Δημήτρης Κελέσης A.M.: 03115037

Γενικός σχεδιασμός Συστήματος

Για την συγκεκριμένη εργασία δημιουργήθηκε ένα σύστημα που επιτυγχάνει επικοινωνία μεταξύ ενός προγράμματος σε Prolog που έχει ουσιαστικά τον ρόλο μίας βάσης δεδομένων στην οποία αποθηκεύεται ολόκληρος ο κόσμος της εργασίας. Αρχικά διαβάζουμε με παρόμοιο τρόπο τα δεδομένα από τα αρχεία csv και δημιουργούμε Strings που αντιστοιχούν στους κανόνες που θέλουμε να εισάγουμε στη βάση. Δηλαδή για τον κανόνα με γενική μορφή something(A,B,C) θα δημιουργήσουμε για κάθε τέτοια εισαγωγή το αντίστοιχο something(1,2,3). Αφού το κάνουμε αυτό για κάθε στοιχείο που ανήκει στον συγκεκριμένο κανόνα εισάγουμε αυτή την πληροφορία στην βάση που αντιπροσωπεύει η Prolog. Έτσι πλέον όλη η γνώση για το κόσμο μπορεί να είναι προσβάσιμη μόνο μέσω της Prolog και αξιοποιώντας επιπλέον κανόνες που θα θέσουμε. Σημειώνεται ότι στην παραπάνω βάση αποθηκεύονται όλοι οι κόμβοι, όλες οι γραμμές, η κίνηση (όπως αυτή δίδεται από το αντίστοιχο αρχείο traffic), τα στοιχεία του πελάτη και τέλος τα στοιχεία για όλα τα ταξί που υπάρχουν. Για την εν λόγω αποθήκευση διαβάζουμε τα αρχεία και ξεχωρίζουμε το κάθε χαρακτηριστικό για κάθε οντότητα και δημιουργούμε ουσιαστικά αυτή την οντότητα. Θα μπορούσαμε να παραλείψουμε κάποια στοιχεία για κάποιες οντότητες τα οποία όπως θα φανεί παρακάτω δεν αξιοποιήσαμε. Αφού δημιουργήσαμε πλέον στην μηχανή της Prolog όλα τα δεδομένα σε μορφή κατάλληλη για επεξεργασία από την γλώσσα μένει να δημιουργήσουμε ένα σύνολο κανόνων σύμφωνα με τους οποίους θα τα επεξεργαζόμαστε και θα εξάγουμε τα αποτελέσματα που μας ενδιαφέρουν για την υλοποίηση του A*. Δηλαδή απαιτούμε από αυτούς τους κανόνες να συμπεράνουμε αν μπορούμε να κινηθούμε από έναν κόμβο σε κάποιον άλλον, να βρούμε το βάρος/κόστος που έχει η παραπάνω μετακίνηση και να ελέγξουμε αν το εκάστοτε ταξί είναι επαρκές για τον πελάτη που καλούμαστε να εξυπηρετήσουμε. Για να αξιοποιήσουμε τα δεδομένα μας επιλέγουμε αρχικά να εντοπίσουμε τον τύπο του δρόμου στον οποίο ανήκει ο τρέχον κόμβος μας και αναλόγως αυτού του τύπου αυθαίρετα δίνουμε τιμές κόστους θεωρώντας ότι οι πιο μεγάλοι δρόμοι είναι πιο γρήγοροι και άρα θα έχουν μικρότερο κόστος αφού θα μπορεί το ταξί να αναπτύξει μεγαλύτερη ταχύτητα σε σχέση με τους δρόμους εντός κατοικημένης περιοχής για παράδειγμα. Ένα άλλο κριτήριο που χρησιμοποιήθηκε είναι αυτό της ύπαρξης ή μη διοδίων, αν αυτά υπάρχουν αντίστοιχα απαιτείται περισσότερος χρόνος στην μετακίνηση λόγω δημιουργίας κίνησης είτε καθυστέρησης για να πληρωθούν. Επίσης αξιοποιήθηκαν τα όρια ταχύτητας στους δρόμους καθώς επηρεάζουν άμεσα το ποιο ταξί θα φτάσει πρώτο. Η κίνηση λήφθηκε υπόψη για τον κάθε δρόμο τον οποίο διασχίζει το ταξί καθώς επίσης και το πλήθος των λωρίδων που υπάρχουν σε αυτό τον δρόμο. Δίνοντας σύμφωνα με τα παραπάνω διάφορες τιμές που ποιοτικά πλησιάζουν την πραγματικότητα και πολλαπλασιάζοντας αυτά τα βάρη μεταξύ τους δημιουργούμε το συνολικό βάρος για τη μετακίνηση από έναν κόμβο σε έναν άλλον. Συγκεκριμένα, για δρόμο με μεγάλο όριο ταχύτητας δίδεται μικρό αντίστοιχο κόστος, για δρόμο με αρκετές λωρίδες κυκλοφορίας ή για δρόμο με λίγη κίνηση ή για δρόμο εκτός αστικού ιστού ή για δρόμο χωρίς διόδια όμοια. Αν ισχύει κάτι αντίθετο από τα παραπάνω τότε αυξάνεται το αντίστοιχο κόστος, το συνολικό κόστος προκύπτει από το γινόμενο των επιμέρους, σύμφωνα με τα χαρακτηριστικά του δρόμου. Επιπλέον, κριτήρια που θα μπορούσαν

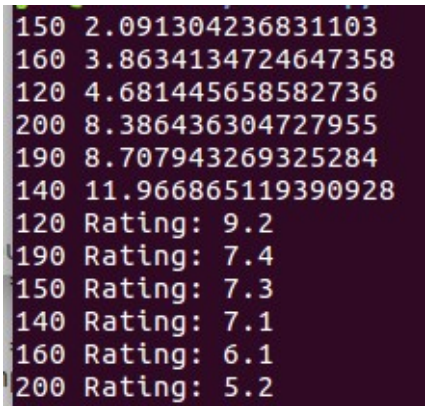
να χρησιμοποιηθούν θα ήταν αυτό της ύπαρξης φωτισμού αν η διαδρομή γινόταν στην διάρκεια της νύχτας, η ύπαρξη εμποδίων, φυσικών ή και μη καθώς και η ύπαρξη ή μη γεφυρών/τούνελ που θα αποτελούσαν συντομεύσεις για την μετακίνηση μας. Προφανώς αν είχαν χρησιμοποιηθεί και τα παραπάνω κριτήρια θα δίναμε αντίστοιχες τιμές ώστε να υπάρχει μεγαλύτερη και μικρότερη ποινή αναλόγως της επιλογής του αλγορίθμου για μετακίνηση σε κάποιον κόμβο. Για λόγους απλότητας των κανόνων της Prolog δεν έγινε περαιτέρω ανάπτυξη κριτηρίων. Αν είχαν αναπτυχθεί όλα τα κριτήρια για την επιλογή μελλοντικού κόμβου και τα δεδομένα μας ήταν πλήρη τότε τα αποτελέσματα που θα παρήγαγε το πρόγραμμα μας θα πλησίαζαν περισσότερο τα πραγματικά.
Πέραν των κριτηρίων για τις διαδρομές που θα μπορέσει να κάνει το εκάστοτε ταξί για να φτάσει στον πελάτη θα πρέπει να ληφθούν υπόψη και τα δεδομένα που μας δίδονται για τον πελάτη και το πως αυτά αλληλεπιδρούν με τα δεδομένα για τα ταξί. Για τον λόγο αυτό θα πρέπει να εξετάσουμε ποια ταξί είναι κατάλληλα για τον κάθε πελάτη, για να το κάνουμε αυτό ελέγχουμε κάποια βασικά χαρακτηριστικά του πελάτη και του ταξί και του οδηγού αυτού. Ειδικότερα ελέγχουμε για κοινή γλώσσα μεταξύ των δύο, αν το ταξί είναι διαθέσιμο, αν το πλήθος των πελατών χωράει στο ταξί καθώς και αν το πλήθος των αποσκευών είναι τέτοιο ώστε να χωρέσει στον αποθηκευτικό χώρο του οχήματος, σύμφωνα και πάλι με αυθαίρετα κριτήρια θεωρούμε ότι για περισσότερες από δύο βαλίτσες απαιτείται μεγάλο όχημα.
Η χρήση της Prolog γίνεται με τα προτεινόμενα εργαλεία και για οποιαδήποτε πληροφορία χρειάζεται το κυρίως πρόγραμμα γίνονται αιτήσεις στην Prolog που επιλύοντας τα αντίστοιχα λεκτικά επιστρέφουν τις ζητούμενες απαντήσεις. Και ταυτόχρονα σε κάποιες περιπτώσεις εισάγουν νέα δεδομένα στην βάση γνώσης.

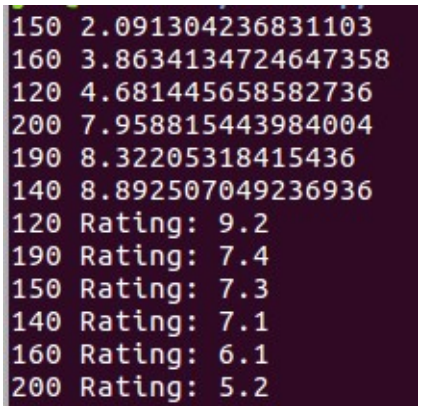
Για τα κατηγορήματα και το τι αντιπροσωπεύει το καθένα έχουμε τα εξής:
client(x, y, x-destination, y-destination, time, people, language, luggage) = Τα στοιχεία του πελάτη σε ολόκληρη την γραμμή του αρχείου εισόδου όπως αυτό διαβάζεται, δηλαδή η θέση του, η επιθυμητή θέση μετάβασης, ο χρόνος, το πλήθος των ατόμων που θα πρέπει να μεταφερθούν, η γλώσσα που μιλά και το πλήθος των αποσκευών του.
taxi(x, y, id, available, rating, longdist, type) = Τα στοιχεία του ταξί όπως αυτά διαβάστηκαν, η θέση του, η διαθεσιμότητα του, η βαθμολογία του οδηγού του, η δυνατότητα του για μεγάλες διαδρομές και ο τύπος του.
lineTraffic(trafid, stime, ftime, traflevel) = Τα στοιχεία για την κίνηση όπως αυτά διαβάστηκαν, το αναγνωριστικό της κίνησης, η ώρα έναρξης και λήξης καθώς και το επίπεδο αυτής.
node(x, y, lineid) = Για κάθε κόμβο του γραφήματος τις συντεταγμένες του και την γραμμή στην οποία ανήκει.
lineSpecs(id, highway, oneway, lit, lanes, maxspeed, railway, boundary, access, natural, barrier, tunnel, bridge, incline, waterway, busway, toll) = Τα στοιχεία για κάθε γραμμή όπως αυτά δίνονται από την εκφώνηση με μοναδική παράληψη το όνομα αυτής καθώς είναι στα ελληνικά και δεν θεωρήθηκε αναγκαία η μοντελοποίηση του στην Prolog.
canMoveFromTo(Ax, Ay, Bx, By) = Αν είναι δυνατή η μετακίνηση από το σημείο (Ax, Ay) στο σημείο (Bx, By).
weightFactor(Ax, Ay, Bx, By, Value) = Αν μπορεί να γίνει η παραπάνω μετακίνηση τότε ποιο είναι το κόστος της, εξαρτάται από τις διάφορες παραδοχές που κάναμε στην ευριστική συνάρτηση αξιοποιώντας τα έξτρα δεδομένα για τους δρόμους, όπως η κίνηση ο φωτισμός κλπ.
isQualifiedDriverForClient(driverid) = Αν ο συγκεκριμένος οδηγός με το driverid είναι κατάλληλος για να μεταφέρει τον προς εξέταση πελάτη στον προορισμό του.

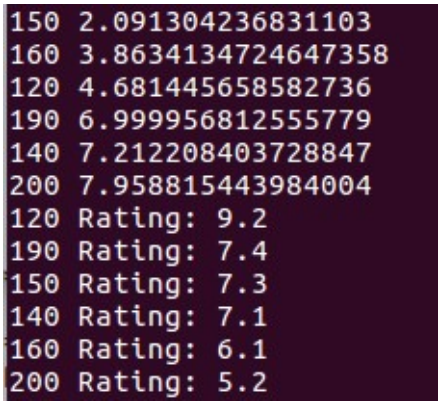
driverRank(driverid) = Rating του οδηγού με το αναφερθέν driverid.
speaksDriver(driverid, language) = Η γλώσσα που μιλά ο οδηγός του ταξί.
maxPassengers(driverid, number) = Το μέγιστο πλήθος επιβατών που μπορούν να χωρέσουν στο ταξί με driverid το παραπάνω.
score(A, 1) = Καθυστέρηση που αντιστοιχεί σε δρόμο τύπου Α.
highwayRank(lineid, s) = Καθυστέρηση σύμφωνα με τον τύπο της γραμμής.
highwayRank1(lineid, s) = Καθυστέρηση σύμφωνα με το αν υπάρχουν διόδια στη γραμμή.
highwayRank2(lineid, s) = Καθυστέρηση σύμφωνα με την μέγιστη επιτρεπτή ταχύτητα της γραμμής.
highwayRank3(lineid, s) = Καθυστέρηση σύμφωνα με το πλήθος λωρίδων της γραμμής.
trafficRank(lineid, s) = Καθυστέρηση σύμφωνα με την ώρα και την υπάρχουσα κίνηση στη γραμμή.
luggugeIn(number, driverid) = Αν έχω περισσότερες από δύο βαλίτσες απαιτώ large όχημα αλλιώς οποιοδήποτε και επιστρέφω το driverid του αντίστοιχου οχήματος που θα βρω.

Για την ανάπτυξη του προγράμματος σε Java αρχικά έχουμε μία γενική κλάση θέσης που περιέχει τις συντεταγμένες και από αυτήν κληρονομούν οι κλάσεις που αντιπροσωπεύουν τους πελάτες, τα ταξί και τους κόμβους που θα δημιουργήσουμε για το γράφημα πάνω στο οποίο θα τρέξουμε τον A*. Το αξιοσημείωτο για αυτή την κλάση είναι η ύπαρξη μεθόδου που υπολογίζει την απόσταση δύο θέσεων πάνω σε σφαίρα κάνοντας χρήση της Haversine μεθόδου σύμφωνα με την ακτίνα της Γης και της συντεταγμένες στο επίπεδο των δύο σημείων, συγκεκριμένα υπολογίζεται η απόσταση του τρέχοντας σημείου από το οποίο την καλούμε από κάποιο άλλο σημείο. Για τα ταξί, τις γραμμές, την κίνηση, τον πελάτη και τους κόμβους έχοντας διαβάσει τα δεδομένα τους από το αρχείο εισόδου κυρίως δημιουργούμε αντίστοιχα queries για την Prolog με τα δεδομένα τους και για κάποια από αυτά ένα instance για το κάθε ένα που περιέχει τη θέση και την ταυτότητα τους. Ουσιαστικά με τα παραπάνω δημιουργούνται αυτά τα δεδομένα που περιγράψαμε στην αρχή που θα χρειαστεί η Prolog. Γίνεται επίσης χρήση της κλάσης που υποδηλώνει την διαδρομή που θα ακολουθήσουμε και περιέχει μία λίστα από κόμβους που αυτή θα διασχίσει, σε τέτοια μορφή θα είναι αποθηκευμένοι οι τελικοί μας κόμβοι στη λύση την οποία θα παρουσιάσουμε. Η διαδρομή περιέχει πέραν της λίστας το κόστος της καθώς επίσης και το ταξί το οποίο θα την πραγματοποιήσει. Για τους κόμβους που αναφέρθηκαν πιο πάνω δημιουργείται ένα γράφημα προσθέτοντας κατάλληλες ακμές μεταξύ τους ώστε να μπορέσουμε να τρέξουμε τον A*, για κάθε κόμβο καθορίζουμε τους γείτονες του το αν αυτός αποτελεί κόμβο στόχο την απόσταση του από τον στόχο, όπως αυτή υπολογίζεται από την συνάρτηση Haversine. Για τις παραπάνω ακμές μεταξύ των κόμβων καθορίζουμε το βάρος κάθε ακμής και την αντίστοιχη τιμή της ευριστικής. Επίσης κάνουμε override τις μεθόδους για την συγκρίσεις μεταξύ των ταξί και των διαδρομών ώστε να δίνουν πραγματικά αποτελέσματα. Τέλος τροποποιώντας την σύγκριση μεταξύ των κόμβων ώστε να ανταποκρίνεται σε εκείνη που απαιτεί ο A* δηλαδή το άθροισμα της απόστασης που έχουμε ήδη διανύσει και της ευριστικής που μένει μέχρι τον στόχο, τρέχουμε τον αλγόριθμο όπως αυτός περιγράφεται θεωρητικά σε διάφορες βιβλιογραφίες. Για κάθε κόμβο βρίσκουμε τους επόμενους του και συνεπώς έτσι υπολογίζουμε για αρχικό και τελικό κόμβο την αντίστοιχη διαδρομή, εδώ τελικός κόμβος είναι ο πελάτης.
Σε σχέση με την πρώτη εργασία αλλάχτηκε αρκετά ο τρόπος εκτέλεσης του A* καθώς σε αυτή την εκδοχή θα πρέπει να αντλεί τα δεδομένα όχι απευθείας από την είσοδο αλλά από την βάση, συνεπώς αν και σε γενικές γραμμές η ύπαρξη κόμβων και ακμών πραγματοποιήθηκε με παρόμοιο τρόπο με την πρώτη εργασία η διαχείριση των δεδομένων και ο συνδυασμός τους για την πραγματοποίηση του αλγορίθμου άλλαξε. Τα δεδομένα τα παίρναμε από την βάση κάθε

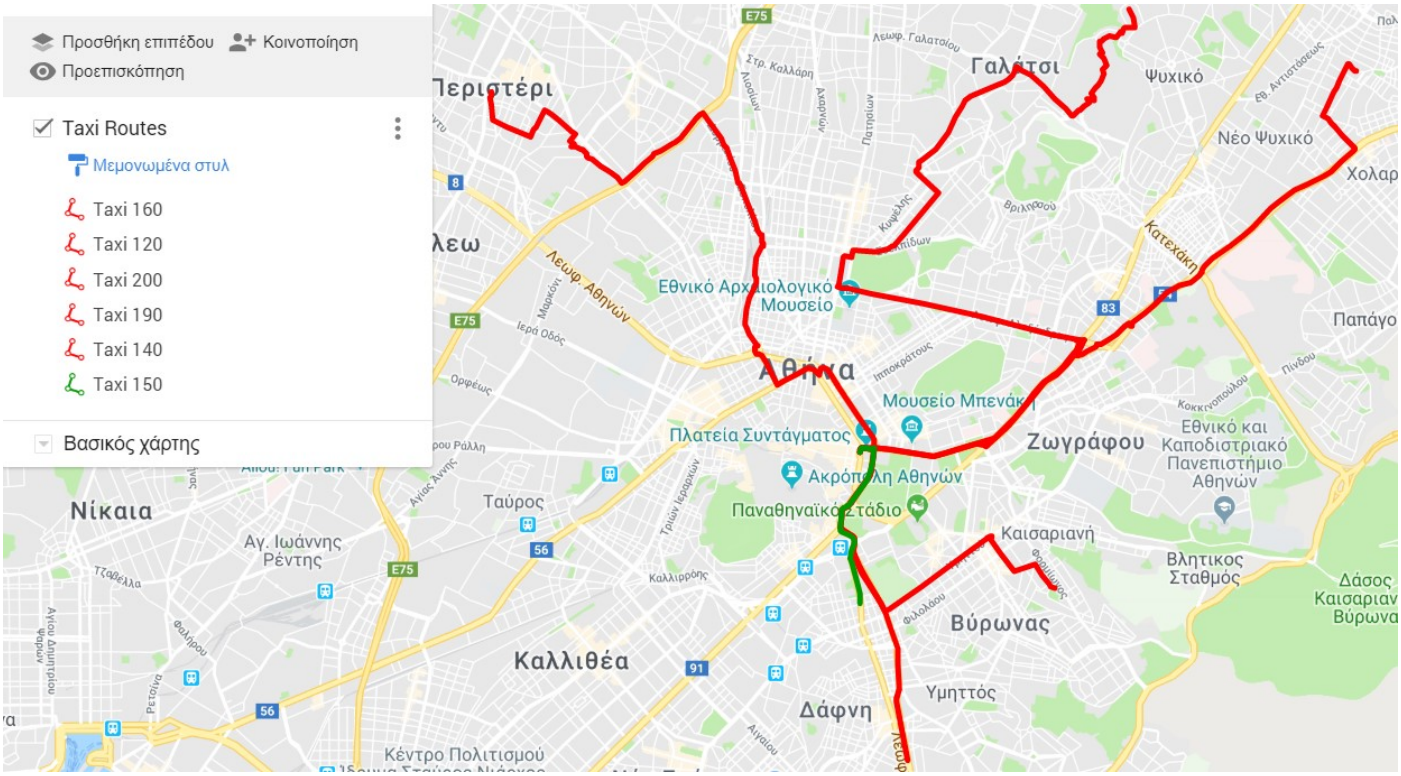
φορά και τα αξιοποιούμε για να κάνουμε τους εκάστοτε υπολογισμούς. Ο τρόπος ανάγνωσης τους παρέμεινε ο ίδιος με την διαφορά ότι αυτά αντί να αποθηκευτούν απευθείας στο κυρίως πρόγραμμα αποθηκεύονται στην βάση. Επίσης άλλαξε ο τρόπος που ορίζουμε το μέγεθος του μετώπου αναζήτησης που αυτό καθορίζεται και επηρεάζει την ακρίβεια και την ταχύτητα του A*. Επίσης τροποποιήθηκε η μέθοδος αναζήτησης νέων καταστάσεων και εύρεσης των πιο κοντινών μελλοντικών καταστάσεων, ώστε να μπορεί να συμπεριλάβει τα νέα στοιχεία αποτελεσματικά και να αξιοποιήσει τις λογικές συνεπαγωγές της Prolog. Ακόμη προστέθηκε η ύπαρξη κλάσης διαδρομής αφού μιλάμε πλέον για μεγάλο πλήθος διαδρομών και θα πρέπει να μπορούμε να τις διατρέχουμε μία προς μία, ανεξάρτητα του ότι στην έξοδο εμφανίζουμε τις καλύτερες. Για να βρούμε μία διαδρομή εφαρμόζουμε τον A* και για να έχουμε δύο διαφορετικές κατατάξεις επιλέγουμε σαν δεύτερο κριτήριο πέραν της συντομότερης διαδρομής την βαθμολογία του κάθε ταξί καθώς αυτό αποτελεί εύλογη διαφοροποίηση. Τέλος το πρόγραμμα παράγει μόνο του το αρχείο kml απευθείας και δεν πρέπει ο χρήστης να το κάνει χειροκίνητα έχοντας τις συντεταγμένες.
Τελικά έχοντας τρέξει τον A* παράγουμε αποτελέσματα για διαδρομές των ταξί και θέσης του πελάτη. Από αυτά κρατάμε τα καλύτερα (στην αρχική προσέγγιση θα κρατούσαμε τα δύο καλύτερα αλλά για τις εικόνες που ακολουθούν τα κρατάμε όλα) για κάθε τιμή μετώπου και αρχείων εισόδου, τα οποία απεικονίζουμε στο kml αρχείο και φαίνονται στις παρακάτω εικόνες.



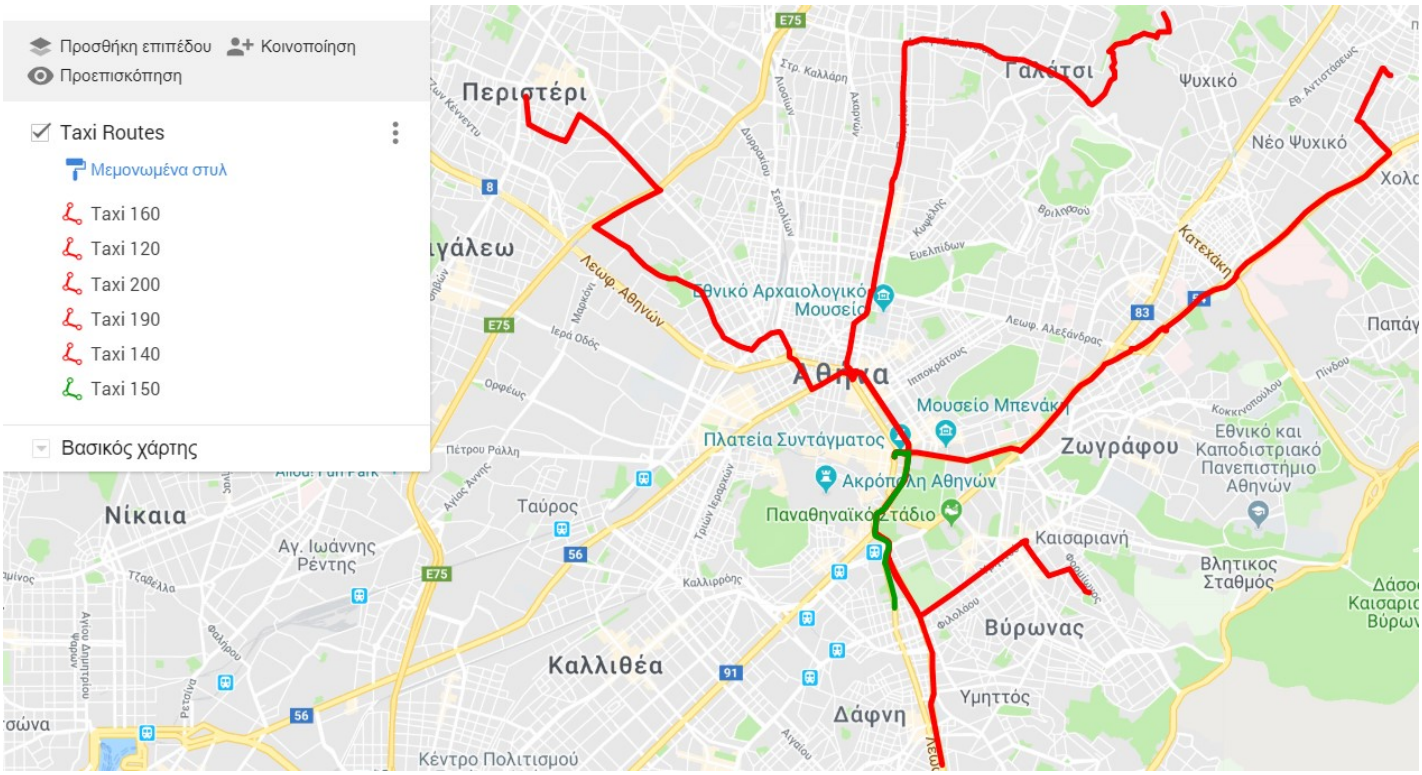




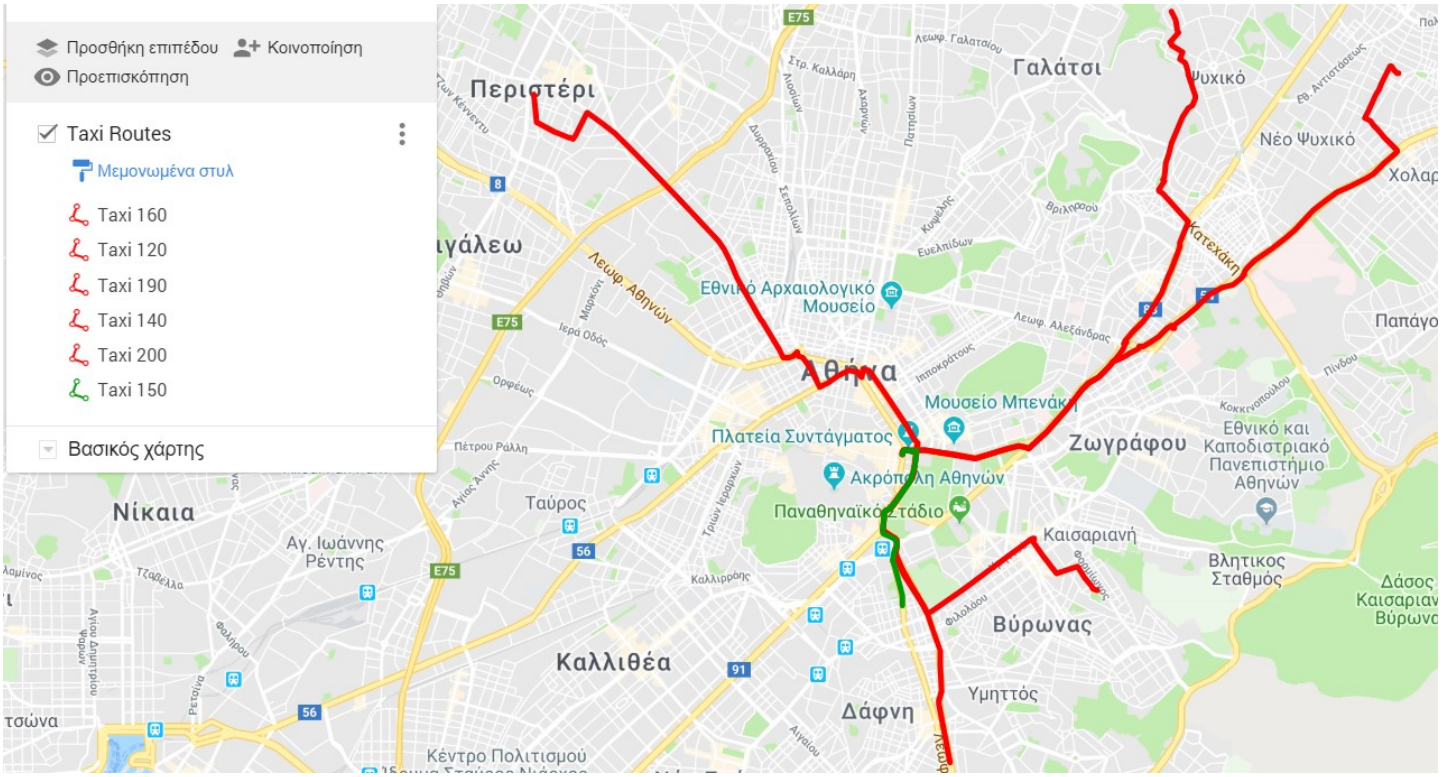
Κατά σειρά τα παραπάνω είναι για μέτωπο ίσο με 7, 23 και 100. Η αρχική κατάταξη είναι σύμφωνα με την χιλιομετρική απόσταση και η επόμενη σύμφωνα με την βαθμολογία των οδηγών των ταξί.



Χάρτης για μέτωπο ίσο με 7.



Χάρτης για μέτωπο ίσο με 23.



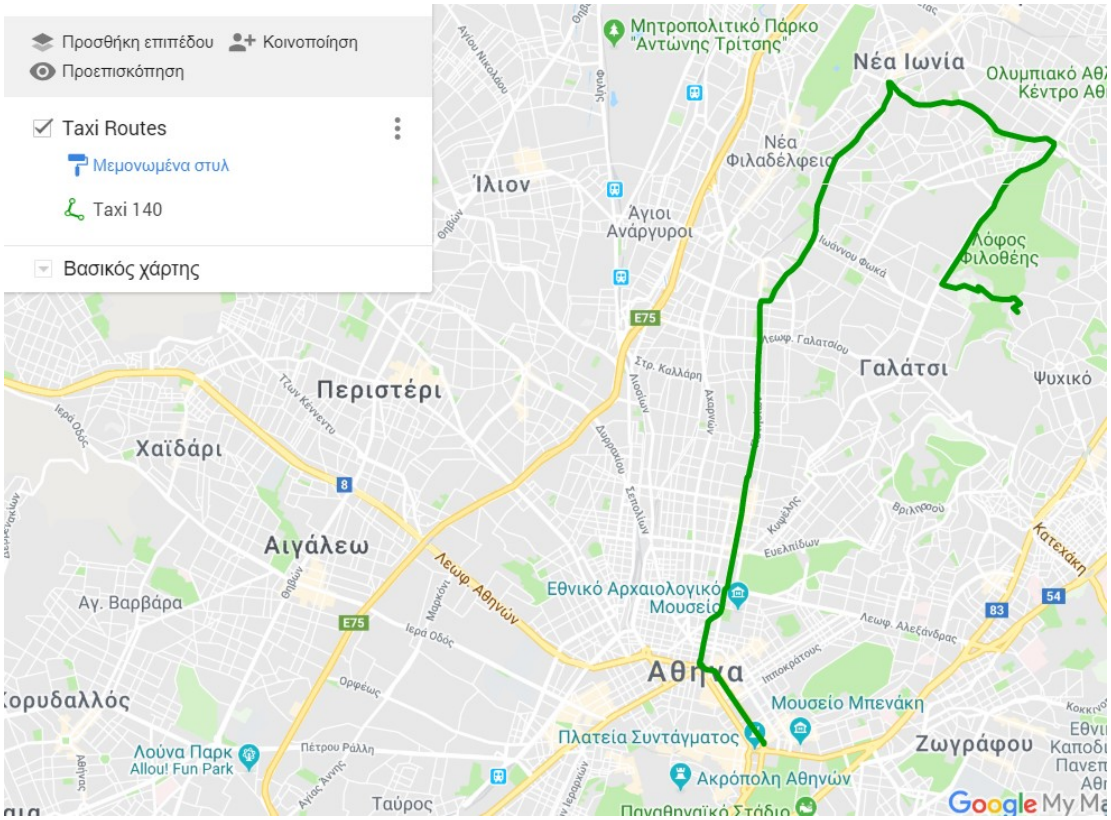
Χάρτης για μέτωπο ίσο με 100.

Παρατηρούμε ότι για διαφορετικές τιμές μετώπου το πρόγραμμα υπολογίζει διαφορετικές διαδρομές για κάποια από τα ταξί. Με πράσινο εμφανίζεται η βέλτιστη από αυτές και με κόκκινο οι υπόλοιπες.

Για το αρχείο που δημιουργήσαμε και είναι διαφορετικό από αυτό που δίδεται στην εκφώνηση σχετικά με την τοποθεσία πελάτη και ταξί θα έχουμε τα ακόλουθα αποτελέσματα για τις συντομότερες διαδρομές με μέτωπο ίσο με 23 θα είναι:

```
140 13.18473386306994
140 Rating: 7.1
```

Παρατηρούμε ότι εφόσον υπάρχουν περισσότερες από δύο αποσκευές και μόνο ένα ταξί ικανοποιεί την προϋπόθεσή μεγέθους ορθά ο αλγόριθμος εμφανίζει σαν αποτέλεσμα μοναδική διαδρομή. Η αναπαράσταση της στον χάρτη είναι η εξής:


Προσθήκη επιπέδου
Κοινοποίηση
Προεπισκόπηση
Taxi Routes
Μεμονωμένα στυλ
Taxi 140
Βασικός χάρτης
Μητροπολιτικό Πάρκο "Αντώνης Τρίτσης"
Νέα Ιωνία
Ολυμπιακό Αθλ
Κέντρο Αθ
Νέα Φιλαδέλφεια
Ίλιον
Άγιοι Ανάργυροι
Λόφος Φιλοθέης
Ιωάννου Φωκά
E75
Λεωφ. Γαλατσίου
Γαλάτσι
Ψυχικό
Στρ. Καλλάρη
Περιστέρι
Χαϊδάρι
Αιγάλεω
Αγ. Βαρβάρα
Λεωφ. Αθηνών
Εθνικό Αρχαιολογικό Μουσείο
Κατεχάκη
Λεωφ. Αλεξάνδρας
Αθήνα
Μουσείο Μπενάκη
Πλατεία Συντάγματος
Ζωγράφου
Ακρόπολη Αθηνών
Λούνα Παρκ
Allou! Fun Park
Ταύρος
Google My Ma