ΔΕΥΤΕΡΟ ΘΕΜΑ ΣΤΗΝ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Ομάδα: Βιδαλάκης Γιώργος ΑΜ el15052

Πολλάτος Βασίλης ΑΜ el15011

Στα πλαίσια του δεύτερου θέματος υλοποιήσαμε μια παραλλαγή του αλγορίθμου A* της πρώτης άσκησης, ο οποίος βρίσκει βέλτιστες διαδρομές σε χάρτη, ελαχιστοποιώντας ένα κριτήριο κόστους J που σταθμίζει μήκος και χρονική διάρκεια διαδρομής. Έχουμε θεωρήσει ένα μοντέλο του κόσμου, το οποίο περιέχει όλες τις πληροφορίες για τους κόμβους, τις διαδρομές του χάρτη, όπως αυτές μας δώθηκαν στα αρχεία csv της άσκησης ενώ για τα ταξί και τους πελάτες χρησιμοποιήσαμε σε πρώτη φάση τα δεδομένα της εκφώνησης και σε δεύτερη φάση κάποια δικά μας.

ΜΟΝΤΕΛΟ ΚΑΙ ΓΝΩΣΗ ΤΟΥ ΚΟΣΜΟΥ

θεωρούμε ότι ο κόσμος αποτελείται από σημεία του χάρτη (κόμβοι), τα οποία βρίσκονται πάνω σε συγκεκριμένα οδικά τμήματα. Ένα σημείο μπορεί να ανήκει είτε σε ένα μόνο οδικό τμήμα (ενδιάμεσα σημεία δρόμων) είτε σε περισσότερα (σημεία διασταύρωσης οδικών τμημάτων). Τα οδικά τμήματα χαρακτηρίζονται από ένα αναγνωριστικό (line_id). Ένας δρόμος μπορεί να εμφανίζεται σπασμένος στο μοντέλο του κόσμου μας, δηλαδή να συνίσταται από μια αλληλουχία οδικών τμημάτων που έχουν το ίδιο όνομα, αλλά διαφορετικό line_id και ο τελευταίος κόμβος του προηγούμενου ταυτίζεται με τον πρώτο κόμβο του επόμενου τμήματος. Οι διασταυρώσεις διαφορετικών δρόμων προσδιορίζονται κι αυτές από κόμβους που ανήκουν ταυτόχρονα σε διαφορετικά τμήματα με διαφορετικά line_id. Τα ταξί κι ο πελάτης αντιστοιχίζονται σε κάποιον από τους υπάρχοντες κόμβους (τον πλησιέστερο προς αυτούς) που ανήκει σε δρόμο.

Λόγω των διασταυρώσεων δρόμων, ένας κόμβος, δηλαδή ένα σημείο (X, Y) που έχει ένα μοναδικό node_id μπορεί να εμφανίζεται συνολικά πολλές φορές, μια σε κάθε οδικό τμήμα που διέρχεται από το συγκεκριμένο σημείο. Έτσι ορίζουμε αύξοντες αριθμούς, που αποτελούν id για τις διαφορετικές εμφανίσεις των κόμβων. Οι αύξοντες αριθμοί inc_num αρχίζουν από μηδέν και αυξάνονται κατά 1 καθώς διαβάζουμε το αρχείο lines.csv.

Οι κόμβοι εμφανίζονται με μια συγκεκριμένη σειρά κατά μήκος του κάθε οδικού τμήματος , η οποία είναι ανάλογη με την κατεύθυνση του δρόμου. Αν το πεδίο oneway του line στο csv αρχείο έχει την τιμή yes σημαίνει πως ο δρόμος είναι μονής κατεύθυνσης και η σειρά με την οποία δίνονται οι κόμβοι των οδικών τμημάτων του δρόμου ακολουθεί την κατεύθυνση του δρόμου. Αν το oneway είναι -1 ή reverse τότε ο δρόμος είναι μονής κατεύθυνσης αλλά η σειρά με την οποία δίνονται οι κόμβοι είναι ανάποδη από την πραγματική κατεύθυνση του δρόμου. Αν το oneway είναι no, τότε ο δρόμος είναι διπλής κατεύθυνσης, οπότε υπάρχει στις διαδρομές η δυνατότητα να διασχίζουμε τους κόμβους με τη σειρά που δίνονται είτε με την αντίθετη. Όταν δε δίνται τιμή στο oneway θεωρούμε πως ο δρόμος είναι διπλής κατεύθυνσης.

Μερικές από τις γραμμές (lines) που μας δίνονται στο csv αρχείο δεν αποτελούν δρόμους κυκλοφορίας οχημάτων. Μπορεί να πρόκειται για γραμμές τρένου, ακτοπλοϊκές γραμμές, λεωφορειογραμμές, σοκάκια, πεζόδρομους, σκάλες, σύνορα περιοχών, φυσικά σύνορα ή δρόμους αμφίβολης κατάστασης που αποφεύγουμε. Ακόμη, μπορεί να πρόκειται για δρόμους που είναι ιδιωτικοί και δεν έχουν άδεια τα ταξί να κυκλοφορούν σε αυτούς. Τέλος μπορεί να υπάρχουν εμπόδια σε δρόμους, οπότε θεωρούμε ότι οι δρόμοι αυτοί είναι κλειστοί και τους αποκλείουμε από το οδικό δίκτυο. Εκτός από την κατεύθυνση υπάρχουν κι άλλα χαρακτηριστικά των δρόμων, τα οποία λαμβάνουμε υπόψιν. Όλες αυτές οι περιπτώσεις για τα πεδία highway, access, railway, seaway, busway, barrier, natural, bountry καταγράφονται στη διεύθυνση https://wiki.openstreetmap.org/wiki/Tags. Στα αρχεία γνώσης προσθέτουμε μόνο έγκυρους δρόμους του οδικού δικτύου.

Ένα άλλο χαρακτηριστικό των δρόμων είναι η κλίση. Η κλίση επηρεάζει με δύο τρόπους το κόστος των διαδρομών. Πρώτον, το πραγματικό μήκος ενός δρόμου είναι ελαφρώς μεγαλύτερο από αυτό που φαίνεται στον επίπεδο χάρτη, δηλαδή από την προβολή του στην επιφάνεια του χάρτη. Συγκεκριμένα, είναι πολλαπλασιαμένο με ένα παράγοντα sqrt(1+tan^2(φ)) όπου tan(φ) η κλίση του δρόμου. Επίσης, η κλίση επηρεάζει το όριο ταχύτητας, με την έννοια ότι σε δρόμους μεγάλης κλίσης υπάρχει φυσικός περιορισμός ταχύτητας . Σε κλίση μεγαλύτερη από 25% θεωρούμε πως η ταχύτητα θα πρέπει να είναι κάτω από 15 km/h, σε κλίση από 20% μέχρι 25% θεωρούμε ταχύητα μικρότερη από 30 km/h και σε μικρότερες κλίσεις δεν τίθεται περιορισμός λόγω κλίσης. Θα μπορούσαμε να προσθέσουμε στο κόστος και την αυξημένη κατανάλωση βενζίνης, όμως στην πραγματικότητα αυτό δε χρεώνεται στον πελάτη. Όταν δε δίνεται κλίση θεωρούμε πως υπάρχει μια μικρή μέση κλίση κάτω από 5%.

Το όριο ταχύτητας και η μέση ταχύτητα επηρεάζεται από διάφορες παραμέτρους. Πρώτον την κλίση, όπως ήδη αναφέραμε. Επίσης, το όριο ταχύτητας, όπου υπάρχει θα πρέπει να γίνεται σεβαστό. Το όριο είτε δίνεται ως speed_limit είτε το συμπεραίνουμε εμείς για κάποιους δρόμους, πχ residential που έχουν πολύ μικρό όριο. Επιπλέον, η κίνηση επηρεάζει την ταχύτητα κυκλοφορίας. Η κίνηση δίνεται για μερικούς μόνο από τους δρόμους του χάρτη και είναι συνάρτηση της ώρας. Για όσους δε δίνεται θεωρούμε πως η κίνηση είναι σε κανονικά επίπεδα, (ούτε πυκνή, αλλά ούτε και μηδενική). Έχουμε θεωρήσει πως σε πυκνή κίνηση η ταχύτητα περιορίζεται σε 5 km/h και σε μέτρια κίνηση το πολύ 30 km/h . Όταν δε δίνεται κάποια πληρφορία για την κίνηση θεωρούμε πως υπάρχει μια μέση κίνηση, οπότε η μέση ταχύτητα είναι 30 km/h. Αν και το default όριο στο οδικό δίκτυο είναι 50 km/h μέσα στην Αθήνα είναι μειωμένο τόσο επειδή υπάρχουν πολλοί μικροί και πυκνοκατοικημένοι δρόμοι με όριο 30 km/h, όσοι και λόγω της κίνησης που θέτει περιορισμούς μεσω φαναριών, συμφόρησης κτλ. Τελικά, συμψηφίζοντας όλα τα παραπάνω παίρνουμε μια τιμή για την ταχύτητα σε κάθε μετάβαση από κόμβο σε κόμβο πάνω σε ένα δρόμο, που είναι <u>το ελάχιστο μεταξύ ορίου ταχύτητας, ταχύτητας που επιτρέπει το επίπεδο κυκλοφοριακής συμφόρησης και ταχύτητας που επιτρέπει η κλίση του δρόμου.</u>

Ο φωτισμός των δρόμων έχει σημασία στον κόσμο μας, καθώς αποφεύγουμε την κίνηση σε δρόμους που δεν έχουν φωτισμό όταν είναι περασμένη η ώρα (21.00 -6.00).

Τα διόδια είναι μια άλλη παράμετρος που θα πρέπει να λάβουμε υπόψιν. Όταν το ταξί εισέρχεται σε δρόμο που έχει διόδια, τότε θα πληρώσει το κόμιστρο και αυτό θα προστεθεί στο χρηματικό κόστος διαδρομής. Για κάθε δρόμο διοδίων το κόμιστρο πληρώνεται μια μόνο φορά, κατά την είσοδο στο συγκεκριμένο (αυτοκινητό)δρομο και στη συνέχεια το ταξί συνεχίζει κανονικά την πορεία του επί του αυτοκινητόδρομου. Το κόμιστρο θεωρήκε 2 ευρώ όσο δηλαδή ειναι στην Αττική Οδό.

Έτσι λοιπόν ορίζουμε ένα κατευθυνόμενο έμβαρο γράφημα G που αποτυπώνει τη γνώση του χάρτη. Τα βάρη των ακμών εκφράζουν την επιβάρυνση ΔJ που προσθέτει η συγκεκριμένη μετάβαση και είναι συνάρτηση της ώρας. Επειδή οι διαδρομές ολοκληρώνονται εντός ενός συγκεκριμένου χρονικού διαστήματος που η κίνηση είναι περίπου σταθερή, οι ακμές του γραφήματος αντιμετωπίζονται σαν χρονοαμετάβλητες κατά την εκτέλεση του αλγορίθμου αναζήτησης. Οι μεταβάσεις που αποτελούν στροφή σε διασταύρωση, κατά τις οποίες δεν υπάρχει αλλαγή θέσης αλλά μόνο αλλαγή δρόμου έχουν μηδενικό (για την ακρίβεια μηδαμινό κόστος (ε), όπως θα δούμε παρακάτω), εκτός κι αν κατά την είσοδο στο νέο δρόμο πληρώνουμε διόδια. Στις μεταβάσεις αυτές πάμε σε επιτρεπτό νέο δρόμο (is_road, not_scary). Οι μεταβάσεις που περιλαμβάνουν αλλαγή θέσης έχουν θετικό κόστος ΔJ και γίνονται μεταξύ διαδοχικών ενδιάμεσων σημείων δρόμων τηρώντας τις κατευθύσεις των δρόμων, μονές ή διπλές.

<u>Η συνάρτηση κόστους J</u> που θεωρούμε είναι το ποσό που χρεώνουν τα ταξί στους πελάτες τους στην πόλη των Αθηνών. Είναι συνάρτηση της διάρκειας κούρσας και των διανυόμεενων χιλιομέτρων. Από τις 24:00 μέχρι τις 5:00 ισχύει “διπλή ταρίφα”. Οι βαλίτσες χρεώνονται επιπλέον (+0.38 ευρώ/βαλίτσα), όπως και τα διόδια. Αναλυτικά η συνάρτηση κόστους έχει ως εξής : max{ 1.14 + C*χιλιομετρική_απόσταση + 10.46*διάρκεια_κούρσας_σε_ώρες + 0.38 * πλήθος_βαλιτσών + κόμιστρα διοδίων , 3.27 } ευρώ.

Η σταθερά C είναι 0.65 στην κανονική και 1.14 στη "διπλή ταρίφα". Source: https://www.athensairporttaxi.com/gr/κοστος-ταξι/κομιστρα-ταξι-ελλαδα

Για τον εκάστοτε πελάτη γνωρίζουμε πόσα άτομα ζητάει να μεταφερθούν, τι γλώσσα μιλάει, πόσες αποσκευές φέρει, που βρίσκεται, που θέλει να πάει και τι ώρα καλεί το ταξί.

Για τα ταξί γνωρίζουμε το που βρίσκονται, τι γλώσσες μιλάει ο οδηγός, αν είναι διαθέσιμα, αν είναι μεγάλων αποστάσεων, τον τύπο του οχήματος, τη χωρητικότητα και το rating του ταξί.

Το ζητούμενο είναι να βρούμε κατάλληλο ταξί που θα έρθει να παραλάβει τον πελάτη και να τον πάει στον προορισμό του. Το ταξί ακολουθεί πάντα διαδρομές που ελαχιστοποιούν το συνολικό κόστος. Ο πελάτης ενδιαφέρεται για τον χρόνο αναμονής μέχρι να τον παραλάβει το ταξί, τη διάρκεια της κούρσας και το ποσό που θα χρεωθεί, ενώ υπάρχουν και κάποιες απαραίτητες προδιαγραφές προκειμένου ένα ταξί να θεωρηθεί κατάλληλο για μια κούρσα. Θα πρέπει το ταξί να είναι διαθέσιμο, το όχημα να χωράει τις αποσκευές καθώς και τους επιβάτες δηλαδή ο τύπος και η χωρητικότητα του οχήματος να είναι κατάλληλα και ο οδηγός να μιλάει τη γλώσσα του πελάτη. Έχουμε ορίσει μέγιστη χωρητικότητα αποσκευών ανάλογα με τον τύπο του οχήματος (2 αποσκευές στα μικρά, 3 αποσκευές στα μεσαία και 5 αποσκευές στα ευρύχωρα ταξί), ενώ η μέγιστη χωρητικότητα ατόμων δίνεται ως χαρακτηστικό του ταξί. Επίσης θα πρέπει το ταξί να είναι μακρινών διαδρομών, αν η διαδρομή είναι μακρινή, δηλαδή μεγαλύτερη απο 50 χιλιόμετρα . Το τελευταίο κριτήριο το ελέγχουμε με βάση τις haversine αποστάσεις, μιας και οι πραγματικές δεν είναι ακόμη γνωστές. Τα ταξί που είναι κατάλληλα για τη συγκεκριμένη κούρσα ταξινομούνται αρχικά ως προς το χρόνο αναμονής του client μέχρι να έρθουν να τον παραλάβουν και στη συνέχεια ως προς το χρόνο αναμονής και το rating των ταξί ταυτόχρονα και κάθε φορά εμφανίζονται στον χρήστη τα κ καλύτερα ταξί της κατατάταξης, όπου το κ δίνεται από το χρήστη ως είσοδος.

ΚΑΝΟΝΕΣ ΚΑΙ ΓΕΓΟΝΟΤΑ PROLOG

Έτσι λοιπόν, δημιουργούμε μια βάση δεδομένων σε prolog που περιέχει όλες τις πληροφορίες για τον ισοδύναμο γράφο του χάρτη, τον πελάτη και τα ταξί. Η βάση είναι όσο το δυνατόν κανονικοποιημένη, δηλαδή περιέχει σχέσεις μεταξύ των attributes όσο το δυνατόν πιο σπασμένες (αποσυζευγμένες), ώστε να μην αναγκαζόμαστε να κάνουμε join πάνω σε πολλά attributes ταυτόχρονα, αν δεν το χρειαζόμαστε πραγματικά. Στη βάση τα κατηγορήματα συνδέουν ένα attribute με το αντίστοιχο id της οντότητας που ανήκει το attribute, πχ το X με το node_id. Τα γεγονότα που αποθηκεύουμε στη βάση είναι τα εξής: node_id(INC_NUM, N_ID), nodeX(N_ID, X), nodeY(N_ID, Y), increasing_number_line_id(INC_NUM, L_ID), line_one_way(L_ID, DIRECTION), is_road(L_ID), no_light(L_ID), has_toll(L_ID), incline(L_ID, INCLINATION), speed_limit(L_ID, SPEED_LIMIT), traffic(L_ID, BITMASK). Τα αρχεία γεγονότων παράγονται από κώδικα java που διαβάζει τα csv αρχεία. Κατά την ανάγνωση γράφει στα αρχεία γεγονότα που προκύπτουν από την είσοδο. Ο κώδικας αυτός τρέχει μια φορά για να παράξει τη βάση γνώσης.

Επίσης έχουμε φτιάξει 4 βασικούς κανόνες prolog που ελέγχουν :

-αν ένα ταξί είναι κατάλληλο για τον πελάτη,

-ποιος είναι ο κοντινότερος κόμβος σε ένα σημείο (X, Y),

-ποιά είναι η haversine απόσταση δύο κόμβων. Η απόσταση αυτή εφαρμόζεται για σημεία σε επιφάνεια σφαίρας που έχουν γνωστά γεωγραφικά μήκη και πλάτη.

$$D(x,y) = 2rarcsin\left(\sqrt{sin^2(\frac{\phi_2-\phi_1}{2}) + cos(\phi_2)\cdot cos(\phi_1)sin^2(\frac{\lambda_2-\lambda_1}{2})}\right)$$

όπου λ τα γωγραφικά μήκη και φ τα γεωμετρικά πλάτη και r η ακτίνα της γης.

-ποιοι είναι οι γείτονες ενός δεδομένου κόμβου στο γράφο και ποιο το κόστος της σύνδεσης (σε δεδομένη ώρα). Ειδική περίπτωση εδώ αποτελούν οι μεταβάσεις που συνδέονται με αλλαγή δρόμου, δηλαδή στροφή σε διασταύρωση (ο νέος δρόμος θα πρέπει να είναι έγκυρος, is_road και να μην είναι αφώτιστος αν είναι νύχτα). Κατά τη μετάβαση αυτή ξεκινάμε από ένα σημείο (X, Y) και καταλήγουμε στο ίδιο (X, Y), αλλά πάνω σε διαφορετικό line_id. Γίνεται δηλαδή μετάβαση από ένα κόμβο με κάποιο αύξοντα αριθμό σε έναν άλλο με ίδιο node_id αλλά διαφορετικό αύξοντα αριθμό. Όπως έχουμε αναφέρει προηγουμένως στον ισοδύναμο γράφο του προβλήματος οι κόμβοι χαρακτηρίζονται από τον αύξοντα αριθμό τους κι όχι το node_id. Απλώς οι μεταβάσεις προς το ίδιο node_id έχουν μηδενικό κόστος. Αν όμως πρόκειται για μετάβαση προς δρόμο με διόδια θέτουμε ως κόστος μετάβασης το κόμιστρο του διοδίου. Στις υπόλοιπες περιπτώσεις δε θέτουμε το κόστος ακριβώς μηδέν, αλλά πολύ μικρό (ε). Έτσι αποφεύγουμε τη δημιουργία κύκλων. Άλλες επιτρεπτές μεταβάσεις είναι αυτές που ο αύξον αριθμός μεταβάλλεται κατά 1 και κινούμαστε κατά μήκους του ίδιου οδικού τμήματος (ίδιο line_id). Εδώ έχουμε μη μηδενικό κόστος μετάβασης και θα πρέπει να λάβουμε υπόψιν την κατεύθυνση του δρόμου. Στους διπλής μπορούμε να μεταβαίνουμε σε αύξοντα αριθμό είτε μικρότερο είτε μεγαλύτερο κατά 1, στους μονής με oneway=yes μπορούμε να μεταβαίνουμε σε αύξοντα αριθμό μεγαλύτερο κατά 1 και στους μονής με oneway = -1 μπορούμε να μεταβαίνουμε σε αύξοντα αριθμό μικρότερο κατά 1.

Οι κανόνες αυτοί χρησιμοποιούν άλλους βοηθητικούς, όπως εύρεση haversine απόστασης, έλεγχο κυκλοφοριακής συμφόρησης σε δεδομένη χρονική στιγμή (το traffic κατά διαστήματα κωδικοποιείται με 12-μπιτο bitmask σε τριαδική βάση, μιας και υπάρχουν 3 επίπεδα του traffic (high, medium, low) και 12 χρονικά διαστήματα (δίωρα) της μέρας που ορίζεται επίπεδο κίνησης), εύρεση ταχύτητας κατά τη μετάβαση από ένα κόμβο του δρόμου σε ένα άλλο, που προκύπτει από το όριο ταχύτητας, την κίνηση, την κλίση κτλ.

Η java χρησιμοποιεί τους κανόνες αυτούς για τις λειτουργίες της (αντιστοίχιση κόμβων σε πελάτη και ταξί, εύρεση ευριστικής, ανάπτυξη μετώπου A* κτλ). Ωστόσο, επειδή είχε μειωμένη ταχύτητα η κλήση αυτών των κανόνων για αναζήτηση λύσεων στα γεγονότα της βάσης, δημιουργήσαμε γεγονότα που περιείχαν όλες τις πλειάδες λύσεις που ικανοποιούσαν τους κανόνες και τα αποθηκεύσαμε στη γνώση. Αυτό το κάναμε καλώντας τους αντίστοιχους κανόνες της prolog με unbound ορίσματα και βάζοντάς τη να γράψει όλες τις πιθανές λύσεις του κανόνα σε ένα άλλο αρχείο prolog σε μορφή γεγονότος. Στη java καλούμε τα γεγονότα αυτά και παίρνουμε τα ίδια αποτελέσματα που θα έδινε η κλήση των κανόνων αλλά πολύ γρηγορότερα.

ΚΩΔΙΚΑΣ JAVA

Υπάρχουν 2 διαφορετικά είδη κώδικα java με διαφορετικό ρόλο το καθένα. Οι πρώτοι είναι preprocessing κώδικες που διαβάζουν τα αρχεία csv και παράγουν γεγονότα γνώσης (αρχεία prolog).

Το άλλο κομμάτι κώδικα, A_star.java , αξιοποιεί τα γεγονότα γνώσης, τρέχει τον αλγόριθμο A* και παράγει βέλτιστα μονοπάτια από τα ταξί προς τον πελάτη και από τον πελάτη προς τον προορισμό. Συγκεκριμένα, τρέχει τον A* μια φορά για κάθε ταξί με αφαιτηρία το ταξί και στόχο τον πελάτη και μια φορά με αφαιτηρία τον πελάτη και στόχο τον τελικό προορισμό. Ο A* μπορεί να χρησιοποιηθεί ως αλγόριθμος για την ελαχιστοποίηση του κριτηρίου κόστους J που θεωρήσαμε επειδή το κριτήριο αυτό είναι αθροιστικό ως προς

τα επί μέρους κόστη των μεταβάσεων, δηλαδή το συνολικό κόστος προκύπτει ως το άθροισμα των κοστών όλων των μεταβάσεων που συνιστούν την συνολική διαδρομή.

Ο αλγόριθμος είναι γενικά ίδιος με την πρώτη άσκηση, μόνο που αλλάζει το κριτήριο κόστους και ο τρόπος που παίρνουμε τους γείτονες ενός κόμβου κατά την ανάπτυξη του μετώπου.Οι γείτονες ενός κόμβου λαμβάνονται με query προς την prolog (next) δίνοντας ως όρισμα τον αύξοντα αριθμό του τρέχοντος κόμβου και την ώρα. Εκτελούμε ένα βρόγχο while που προχωράει σε επόμενες λύσεις του κατηγορήματος, δηλαδή επόμενους γείτονες, μέχρι να μην υπάρχει άλλη λύση του κατηγορήματος, δηλαδή να μην υπάρχει άλλος γείτονας του τρέχοντος. Για κάθε γείτονα το κατηγόρημα μας επιστρέφει και το κόστος της μετάβασης από τον τρέχοντα κόμβο προς το συγκεκριμένο γείτονα. Τώρα χρειάζεται εκτός από τα κόστη να κρατάμε και το συνολικό χρόνο που απαιτείται για να φτάσουμε σε κάθε κόμβο προκειμένου να επιτευχθεί το τρέχον κόστος. Τις τούπλες (node, cost, time) του μετώπου τις κρατάμε σε priority queue με το κόστος να αποτελεί priority και το χρόνο να αποτελεί παρεπόμενο. Κρατάμε σε παράλληλους πίνακες τα τρέχοντα βέλτιστα κόστη και τους τρέχοντες αντίστοιχους χρόνους και αποστάσεις. Και πάλι στη φάση της χαλάρωσης βελτιώνουμε τα τρέχοντα κόστη αν βρεθεί καλύτερη διαδρομή από την τρέχουσα και ενημερώνουμε τον πατέρα, το χρόνο και την απόσταση. Αν βγάλουμε από το priority queue ένα κόμβο και έχει τιμή κόστους χειρότερη από την τρέχουσά του ή χειρότερη από το κόστος της καλύτερης ως τώρα υπολογισμένης διαδρομής προς το στόχο δεν τον ξανα-αναπτύσσουμε, ενώ αν έχει μικρότερη ή ίση (δεν έτυχε στη συγκεκριμένη άσκηση) τον αναπτύσσουμε και μπορεί να προκύψουν έτσι εν δυνάμει πολλά εναλλακτικά μονοπάτια. Αυτό υποστηρίζεται από το συγκεκριμένο κώδικα, όπως ακριβώς στον κώδικα της πρώτης άσκησης τον οποίο τροποιήσαμε.

Η java χρησιμοποιεί και άλλα κατηγορήματα της prolog προκείμενου να έχει πρόσβαση στη γνώση. Καλεί query για να πάρει τα id των κόμβων στους οποίους αντιστοιχίζονται ο πελάτης και τα ταξί, query taxi_adequacy για να βρει τα υποψήφια ταξί, haversine απόστασης των διαφόρων κόμβων από τον κόμβο στόχο του A* κατά τη δημιουργία των ευριστικών εκτιμήσεων, ranking για να βαθμολογήσει τα ταξί αφού υπολογίσει τις βέλτιστες διαδρομές, client για να πάρει τα χαρακτηριστικά του πελάτη και query για να πάρει τα X και Y των κόμβων κατά την κατασκευή των KML αρχείων με τα βέλτιστα μονοπάτια. Έχει υπάρξει παραμετροποίηση του κώδικα επίλυσης ώστε να τρέχει σε μορφή συνάρτησης που καλούμε μια φορά για τα δεδομένα της εκφώνησης και μια φορά για τα δεδομένα της επιλογής μας.

<u>Η ευριστική εκτίμηση</u> που επιλέξαμε είναι εφαρμογή του τύπου για το χρηματικό κόστος J η οποία να δίνει τιμές κοντά στις πραγματικές όμως υποεκτιμάει το κόστος, όπως θέλουμε στον A* . Αυτό γίνεται βάζοντας στον τύπο του κόστους ως εκτιμούμενη διανυόμενη διαδρομή την haversine απόσταση, η οποία είναι σίγουρα μικρότερη της πραγματικής και ως ταχύτητα 100km/h , η οποία είναι ίση με τη μέγιστη επιτρεπτή στο συγκεκριμένο κόσμο.

Εκτέλεση προγράμματος και σχολιασμός αποτελεσμάτων

Αρχικά γίνεται μια σχετικά χρονοβόρα διαδικασία που κρατάει περίπου ένα λεπτό κατά την οποία γίνονται τα consults των αρχείων prolog.

Εκτέλεση **με τα δεδομένα των csv αρχείων** για ταξί και πελάτη:

Τα ταξί με id 130 και 170 απορρίφθηκαν γιατί δεν ήταν available, τα 110 και 180 απορρίφθηκαν γιατί ο πελάτης μιλούσε ελληνικά ενώ οι οδηγοί τους μόνο αγγλικά και το ταξί 100 απορρίφθηκε γιατί χωρούσε 1-2 άτομα και ο πελάτης ζήτησε μεταφορά τριών.

Τα υπόλοιπα ταξί ταξινομήθηκαν ως προς το χρόνο αναμονής του πελάτη μέχρι τα ταξί να έρθουν σε αυτόν και παρουσιάζονται τα κ=5 πρώτα. Το κ=5 δώθηκε ως είσοδος στο πρόγραμμα java. Στη συνέχεια

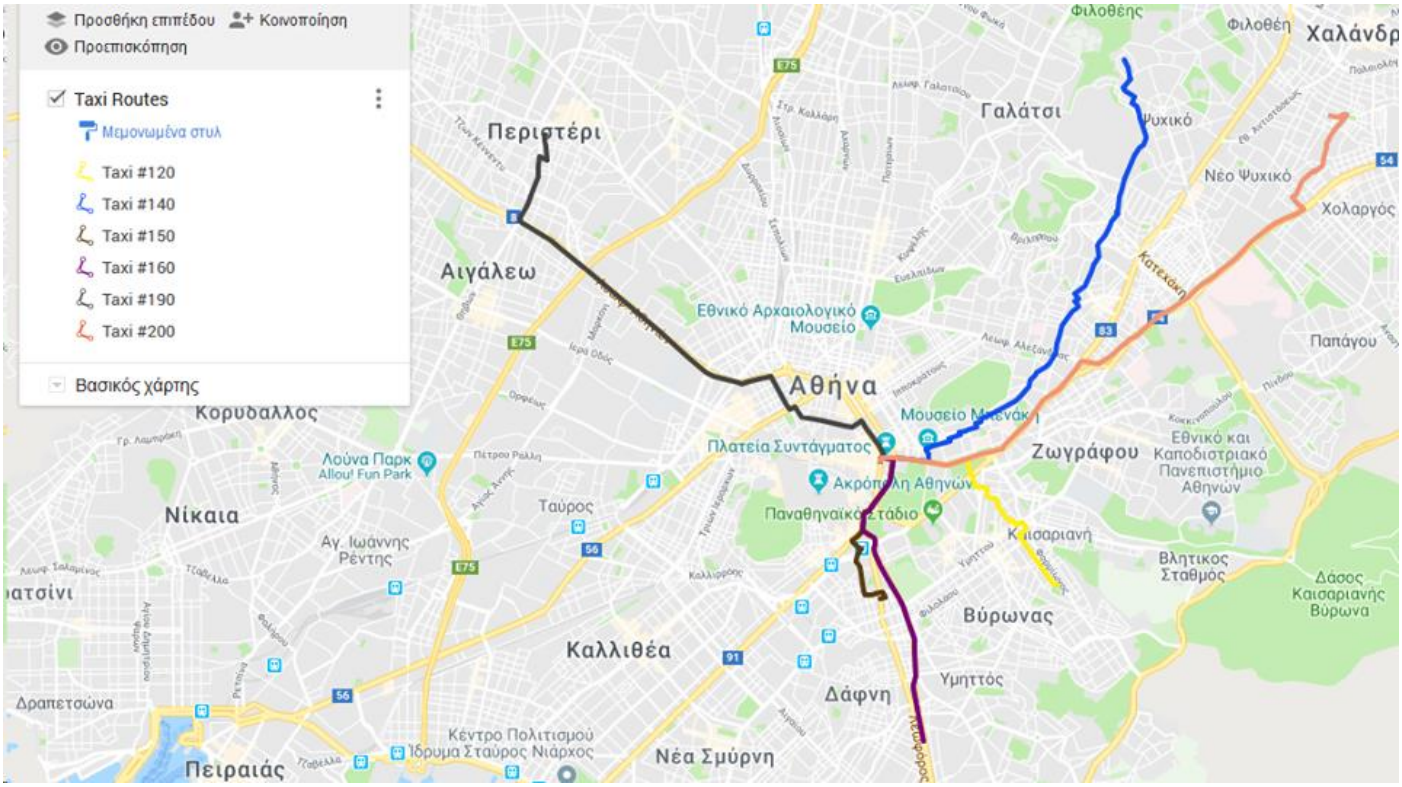
παρουσίαζονται τα κ πρώτα ταξινομημένα με βάση το χρόνο αναμονής και το rating. Φαίνεται για κάθε ταξί το χρηματικό κόστος της κούρσας, η διάρκεια, ο χρόνος αναμονής και η απόσταση.

```
C:\Users\USER\Desktop\AI_2>java A_star
Consulting Prolog files... 100%
Consults done!
Executing A*...
A* done.

Please insert k (number of taxis whose info will be displayed). k = 5

Rank by proximity:
Taxi #150:
        Ranking = 98,27
        Cost = 7,63 Euros
        Waiting time = 4 minutes
        Time to destination = 10 minutes
        Total time = 15 minutes
        Distance = 2,401 Km
Taxi #120:
        Ranking = 98,31
        Cost = 7,63 Euros
        Waiting time = 5 minutes
        Time to destination = 10 minutes
        Total time = 16 minutes
        Distance = 3,318 Km
Taxi #160:
        Ranking = 48,77
        Cost = 7,63 Euros
        Waiting time = 7 minutes
        Time to destination = 10 minutes
        Total time = 18 minutes
        Distance = 3,837 Km
Taxi #190:
        Ranking = 38,37
        Cost = 7,63 Euros
        Waiting time = 11 minutes
        Time to destination = 10 minutes
        Total time = 22 minutes
        Distance = 6,855 Km
Taxi #140:
        Ranking = 32,01
        Cost = 7,63 Euros
        Waiting time = 13 minutes
        Time to destination = 10 minutes
        Total time = 24 minutes
        Distance = 6,912 Km
------------------------------------------------
```

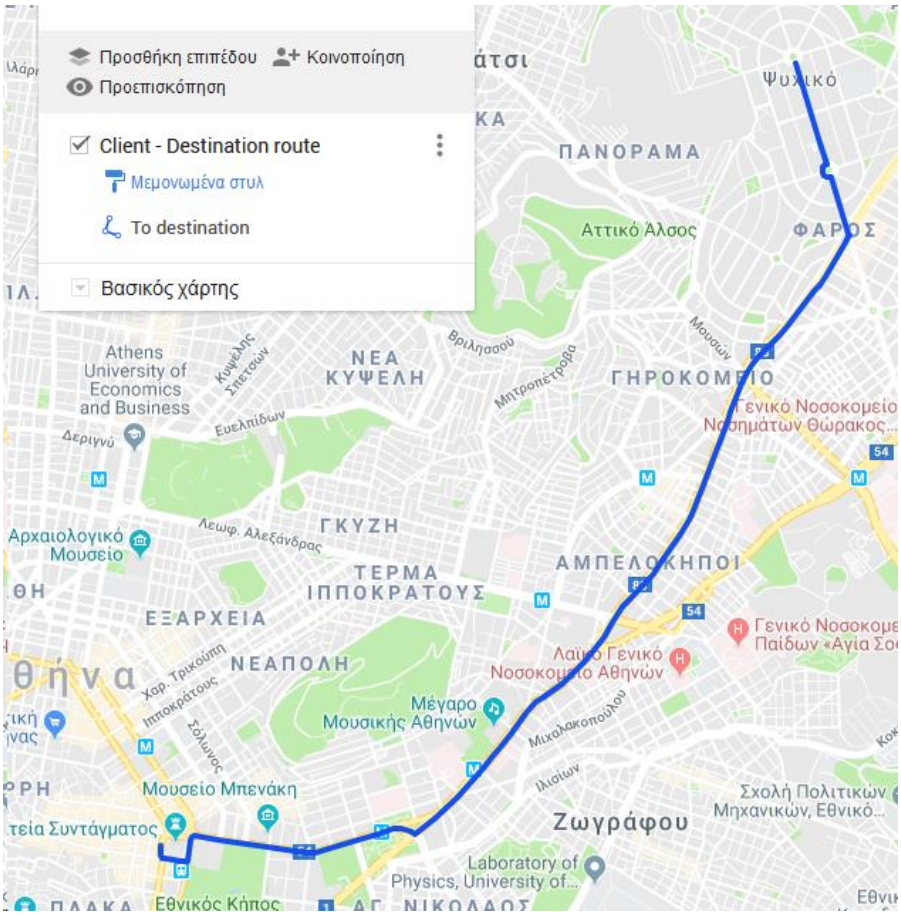
```
------------------------------------------------

Rank by optimality:
Taxi #120:
        Ranking = 98,31
        Cost = 7,63 Euros
        Waiting time = 5 minutes
        Time to destination = 10 minutes
        Total time = 16 minutes
        Distance = 3,318 Km
Taxi #150:
        Ranking = 98,27
        Cost = 7,63 Euros
        Waiting time = 4 minutes
        Time to destination = 10 minutes
        Total time = 15 minutes
        Distance = 2,401 Km
Taxi #160:
        Ranking = 48,77
        Cost = 7,63 Euros
        Waiting time = 7 minutes
        Time to destination = 10 minutes
        Total time = 18 minutes
        Distance = 3,837 Km
Taxi #190:
        Ranking = 38,37
        Cost = 7,63 Euros
        Waiting time = 11 minutes
        Time to destination = 10 minutes
        Total time = 22 minutes
        Distance = 6,855 Km
Taxi #140:
        Ranking = 32,01
        Cost = 7,63 Euros
        Waiting time = 13 minutes
        Time to destination = 10 minutes
        Total time = 24 minutes
        Distance = 6,912 Km
```

Παρατηρούμε ότι τα αποτελέσματα είναι λογικοφανή.

Στο αρχείο doc.klm αποθηκεύουμε τα βέλτιστα μονοπάτια από τα διάφορα ταξί στον πελάτη. Παρακάτω βλέπουμε screenshot από το χάρτη. Μόνο 6 ταξί ήταν adequate.



Στο αρχείο doc_dest.kml φαίνεται η βέλτιστη διαδρομή από τον πελάτη προς τον προορισμό. Παρακάτω βλέπουμε αυτή τη διαδρομή στο χάρτη.



Εκτέλεση του αλγορίθμου με **τα δικά μας δεδομένα** για πελάτη και ταξί: Η διαδρομή αυτή τη φορά είναι νυκτερινή (διπλή ταρίφα) και έχουμε άλλες θέσεις ταξί πελάτη και προορισμού.

Και πάλι έχουμε μια αρχική καθυστέρηση που οφείλεται σε unconsults και consults που γίνονται.

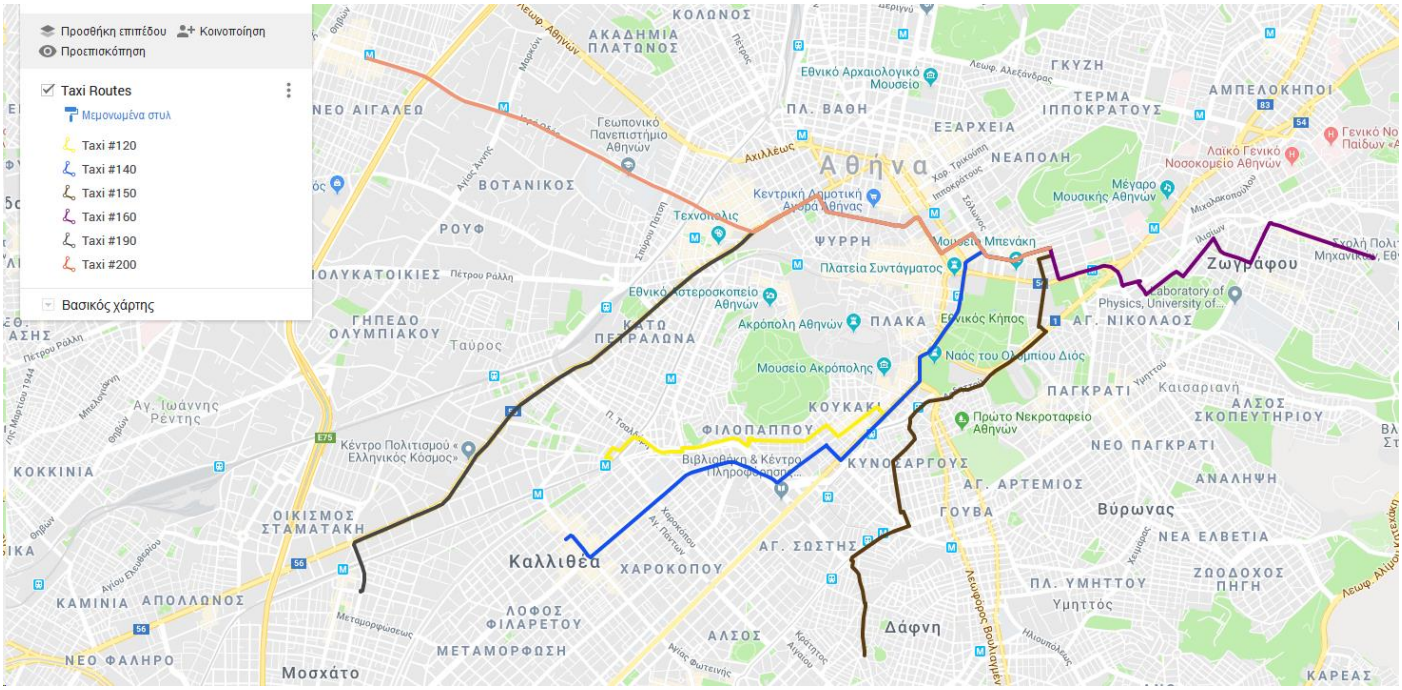
Τα κ=6 πρώτα ταξί παρουσιάζονται όπως προηγουμένως (δώσαμε κ=15 οπότε εμφάνισε όσα υπήρχαν).

```
Partial reconsulting...

Partial reconsulting done!
Executing A*...
A* done.

Please insert k (number of taxis whose info will be displayed). k = 15

Rank by proximity:
Taxi #160:
        Ranking = 61,08
        Cost = 15,90 Euros
        Waiting time = 5 minutes
        Time to destination = 13 minutes
        Total time = 19 minutes
        Distance = 3,500 Km
Taxi #140:
        Ranking = 49,63
        Cost = 15,90 Euros
        Waiting time = 8 minutes
        Time to destination = 13 minutes
        Total time = 21 minutes
        Distance = 5,389 Km
Taxi #150:
        Ranking = 49,41
        Cost = 15,90 Euros
        Waiting time = 8 minutes
        Time to destination = 13 minutes
        Total time = 22 minutes
        Distance = 4,433 Km
Taxi #120:
        Ranking = 61,87
        Cost = 15,90 Euros
        Waiting time = 8 minutes
        Time to destination = 13 minutes
        Total time = 22 minutes
        Distance = 4,895 Km
Taxi #200:
        Ranking = 26,07
        Cost = 15,90 Euros
        Waiting time = 11 minutes
        Time to destination = 13 minutes
        Total time = 25 minutes
        Distance = 6,214 Km
Taxi #190:
        Ranking = 33,12
        Cost = 15,90 Euros
        Waiting time = 13 minutes
        Time to destination = 13 minutes
        Total time = 26 minutes
        Distance = 7,341 Km
--------------------------------------------
```

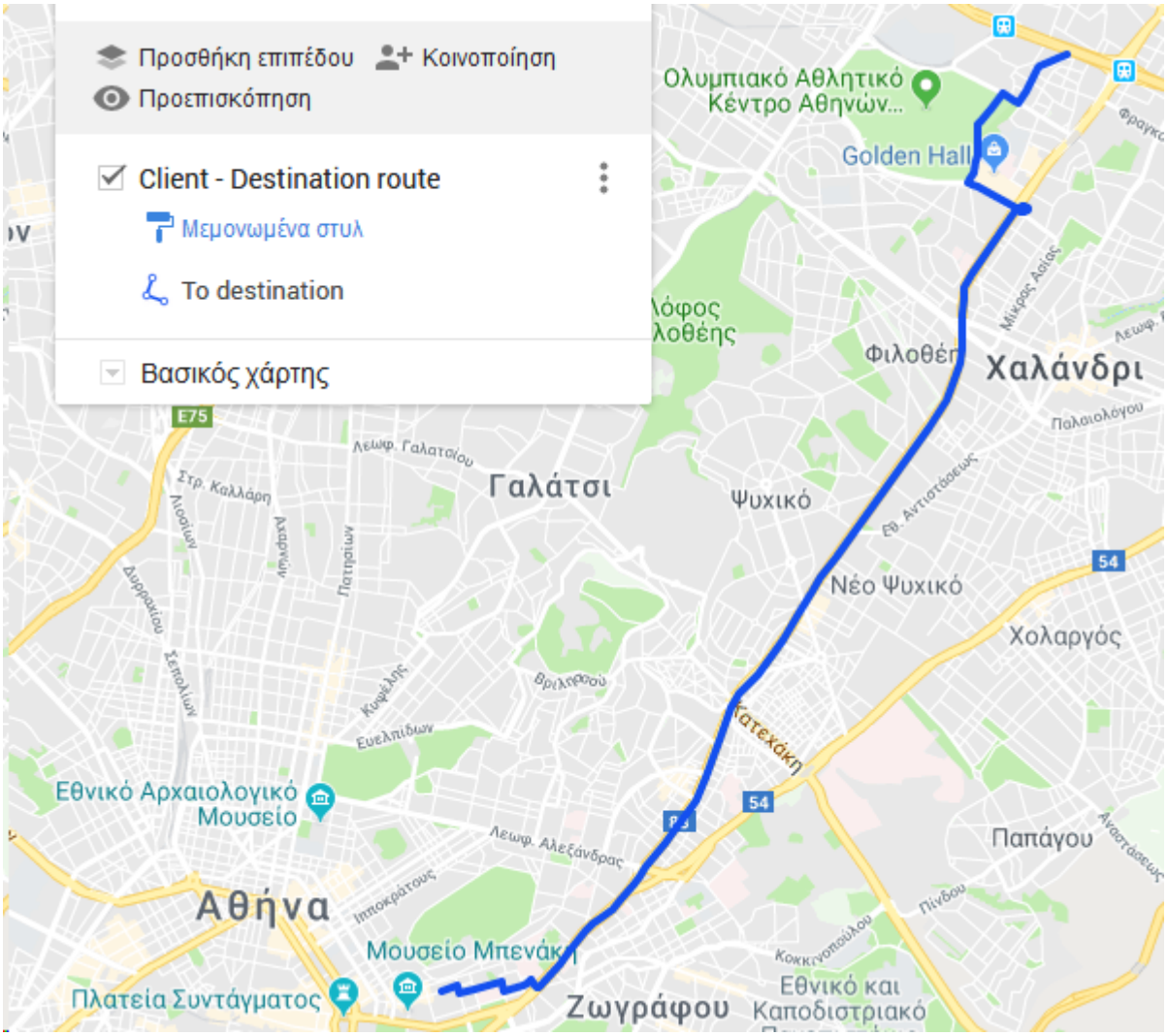
```
--------------------------------------------

Rank by optimality:
Taxi #120:
        Ranking = 61,87
        Cost = 15,90 Euros
        Waiting time = 8 minutes
        Time to destination = 13 minutes
        Total time = 22 minutes
        Distance = 4,895 Km
Taxi #160:
        Ranking = 61,08
        Cost = 15,90 Euros
        Waiting time = 5 minutes
        Time to destination = 13 minutes
        Total time = 19 minutes
        Distance = 3,500 Km
Taxi #140:
        Ranking = 49,63
        Cost = 15,90 Euros
        Waiting time = 8 minutes
        Time to destination = 13 minutes
        Total time = 21 minutes
        Distance = 5,389 Km
Taxi #150:
        Ranking = 49,41
        Cost = 15,90 Euros
        Waiting time = 8 minutes
        Time to destination = 13 minutes
        Total time = 22 minutes
        Distance = 4,433 Km
Taxi #190:
        Ranking = 33,12
        Cost = 15,90 Euros
        Waiting time = 13 minutes
        Time to destination = 13 minutes
        Total time = 26 minutes
        Distance = 7,341 Km
Taxi #200:
        Ranking = 26,07
        Cost = 15,90 Euros
        Waiting time = 11 minutes
        Time to destination = 13 minutes
        Total time = 25 minutes
        Distance = 6,214 Km
Successful termination! (Press Enter to exit)
```

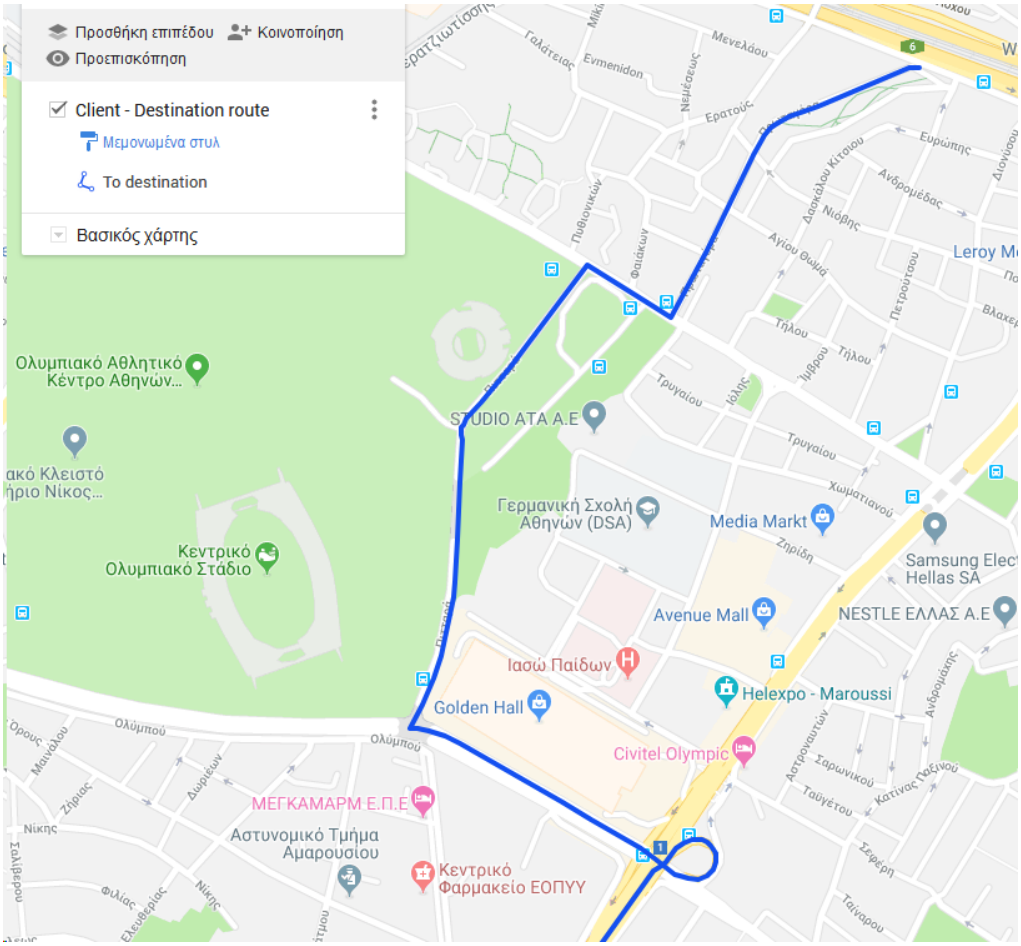
Βλέπουμε και πάλι πως τυπώνονται οι σωστές αποστάσεις χρόνοι και κόστη.

Στο doc_custom.kml αποθηκεύονται οι βέλτιστες διαδρομές από τα ταξί στον πελάτη, οι οποίες φαίνονται στον παρακάτω χάρτη:



Στο αρχείο doc_dest_custom.kml αποθηκεύεται η βέλτιστη διαδρομή από τον πελάτη προς την προορισμό και τη βλέπουμε παρακάτω στο χάρτη.





Εδώ βλέπουμε λεπτομέρεια από τη διαδρομή προς τον προορισμό η οποία περνάει από ένα `trunk_link`.