξη αποσκευών ή όχι, ο αριθμός τους κλπ.
2. **taxis.csv:** περιέχει, για κάθε ένα από τα ταξί : το id & τη θέση του, ομοίως με την προηγούμενη εκδοχή της άσκησης, και επιπλέον χαρακτηριστικά, όπως η διαθεσιμότητα, ο τύπος του οχήματος ή το πλήθος των επιβατών που μπορεί να μεταφέρει
3. **nodes.csv:** περιέχει τις συντεταγμένες των διαφόρων σημείων των δρόμων, το id κάθε σημείου, καθώς και το id του δρόμου στον οποίο ανήκει
4. **lines.csv:** στο αρχείο αυτό βρίσκονται πληροφορίες που αφορούν τις γεωγραφικές γραμμές που υπάρχουν στο χάρτη όπως το είδος της γραμμής, η μέγιστη επιτρεπόμενη ταχύτητα (αν πρόκειται για δρόμο) κλπ.
5. **traffic.csv:** περιέχει πληροφορίες σχετικές με την κυκλοφοριακή κίνηση κάποιων δρόμων, ανά συγκεκριμένα χρονικά διαστήματα

Υποσημείωση: Στα δικά μας αρχεία αφαιρέσαμε τις στήλες με τα ελληνικά λόγω λανθασμένης αναγνώρισης του λογισμικού.

Ζητούμενα της άσκησης είναι η εύρεση της διαδρομής που τελικά θα ακολουθήσει ο πελάτης και 2 λίστες με τα πιθανά ταξί που μπορεί να χρησιμοποιήσει, ταξινομημένες ως προς κάποιο χαρακτηριστικό καθεμιά. Ο υπολογισμός των διαφόρων διαδρομών που απαρτίζουν τη λύση, γίνεται και πάλι με χρήση **της παραλλαγή του αλγορίθμου A***.

Ανάλυση λύσης

Το πρόβλημα επιλύθηκε με την ανάπτυξη προγραμμάτων στις γλώσσες Java και Prolog.

Το πρόγραμμα σε Java είναι παρόμοιο με εκείνο του πρώτου θέματος με ορισμένες παραλλαγές ώστε να συνεργάζεται με το πρόγραμμα σε Prolog.

Το πρόγραμμα σε Prolog αποτελεί ένα σύνολο κανόνων που μας βοηθούν να εξάγουμε συμπεράσματα σχετικά με την καταλληλότητα των δρόμων και των ταξί. Τα δύο προγράμματα συνεργάζονται για την εύρεση της βέλτιστης λύσης. Για την επίτευξη αυτής της συνεργασίας χρησιμοποιούμε το σύστημα JIProlog όπως προτάθηκε στην εκφώνηση.

Συγκεκριμένα, το πρόγραμμα σε Java δεν επεξεργάζεται τα δεδομένα που αφορούν τις ιδιότητες των ταξί και των γεωγραφικών γραμμών, αντιθέτως, απευθύνει ερωτήματα προς το πρόγραμμα σε Prolog για το σκοπό αυτό. Κατά τη διάρκεια της ανάγνωσης των αρχείων εισόδου, δημιουργούνται παράλληλα και κάποια κατηγορήματα σε Prolog.

Κανόνες προγράμματος Prolog

- **looktraffic:** Παράγει μια εκτίμηση του κόστους με βάση την κίνηση. Συγκεκριμένα, όταν υπάρχουν πληροφορίες για την κίνηση σχετικά με κάποιον δρόμο, αυτές δίνονται ως όρισμα στον κανόνα και ανάλογα τη χρονική στιγμή κατά την οποία πραγματοποιείται η κλήση στον δρόμο αντιστοιχίζεται μια τιμή, η οποία συνεκτιμάται κατά τον υπολογισμό του συνολικού κόστους.
- **candrive:** Στο αρχείο lines.csv περιέχονται πληροφορίες για πολλές γεωγραφικές γραμμές, από τις οποίες ορισμένες μόνο αντιστοιχούν σε δρόμους. Επομένως, ο συγκεκριμένος κανόνας δέχεται την πληροφορία για την κάθε γραμμή που υπάρχει στο αρχείο εισόδου και ελέγχει το είδος της γραμμής. Προφανώς η αποδεκτή περίπτωση είναι όταν υπάρχει κατάφαση στο χαρακτηριστικό **highway**, ενώ οι γραμμές που έχουν κατάφαση σε κάποιο από τα χαρακτηριστικά **railway**, **boundary**, **barrier**, **waterway**, **busway** αποκλείονται από τον κανόνα.
- **suitable:** Μέσω αυτού του κανόνα εξακριβώνεται η καταλληλότητα του κάθε ταξί για τη συγκεκριμένη μεταφορά. Κατά την κλήση του, δέχεται ως ορίσματα τις πληροφορίες που δίνονται για κάθε ταξί στο taxis.csv, όπως η γλώσσα που μιλά ο οδηγός, ο αριθμός των επιβατών, η ύπαρξη αποσκευών, ο αριθμός τους κλπ. Αρχικά από τον κανόνα επιβεβαιώνεται η διαθεσιμότητα του ταξί και στη συνέχεια διαπιστώνεται αν τα προαναφερθέντα χαρακτηριστικά ικανοποιούν τις ανάγκες του πελάτη.
- **roadcost:** Στη νέα εκδοχή του προβλήματος κατά την εκτέλεση της παραλλαγής του αλγορίθμου A*, είναι απαραίτητο η εκτίμηση του κόστους των επόμενων καταστάσεων να γίνεται με βάση περισσότερα κριτήρια, πέρα από την απόσταση από την αφετηρία. Τέτοια κριτήρια είναι η κίνηση, το όριο ταχύτητας, η ύπαρξη διοδίων, οι λωρίδες κυκλοφορίας κλπ. Ο συγκεκριμένος κανόνας δέχεται αυτές

τις παραμέτρους και παράγει μια εκτίμηση του κόστους, με βάση την τιμή κάθε χαρακτηριστικού.

Η υλοποίηση των παραπάνω κανόνων φαίνεται μέσα στο αρχεία Prolog.pl

Δημιουργηθέντα γεγονότα Prolog

- **direction(Id, D):** Διαμορφώνεται με βάση τις πληροφορίες για την κατεύθυνση των δρόμων που υπάρχουν στο lines.csv. Αν σε μια γραμμή με id = k, το χαρακτηριστικό oneway έχει την τιμή "yes", τότε το κατηγόρημα που αντιστοιχεί στην εν λόγω γραμμή είναι το direction(k, 1). Αν το χαρακτηριστικό έχει την τιμή -1, το κατηγόρημα γίνεται direction(k, -1), ενώ αν έχει τιμή "no" ή αν δεν έχει καθόλου τίμη, γίνεται direction(k, 0).
- **next(X1, Y1, X2, Y2, Id_l):** Χρησιμοποιώντας την κατεύθυνση της γραμμής, όπως περιγράφηκε στο παραπάνω κατηγόρημα, σε συνδυασμό με το αρχείο nodes.csv, διαμορφώνουμε κατηγορήματα next που αναφέρονται στους γείτονες του σημείου με συντεταγμένες (X1, Y1).
- **trafficcost(Id, C):** Κατηγόρημα που αναφέρεται στο κόστος κίνησης της κάθε γραμμής, όπως αυτο υπολογίζεται με χρήση του κανόνα looktraffic.
- **cost(Id, C):** Κατηγόρημα που αναφέρεται στο κόστος της κάθε γραμμής, όπως αυτό υπολογίζεται με χρήση του κανόνα roadcost.
- **valid(Id):** Κατηγόρημα που αναφέρεται στα Id των ταξί που έχουν κριθεί κατάλληλα για τις ανάγκες του πελάτη, σύμφωνα με τον κανόνα suitable.
- **rating(Id, R):** Κατηγόρημα που αναφέρεται στην αξιολόγηση η οποία συνδέεται με κάθε ταξί, η οποία δίνεται στο αρχείο εισόδου taxis.csv.

Τα παραπάνω αναφέρονται ονομαστικά και στο αρχείο Prolog.Pl

Κλάσεις προγράμματος Java

- **client, taxi, point, road, traffic:** Κλάσεις στις οποίες αποθηκεύονται οι πληροφορίες που δίνονται στα αντίστοιχα αρχεία εισόδου.
  Το πρόγραμμα σε Java, όπως έχει ήδη αναφερθεί, δεν επεξεργάζεται τα συγκεκριμένα δεδομένα, παρά τα οργανώνει σε βάση δεδομένων, την οποία παρέχει στο πρόγραμμα σε Prolog, παράλληλα με τα ερωτήματα που του διατυπώνει.
- **freader:** Η κλαση που πραγματοποιεί το διάβασμα από τα αρχεία εισόδου, ομοίως με το πρώτο θέμα.

- **Solution:** Η κύρια κλάση της λύσης, που περιέχει τη main και διαχειρίζεται τη ροή.
- **search_front:** Η κλάση που διαχειρίζεται τους κόμβους που επισκέπτεται η παραλλαγή του αλγορίθμου του A*, ομοίως με το πρώτο θέμα.
- **Astar:** Η κλάση που τρέχει την παραλλαγή του A*.

Ανάγνωση αρχείων εισόδου και οργάνωση δεδομένων στη βάση Prolog

Κατά την εκκίνηση του προγράμματος δημιουργείται στη Solution μια οντότητα freader η οποία πραγματοποιεί το διάβασμα από τα αρχεία εισόδου. Τα δεδομένα που αποθηκεύονται αφορούν τον πελάτη, τα ταξί, τα χαρακτηριστικά και την κίνηση των δρόμων και τους κόμβους. Παράλληλα δημιουργούνται και τα διάφορα κατηγορήματα.

Για όσους δρόμους έχουμε πληροφορία κίνησης, προσθέτουμε κατηγορήματα trafficcost στη βάση. Οι διάφορες γεωγραφικές γραμμές ελέγχονται μέσω του κανόνα candrive κι επιλέγονται μόνο αυτές που αναφέρονται σε δρόμους. Για αυτές δημιουργούμε τα αντίστοιχα κατηγορήματα direction, υπολογίζουμε το κόστος τους μέσω του κανόνα roadcost και προσθέτουμε και τα αντίστοιχα κατηγορήματα cost. Κάνουμε τις απαραίτητες παραδοχές, θεωρώντας αυθαίρετες τιμές για όσα χαρακτηριστικά των δρόμων δεν έχουμε πληροφορία· η μέγιστη ταχύτητα εκτιμάται ανάλογα με το είδος του δρόμου, η μη αναφορά σε διόδια εκλαμβάνεται ως απουσία τους κλπ.

Κατόπιν, στη βάση προστίθενται τα κατηγορήματα τύπου next, χρησιμοποιώντας τις πληροφορίες που προκύπτουν από την κατεύθυνση κάθε γραμμης.

Τέλος, κατά την ανάγνωση των δεδομένων που αφορούν τα ταξί εξετάζεται ποιά απο αυτά είναι κατάλληλα για τη συγκεκριμένη περίσταση με χρήση του κανόνα suitable και προστίθενται στη βάση τα αντίστοιχα κατηγορήματα τύπου valid.

Ροή προγράμματος κατά την εκτέλεση της παραλλαγής του A*

Η παραλλαγή του αλγορίθμου του A* υλοποιείται με τον ίδιο τρόπο όπως και στο πρώτο θέμα. Αρχικά κανονικοποιούμε τη θέση του πελάτη (η οποία όπως έχει αναφερθεί δεν είναι υποχρεωτικό να ταυτίζεται με κάποιο δοθέν σημείο), επιλέγοντας το σημείο από το αρχείο nodes.csv που απέχει απ' αυτήν τη μικρότερη απόσταση. Η ίδια διαδικασία επαναλαμβάνεται και για τον προορισμό του πελάτη.

Κατόπιν, τα ταξί διατάσσονται με κριτήριο την ευκλείδεια απόστασή τους από τον πελάτη.

Η παραλλαγή του A* εκτελείται επαναληπτικά για κάθε ταξί που αντιστοιχίζεται σε αληθές κατηγόρημα valid(Id). Το κοντινότερο σημείο στη θέση του ταξί επιλέγεται και τοποθετείται στο μέτωπο αναζήτησης του A*.

Αντίθετα από το πρώτο θέμα, αποφεύγουμε ακμές που αντιστοιχούν σε ατέρμονους βρόχους με χρήση hashmap.

Ακόμη, και οι γειτονικοί κόμβοι προσδιορίζονται διαφορετικά από ότι στην προηγούμενη άσκηση. Συγκεκριμένα απευθύνουν ερώτημα στο πρόγραμμα Prolog δεδομένων των κατηγορημάτων της μορφής next(X1, Y1, X2, Y2, Id_l).

Μια ακόμη διαφορά έγκειται στον τρόπο υπολογισμού της απόστασης με τους γειτονικούς κόμβους. Αυτή στην προκειμένη περίπτωση υπολογίζεται συνεκτιμώντας το κόστος της γραμμής στην οποία ανήκει, όπως αυτό δηλώνεται στο κατηγόρημα cost(Id_l, C). Η συνάρτηση h -ευριστική- είναι και σε αυτή την περίπτωση η ευκλείδεια απόσταση του κάθε κόμβου από τον στόχο.

Μόλις ολοκληρωθεί η παραλλαγή του A* για το συγκεκριμένο ταξί, υπολογίζεται η συνολική του απόσταση από τον πελάτη με τη βοήθεια της συνάρτησης caclulate_f(), που έχει προστεθεί στην κλάση search_front. Το ζεύγος (id_ταξί, απόσταση) προστίθεται σε ένα ArrayList, το οποίο μετά την εξέταση όλων των ταξί ταξινομείται με βάση την απόσταση και τα πρώτα στοιχεία του (μέχρι 5) εμφανίζονται στον πελάτη.

Τα ταξί αυτά εμφανίζονται ακόμη μια φορά, ταξινομημένα με βάση τη βαθμολογία που έχουν λάβει από τους πελάτες. Ο πελάτης εισάγει το Id του ταξί που επέλεξε και πάλι με χρήση της παραλλαγής του αλγορίθμου A* βρίσκουμε τη βέλτιστη διαδρομή που πρέπει να ακολουθήσει το ταξί από τη στιγμή που θα τον παραλάβει μέχρι τον προορισμό του.

Αποτελέσματα

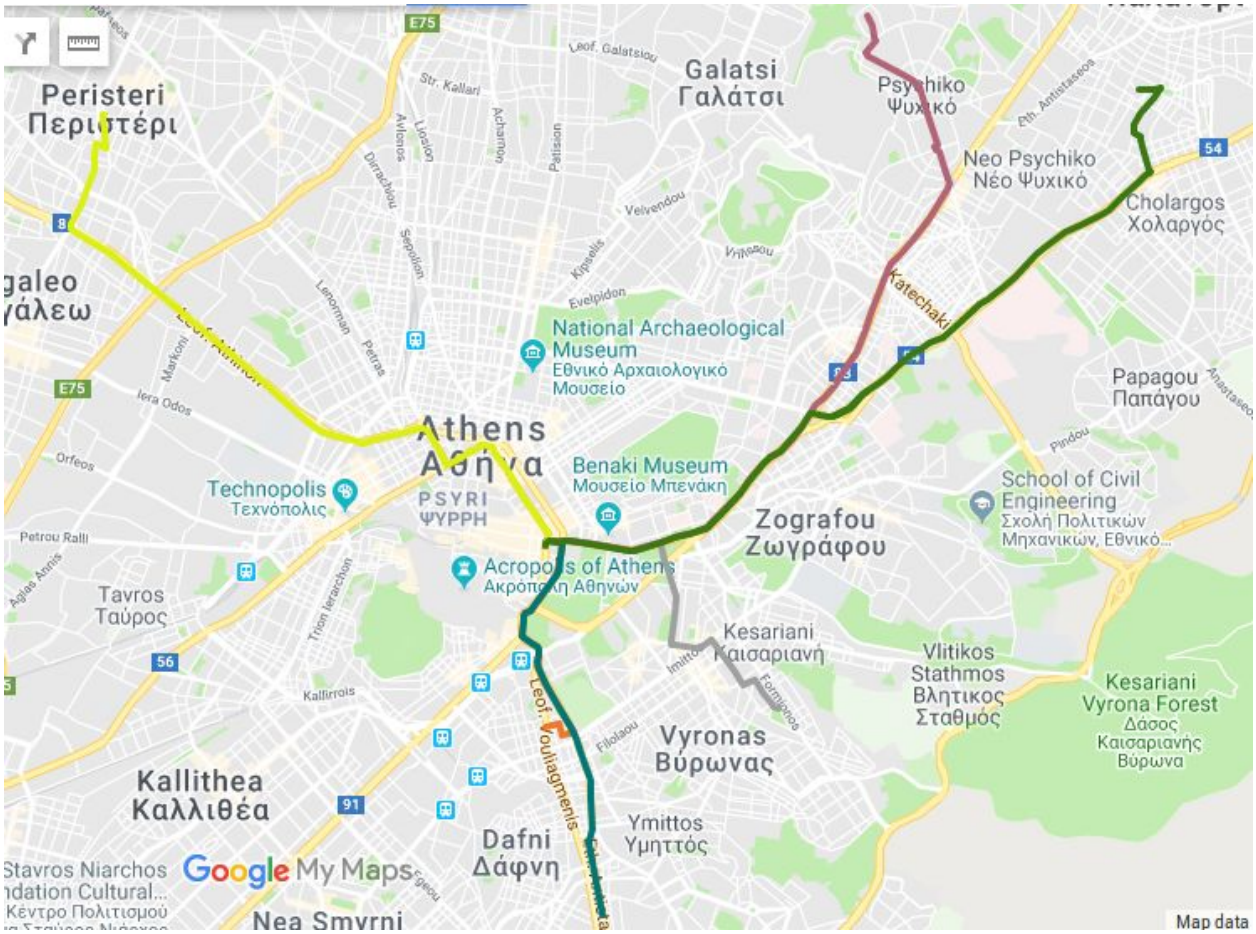
- **Δοθέν παράδειγμα**

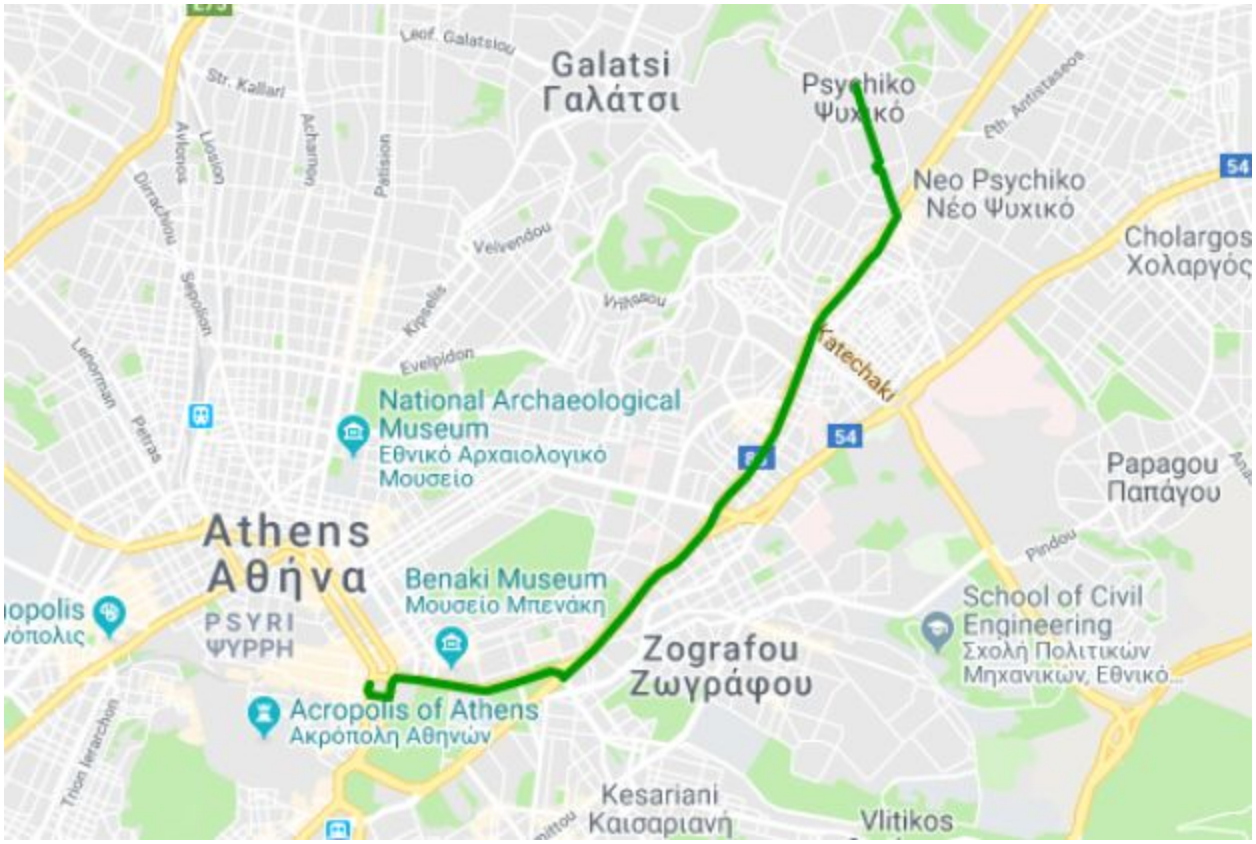
```
Hey you.
These 5 taxis are reaaaaady to ruuuumble !
1. Taxi with id = 150 has distance from you = 0.202489
2. Taxi with id = 160 has distance from you = 0.273928
3. Taxi with id = 120 has distance from you = 0.311302
4. Taxi with id = 190 has distance from you = 0.595420
5. Taxi with id = 140 has distance from you = 0.628103

You wanna rating order? No problemo Bob !
```

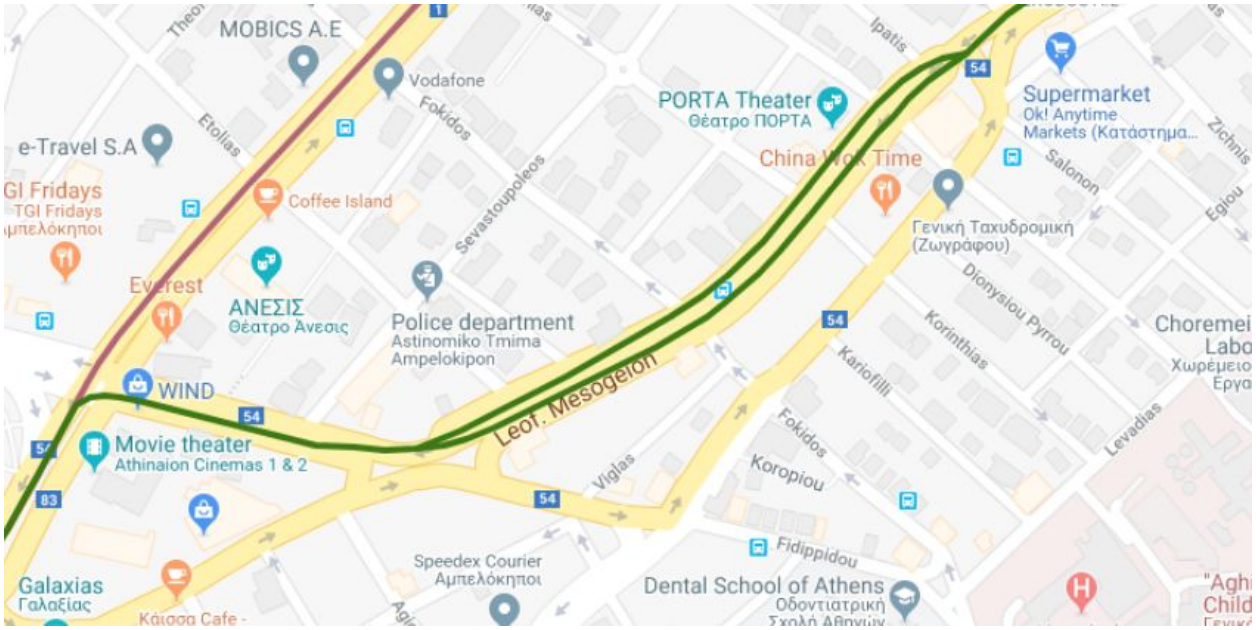
Τα kml αρχεία για την απεικόνιση των διαδρομών όλων των ταξί και της διαδρομής από τον πελάτη προς τον προορισμό του βρίσκονται στα αρχεία ask2 all routes.kml και ask2 client.kml αντίστοιχα.

Εδώ φαίνονται και φωτογραφίες από τους χάρτες αντίστοιχα:





Στην διαδρομή από το ταξί προς τον πελάτη με id 200 βλέπουμε πως ο αλγόριθμος μας βρίσκει 3 εναλλακτικά μονοπάτια για να φτάσει μέχρι τον πελάτη όπως φαίνεται παρακάτω



- **Δικό μας παράδειγμα**

(αρχεία client2.csv και taxis.csv έναντι των client.csv και taxis.csv)

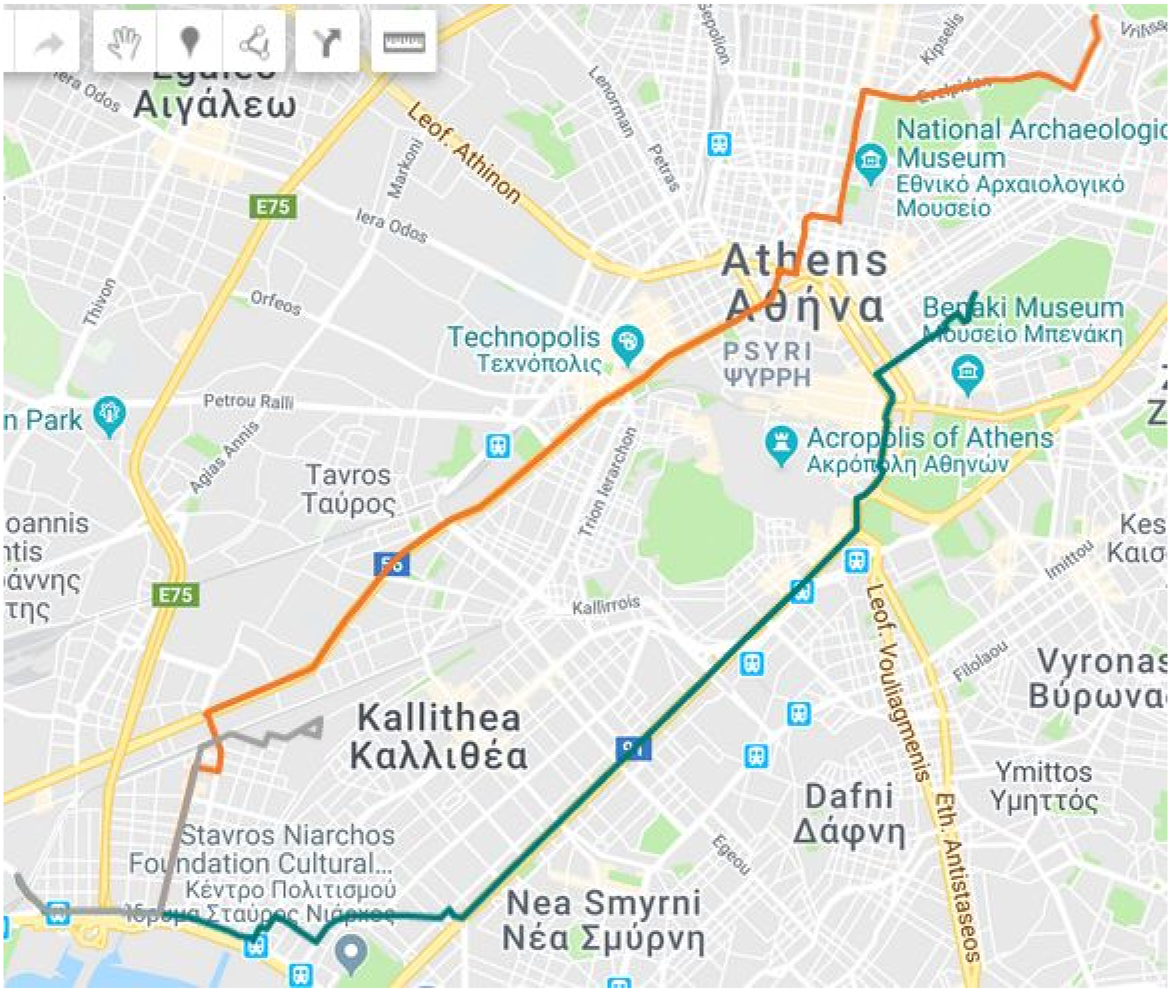
```
Hey you.
These 3 taxis are reaaaaady to ruuuumble !
1. Taxi with id = 250 has distance from you = 0.278662
2. Taxi with id = 220 has distance from you = 0.737392
3. Taxi with id = 230 has distance from you = 0.867183

You wanna rating order? No problemo Bob !
1. Taxi with id = 230 and rating = 9.200000
2. Taxi with id = 220 and rating = 8.000000
3. Taxi with id = 250 and rating = 7.100000
Gimme the id of the one that you want, that you really really want.
220
Kind of a weird choice but whatevs... Taxi with id = 220 is on it's way
```

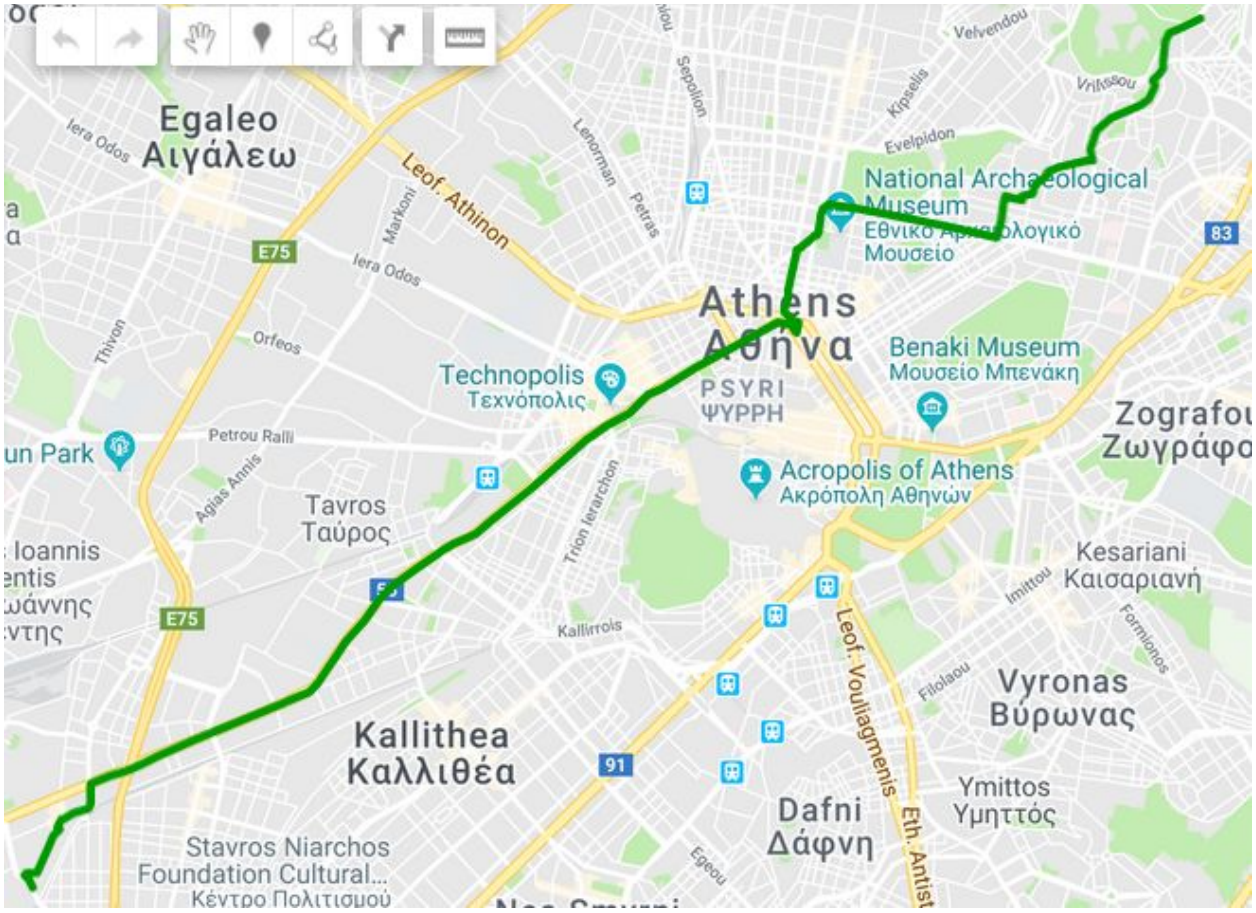
Τα csv αρχεία που αλλάξαμε και αφορούν την θέση του πελάτη και τον ταξί είναι τα client2.csv και taxis2.csv αντίστοιχα.

Τα kml αρχεία για την απεικόνιση των διαδρομών όλων των ταξί και της διαδρομής από τον πελάτη προς τον προορισμό του βρίσκονται στα αρχεία ask2_2 all routes.kml και ask2_2 client.kml αντίστοιχα.

Εδώ φαίνονται και φωτογραφίες από τους χάρτες:



Αιγάλεω
Leof. Athinon
Iera Odos
Orfeos
Petrou Ralli
Agias Annis
Thivon
E75
Markoni
Lenorman
Petras
Sepolion
Kipselis
Evelpidon
National Archaeological Museum
Εθνικό Αρχαιολογικό Μουσείο
Athens
Αθήνα
Technopolis
Τεχνόπολις
PSYRI
ΨΥΡΡΗ
Benaki Museum
Μουσείο Μπενάκη
Acropolis of Athens
Ακρόπολη Αθηνών
Tavros
Ταύρος
Trion Ierarchon
Kallirrois
Imittou
Filolaou
Leof. Vouliagmenis
Eth. Antistaseos
Vyronas
Kallithea
Καλλιθέα
Dafni
Δάφνη
Ymittos
Υμηττός
Egeou
Stavros Niarchos Foundation Cultural...
Κέντρο Πολιτισμού Ίδρυμα Σταύρος Νιάρχος
Nea Smyrni
Νέα Σμύρνη



Εδώ ο αλγόριθμος δεν εμφανίζει κάποιο εναλλακτικό μονοπάτι, σε αντίθεση με το πρώτο θέμα. Αυτό οφείλεται στην προσθήκη όλων των παραμέτρων μέσω της Prolog τα οποία κατέστησαν την ακρίβεια της αναζήτησης της διαδρομής πολύ ακριβέστερη.

Links για τους χάρτες:
Taxi to Client δοθέν παράδειγμα
Client to Destination δοθέν παράδειγμα
Taxi to Client δικό μας παράδειγμα
Client to Destination δικό μας παράδειγμα