ΣΚΟΠΟΣ

Σκοπός της άσκησης είναι η δημιουργία ενός συστήματος τεχνητής νοημοσύνης , το οποίο δεδομένων των γεωγραφικών θέσεων ενός client, των διαθέσιμων ταξί, διαφόρων σημείων οδών του οδικού δικτύου αλλά και διαφόρων άλλων κόμβων να εντοπίζει μια λίστα με τα κοντινότερα ταξί στον πελάτη και να του τα παρουσιάζει αναλόγως με κάποια κριτήρια. Δεδομένο του προβλήματος αποτελεί επίσης ένα αρχείο που περιέχει τους δρόμους του οδικού δικτύου καθώς και πληροφορίες για αυτούς σχετικά με το μέγεθός τους, την κατεύθυνσή τους (μονής ή διπλής), εμπόδια που υπάρχουν σε αυτούς και διάφορες άλλες πληροφορίες που η χρήση τους επαφίεται στην ευχέρεια του προγραμματιστή.Τέλος, υπάρχει ένα ακόμα αρχείο που περιλαμβάνει επίσης της οδούς, με πληροφορία για αυτές την κίνηση που υπάρχει σε συγκεκριμένες ώρες.

ΥΛΟΠΟΙΗΣΗ

Στο σύστημά μας, υλοποιήσαμε τον αλγόριθμο τεχνητής νοημοσύνης A* σύμφωνα με τα δεδομένα, τα ζητούμενα και τις επιταγές του προβλήματος. Για τον υπολογισμό της πραγματικής αλλά και της ευριστικής τιμής των κόμβων τις οποίες λαμβάνει υπόψη του ο αλγόριθμος, δημιουργήσαμε μια συνάρτηση η οποία λαμβάνει υπόψιν της την πυθαγόρεια απόσταση μεταξύ δυο κόμβων, τον αριθμό των λωρίδων στον μεταξύ τους δρόμο αλλά και της κίνησης που υπάρχει στους αντίστοιχους δρόμους.

Για να έρθουν τα δεδομένα σε διαχειρίσιμη και ισότιμη μορφή μεταξύ τους, πολλαπλασιάσαμε την πυθαγόρεια απόσταση με έναν συντελεστή 10000. Για τον παράγοντα των λωρίδων ανά δρόμο και για τον παράγοντα της κίνησης διακρίναμε περιπτώσεις, αναλόγως με την πυθαγόρεια απόσταση των δύο εξεταζόμενων κόμβων, ώστε τα αποτελέσματα να είναι στην ίδια κλίμακα και ο αλγόριθμος να λειτουργεί σε φυσιολογικά πλαίσια.

Τα δεδομένα του προβλήματος αξιοποιήθηκαν σε σημαντικό βαθμό, καθώς για την επιλογή των δρόμων λαμβάνονται υπόψιν στοιχεία που προσδιορίζουν αν ο δρόμος είναι διαθέσιμος ή όχι. Τέτοια αποτελούν φυσικά εμπόδια, υδάτινοι δρόμοι, μονοπάτια και διάφορα άλλα χαρακτηριστικά που καθιστούν μέρος των οδών μη προσβάσιμο από οχήματα όπως τα ταξί που εξετάζουμε στη συγκεκριμένη άσκηση. Υπόψιν

λήφθηκαν επίσης οι απαιτήσεις του πελάτη για αποσκευές, ο αριθμός ατόμων προς μετακίνηση, η γλώσσα ομιλίας, η τοποθεσία και ο προορισμός πελάτη, βάσει των οποίων έγινε και η επίλογη κάποιων από των ταξί ως διαθέσιμων, τα οποία εξετάστηκαν στη συνέχεια από τον αλγόριθμο A*.

ΤΕΧΝΙΚΗ ΥΛΟΠΟΙΗΣΗΣ

Ο κώδικας μας χωρίζεται στον κύριο κώδικα που είναι γραμμένος σε Java αλλά και σε ένα σημαντικό κομμάτι του γραμμένο σε Prolog. Τα αρχεία .java είναι τα ακόλουθα : Proj.java, FrontTuple.java, Taxi.java, Node.java, FinalTuple.java. Το αρχείο με τα γεγονότα Prolog περιλαμβάνει τα ακόλουθα : belongsTo.pl, lineLanes.pl, lineObstacle.pl, lineOneway.pl,lineTraffic.pl, next.pl, node.pl ,nodeF.pl, και το αρχείο με τους κανόνες Prolog το rules.pl το οποίο περιέχει διάφορους κανόνες που χρησιμοποιούνται στη συνέχεια στο πρόγραμμα μέσω της Java για ερωτήματα (query) πάνω στην prolog. Η χρήση της Prolog λοιπόν έγινε ως μια μεγάλη βάση δεδομένων την οποία χρησιμοποιούμε άμεσα και προσπελαύνουμε μέσω της Java με χρήση της jipprolog.engine .

Χρονική Πολυπλοκότητα

Για την υλοποίηση του κώδικα χρησιμοποιήσαμε δομές όπως η ArrayList, η LinkedList, απλούς πίνακες αλλα και μια ουρά PriorityQueue. Η υλοποίηση του αλγορίθμου A* έγινε με έναν απλό βρόγχο επανάληψης και χρήση της PriorityQueue, ώστε να αποφεύγονται μεγάλες δεσμεύσης μνήνης στο stack, καθώς επίσης και να επιτυγχάνεται η λύση του προβλήματος σε μικρό χρόνο. Συγκεκριμένα, το διάβασμα των δεδομένων πραγματοποιείται σε γραμμικό χρόνο O(n) και ο αλγόριθμος A* σε χρόνο O(n*logn) καθώς έχουμε n δεδομένα και το Priority Queue λειτουργεί σε πολύ καλό χρόνο O(logn) . Συνεπώς η συνολική πολυπλοκότητα του προβλήματος είναι O(n* logn) που σημαίνει πως για τα δεδομένα της άσκησης τρέχει σχεδόν ακαριαία (1-2 sec). Ωστόσο, το πρόγραμμα δεν τελειώνει στον παραπάνω χρόνο συνολικά, αφού η φόρτωση των δεδομένων και των κανόνων Prolog το καθύστερει παρόλο που ο αλγόριθμος είναι ακαριαίος , με αποτέλεσμα να χρειάζεται περίπου μισό λεπτό για να δώσει την τελική ολοκληρωτική απάντηση.

Το αρχείο rules.pl με τους κανονες της Prolog περιλαμβάνει αρχικά κάποια includes ώστε να μπορεί να ψάξει στα γεγονότα της Prolog και στην συνέχεια αναλυτικά τους εξής κανόνες :

- findF(X,F)

Το οποίο παίρνει έναν κόμβο X και επιστρέφει την F του δηλ. την ευριστική του κόμβου

- findLane(Line_id, Lanes)

Που παίρνει ένα lineid και επιστρέφει τα Lanes του αντίστοιχου line

- floorTime(Time, StartTime, EndTime)

Που παίρνει ένα time (του πελάτη) και φτιάχνει τα όρια στα οποία αυτό εντάσσεται σύμφωνα με τα δεδομένα της εκφώνησης για την κίνηση

- findTraffic(Line_id, Time, Traffic)

Που παίρνει ένα lineid και ένα χρόνο time και επιστρέφει την κίνηση που έχει η συγκεκριμένη γραμμή-οδός το συγκεκριμένο χρόνο

- findXY(Id, X, Y)

Που παίρνει ένα id ενός κόμβους και επιστρέφει τις συντεταγμένες του

- findNext(X, Y, Gy, Fx, Fy)

Που παίρνει ένα id του κόμβου X και επιστρέφει όλους του τους γείτονες δηλαδή τους κόμβους με τους οποίους συνδέεται. Το Y είναι το id των κόμβων αυτών και τα Gy, Fx, Fy είναι τιμές των κόμβων που χρησιμοποιούνται για τον υπολογισμό της συνολικής ευριστικής.

Ο Αλγόριθμος

Για την υλοποίηση του Αλγορίθμου, αρχικά βρίσκουμε τον κόμβο που είναι κοντινότερος στη θέση του πελάτη και τον βάζουμε ως στόχο του αλγορίθμου. Αντίστοιχα λειτουργούμε για τα ταξί βρίσκοντας τους κοντινότερους σε αυτά κόμβους και θεωρώντας αυτά τα σημεία ως σημεία εκκίνησης του Αλγορίθμου.

Ξεκινώντας από τα ταξί κάθε φορά λοιπόν κάνουμε μια επανάληψη while έως ότου βρούμε τον στόχο. Χρησιμοποιούμε το μέτωπο αναζήτησης του αλγορίθμου A* το οποίο υλοποιείται μέσω της PriorityQueue με το όνομα Front ώστε κάθε φορά να έχουμε το min στοιχείο ως προς την ευρυστική που θα αναπτύξει ο αλγόριθμος σε λογαριθμικό χρονο. Η PriorityQueue περιλαμβάνει τους κόμβους σε μορφή τούπλας, καθώς χρειαζόμαστε διάφορα στοιχεία στη συνέχεια ώστε να μπορέσουμε να τα παρουσιάσουμε σωστά στο χρήστη. Το κλειστό σύνολο με τους κόμβους που έχει επισκεφθεί ο αλγόριθμος υλοποιείται σε εμάς μέσω ενός πίνακα συχνότητας, το closure, ο οποίος έχει μέγεθος όσο

όλοι οι κόμβοι. Αρχικοποιείται στο 0 και μόλις κάποιος κόμβος αναπτυχθεί από το Front και γίνει remove από το Front, τότε η αντίστοιχη θέση του στον πίνακα closure παίρνει την τιμή 1, που δείχνει πως ο αντίστοιχος κόμβος αναπτύχθηκε και ανήκει πλέον στο κλειστό σύνολο. Συνεπώς, αν βρεθεί κάποιος άλλος με το ίδιο id στη συνέχεια δεν θα εξεταστεί από τον A*. Η συνεργασία μεταξύ του Front και closure και η ολοκλήρωση του A* γίνεται μέσω της κλήσης του κατηγορήματος next της Prolog μέσω της Java, το οποίο επιστρέφει τους γείτονες του εξεταζόμενου κόμβου, δηλαδή τους κόμβους που μπορεί να συνεχίσει να επισκέπτεται ο A*. Αυτοί οι κόμβοι στη συνέχεια τοποθετούνται στο Front και ο αλγόριθμος συνεχίζει την εκτέλεση του έως ότου φτάσει στον κόμβο-στόχο που είναι ο κόμβος του πελάτη. Για την επαναφορά του μονοπατιού από τον κόμβο εκκίνησης έως τον κόμβο στόχο χρησιμοποιείται ένας πίνακας path_list ο οποίος για κάθε κόμβο έχει μια τιμή που αντιστοιχή στον πατέρα του αντίστοιχου κόμβου, δηλαδή στον προηγούμενο κόμβο του στο μονοπάτι.

Ετσι για κάθε ταξί, όταν ολοκληρωθεί ο A*, το path_list περιλαμβάνει το συνολικό και τελικό μονοπάτι της λύσης, το οποίο τελικά αποθηκεύεται στη λίστα final_list που περιλαμβάνει τα μονοπάτια για κάθε ταξί.

Έπειτα, παρουσιάζονται στο χρήστη όπως ζητήθηκε από την εκφώνηση της άσκησης δύο κατατάξεις των διαθέσιμων ταξί.
-Η πρώτη είναι βάση της κοντινότερης συνολικής απόστασης του ταξί από τον πελάτη και περιλαμβάνει τα id των ταξί .
-Η δεύτερη είναι βάση της αξιολόγησης των διαθέσιμων ταξί από τους χρήστες και περιλαμβάνει το id των ταξί και την αντίστοιχη αξιολόγηση.

Τέλος παρουσιάζονται στην οθόνη του χρήστη και η συντεταγμάνες της διαδρομής του κοντινότερου στον πελάτη ταξί καθώς επίσης και οι συντεταγμένες της διαδρομής του πελάτη από το σημείο παραλαβής του από το ταξή έως τον προορισμό του.

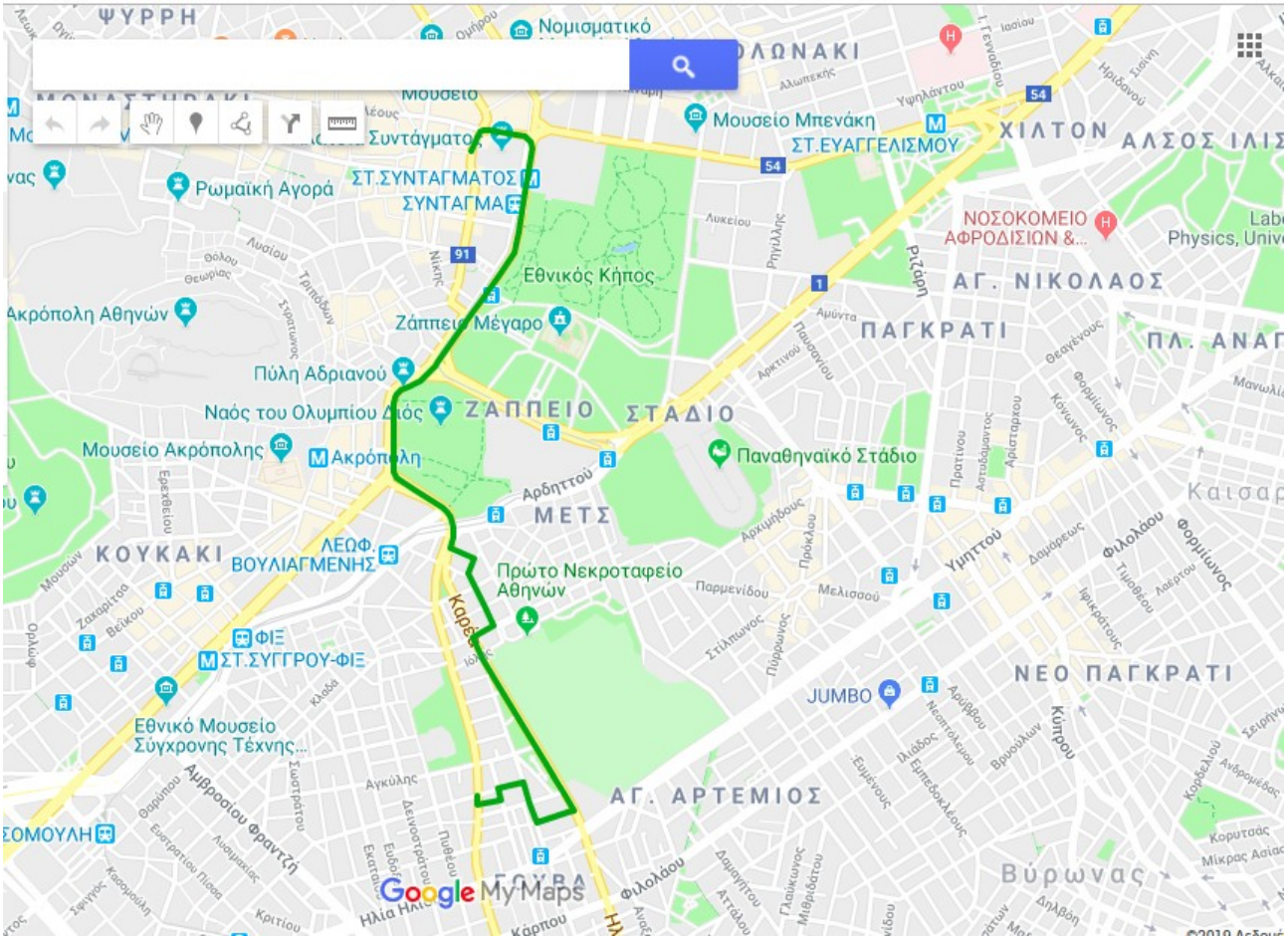
Παρακάτω φαίνονται οι δύο κατατάξεις :

```
The closest to the client available taxis are:
Taxi ID: 150
Taxi ID: 160
Taxi ID: 120
Taxi ID: 190
Taxi ID: 140
Taxi ID: 200
The best user rating available taxis by ascending order are:
Taxi ID: 200 with user rating 5.2
Taxi ID: 160 with user rating 6.1
Taxi ID: 140 with user rating 7.1
Taxi ID: 150 with user rating 7.3
Taxi ID: 190 with user rating 7.4
Taxi ID: 120 with user rating 9.2
```

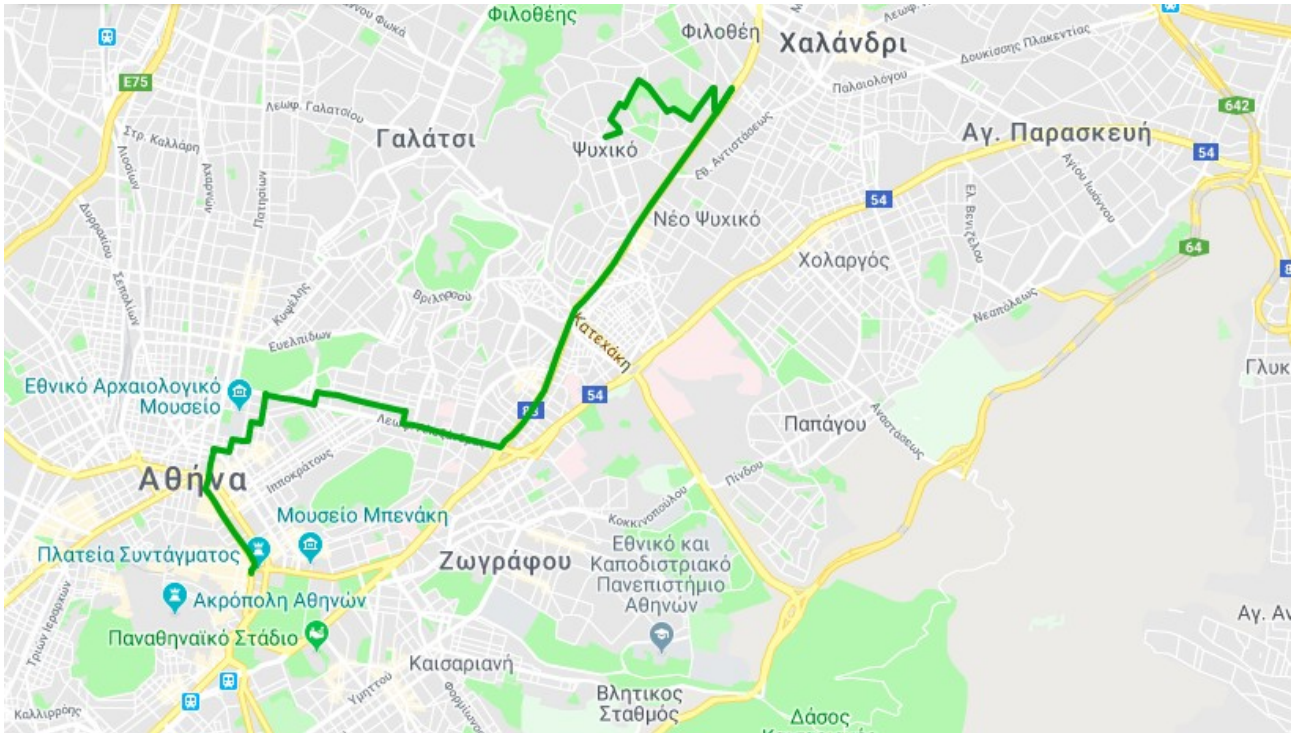
KML ΑΡΧΕΙΑ

KML ΓΙΑ ΤΑ ΔΕΔΟΜΕΝΑ ΤΗΣ ΕΚΦΩΝΗΣΗΣ

ΒΕΛΤΙΣΤΟ ΤΑΞΙ

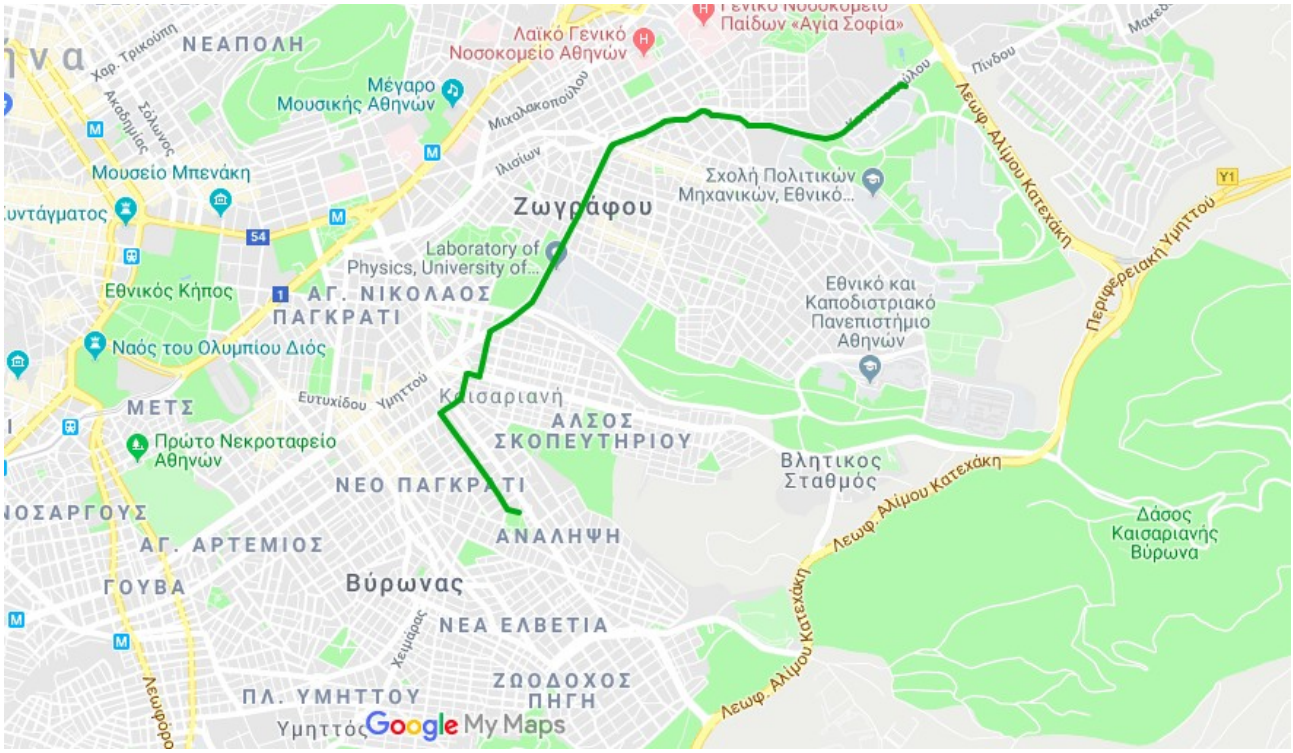


ΔΙΑΔΡΟΜΗ ΠΡΟΟΡΙΣΜΟΥ

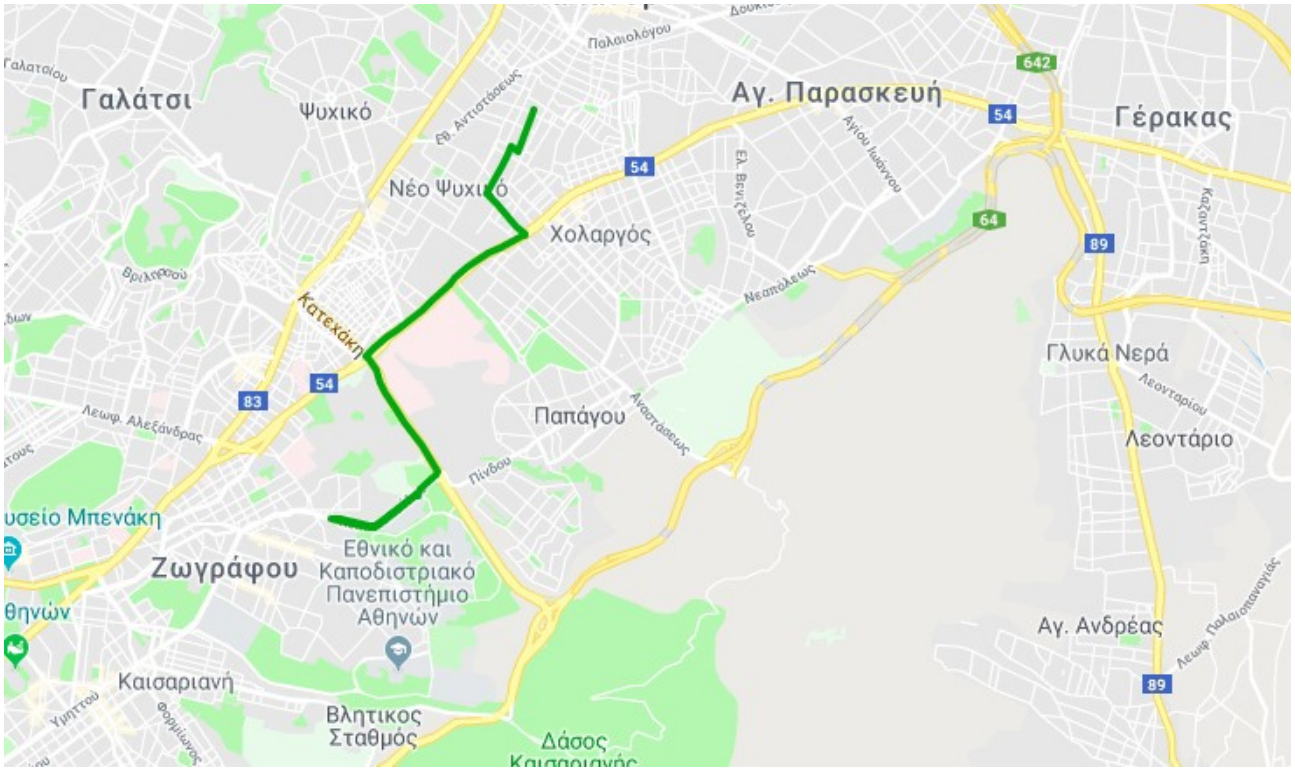


KML ΓΙΑ ΔΙΚΑ ΜΑΣ ΔΕΔΟΜΕΝΑ

ΒΕΛΤΙΣΤΟ ΤΑΞΙ



ΔΙΑΔΡΟΜΗ ΠΡΟΟΡΙΣΜΟΥ


Γαλάτσι
Ψυχικό
Αγ. Παρασκευή
Γέρακας
Νέο Ψυχικό
Χολαργός
Κατεχάκη
Παπάγου
Γλυκά Νερά
Λεοντάριο
Ζωγράφου
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Αγ. Ανδρέας
Καισαριανή
Βλητικος Σταθμός
Δάσος
54
642
64
89
83