Τεχνητή Νοημοσύνη-2^o Θέμα

Ιωάννης Αναγνωστίδης 03115094
Νικόλαος Μπούας 03115005

10 Φεβρουαρίου 2019

Περιεχόμενα

1 Εισαγωγή και Σκοπός **1**

2 Υλοποίηση **2**

3 Εκτέλεση **3**

4 Αναφορές **5**

1 Εισαγωγή και Σκοπός

Ο σκοπός της συγκεκριμένης άκσησης είναι η δημιουργία μιας εφαρμογής που θα προτείνει σε έναν πελάτη ένα σύνολο από διαθέσιμα ταξί λαμβάνοντας υπ΄ όψη πολλαπλούς παραμέτρους. Συγκεκριμένα, χρησιμοποιώντας δεδομένα σχετικά με το οδικό δίκτυο θα προσδιορίσουμε τις συντομότερες χρονικά διαδρομές, έχοντας λάβει υπ΄ όψη παραμέτρους όπως η κατεύθυνση, η κίνηση καθώς και το είδος του δρόμου. Στη συνέχεια, για κάθε ένα από τα διαθέσιμα ταξί θα χρησιμοποιούμε πληροφορίες που σχετίζονται με την ποιότητα των προσφερόμενων υπηρεσιών, όπως είναι η ικανότητα επικοινωνίας του πελάτη με τον οδηγό, ώστε να εξάγουμε μια βαθμολογία για κάθε ένα από αυτά. Αξίζει να αναφερθεί ότι ανάγνωση των δεδομένων καθώς και η προκαταρκτική επεξεργασία τους πραγματοποιείται από ένα πρόγραμμα οδηγός γραμμένο σε *Java*, ωστόσο η αναπαράσταση και η επεξεργασία της γνώσης βασίζεται σε μία βάση δεδομένων *Prolog*.

2 Υλοποίηση

Η διαδικασία εκτέλεσης του αλγορίθμου A^* παραμένει κατά βάση η ίδια με εκείνη του προηγούμενη προγραμματιστικού θέματος, ενώ οι διαφορές εντοπίζονται στον τρόπο που επεξεργάζονται τα δεδομένα. Συγκεκριμένα, η υλοποίηση αυτή βασίζεται σε κατηγορήματα της $Prolog$ για να αποθηκεύσουμε πληροφορίες και να πραγματοποιήσουμε επερωτήσεις στην αντίστοιχη βάση δεδομένων. Έτσι, τα βασικά σημεία της υλοποίησης είναι τα ακόλουθα:

Πρώτα απ΄ όλα, γενικεύουμε το κόστος διάσχισης ενός δρόμου ως εξής: Ορίζουμε έναν συντελεστή κόστους που είναι μεγαλύτερος ή ίσος από το 1 και προκύπτει με βάση την κίνηση και το μέγεθος του δρόμου. Με τον τρόπο αυτόν, εξασφαλίζουμε ότι η ευριστική προσφέρει μια εκτίμηση που είναι μικρότερη ή ίση από την πραγματική. Στο πλαίσιο αυτό, ορίζουμε ένα κατηγόρημα $trafficCost(RoadId, TrafficCost)$ που για κάθε δρόμο περιέχει την πληροφορία για την κίνησή του. Συγκεκριμένα, χρησιμοποιώντας ένα κατηγόρημα $findTrafficCost$ θεωρήσαμε ότι το κόστος είναι 2, για υψηλή κίνηση και 1 αλλιώς, ενώ η απουσία πληροφορίας αναιρεί την εισαγωγή του όρου $trafficCost$.

Στη συνέχεια, για να ελέγξουμε αν ένας δρόμος μπορεί να διασχιστεί, ορίσαμε ένα κατηγόρημα $isDrivable$, αξιοποιώντας τις πληροφορίες του αρχείου $lines.csv$. Έτσι, ορίσαμε ένα κατηγόρημα $isDrivable$, στο οποίο με βάση το είδος του δρόμου, προσδιορίζουμε αν είναι δυνατή η διάσχισή του από ένα ταξί. Πιο συγκεκριμένα, λαμβάνουμε υπ΄ όψη αν ο δρόμος είναι $railway, boundary, waterway, busway$, καθώς και αν πρόκειται για $path, pedestrian, footway$ και $steps$, οπότε και δεν είναι δυνατή η διάσχισή του. Στη συνέχεια, είμαστε σε θέση να προσθέσουμε όρους της μορφής $direction(RoadId, Direction)$, που ορίζουν την φορά διάσχισης του δρόμου, με τους αριθμούς 1 και -1 να ορίζουν μονόδρομους με τις κατευθύνσεις που ορίζονται από την φορά των κόμβων που μας δίνονται και ο αριθμός 0 να υποδηλώνει ότι πρόκειται για δρόμο διπλής κυκλοφορίας.

Το επόμενο βήμα είναι να προσδιορίσουμε το κόστος του κάθε δρόμου με την γενικευμένη έννοια για κάθε δρόμο. Για τον υπολογισμό αυτόν, θα λάβουμε υπ΄ όψη πληροφορίες όπως είναι η ύπαρξη διοδίων, η μέγιστη επιτρεπόμενη ταχύτητα, ο φωτισμός αλλά και η κίνηση.

Τέλος, επεξεργαζόμαστε το αρχείο $nodes.csv$ ώστε να προσδιορίσουμε όλες τις αποδεκτές εναλλακτικές κινήσεις από ένα δεδομένο σημείο. Για να πετύχουμε αυτό, προσθέτουμε στη βάση δεδομένων της $prolog$ κατηγορήματα της μορφής $nextNode(Lat1, Long1, Lat2, Long2, RoadId)$, δηλαδή, καθώς διασχίζουμε το αρχείο $nodes.csv$, αν εντοπίσουμε δύο διαδοχικούς κόμβους με το ίδιο id προσθέτουμε τους κατάλληλους όρους ανάλογα με τη φορά διάσχισης. Κατ΄ αυτόν τον τρόπο, έχουμε όλες τις δυνατές μεταβάσεις που αποτελούν τον πυρήνα της υλοποίησης του A^*.

Όπως γίνεται αντιληπτό, οι παραπάνω επεξεργασία των δεδομένων για ένα μεγάλο οδικό δίκτυο προσθέτει σημαντική χρονική επιβάρυνση στην εφαρμογή μας. Για το λόγο αυτό, σε μία πραγματική εφαρμογή θα ήταν αναγκαίο να έχουμε πραγματοποιήσει ένα $pre-process$ στα δεδομένα για το στατικό κομμάτι της εμπλεκόμενης πληροφορίας, αν όχι για την ολότητα του οδικού δικτύου για έναν βασικό κορμό του. Έτσι, κατά τη διάρκεια ενός $query$ θα ήταν αρκετό να λάβουμε υπ΄ όψη και ορισμένες δυναμικές πληροφορίες όπως είναι ο χρόνος που εκφράζεται το αίτημα για εξυπηρέτηση από τον πελάτη.

Στο επόμενο τμήμα της υλοποίησης αξιοποιούμε τις πληροφορίες που μας δίνονται για τα

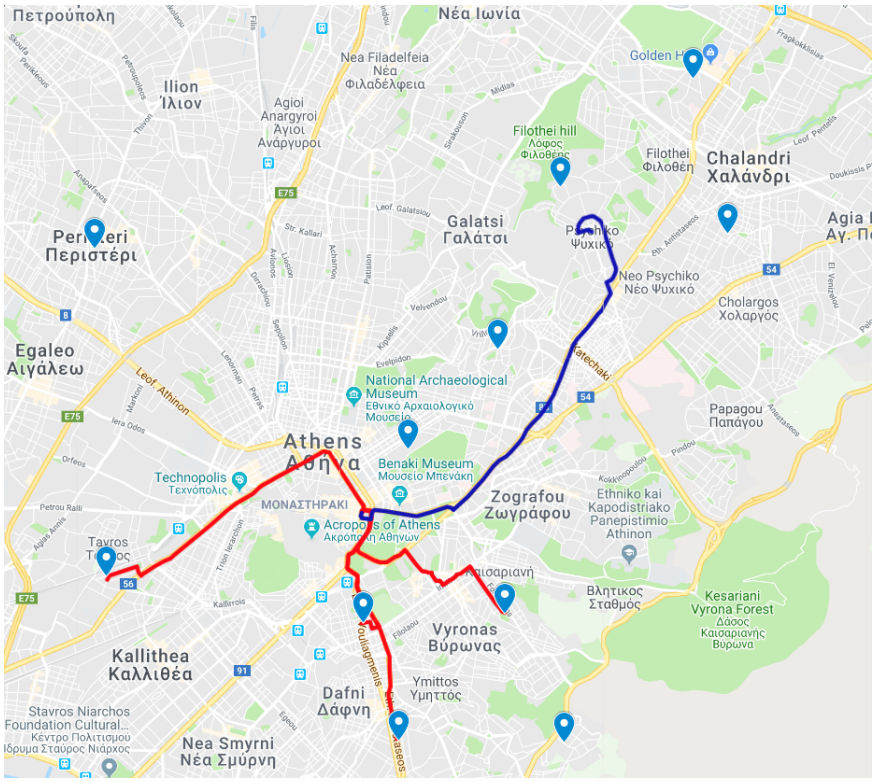
ταξί στο αρχείο $taxis.csv$ για να ελέγξουμε αν το ταξί έχει την απαιτούμενη χωρητικότητα για να μετακινήσει τους πελάτες, καθώς επίσης και για τον προσδιορισμό της επικαιροποιημένης βαθμολογίας του με βάση τη γλώσσα ομιλίας του πελάτη. Για να πετύχουμε αυτό ορίζουμε τα κατηγορήματα $isValidTaxi$ και $findRating$ αντίστοιχα, που όπως γίνεται αντιληπτό λαμβάνουν υπ΄ όψη και τις πληροφορίες που σχετίζονται με τον πελάτη στο αρχείο $client.csv$. Κατ΄ αυτόν τον τρόπο, αφού πραγματοποιηθεί η επεξεργασία και η αποθήκευση πληροφοριών σχετικά με το οδικό δίκτυο εκτελούμε τον αλγόριθμο $A*$ για κάθ΄ ένα από τα ταξί με τελικό προορισμό την τοποθεσία του πελάτη, ή στην περίπτωση που δεν βρίσκεται σε προσβάσιμη τοποθεσία στον κοντινότερο προσβάσιμο κόμβο προς τον πελάτη. Αξίζει να αναφερθεί επιπλέον ότι έχουμε λάβει υπ΄ όψη για κάθε ταξί αν είναι σε θέση να πραγματοποιήσει διαδρομές μεγάλης απόστασης, καθώς επίσης και αν είναι διαθέσιμο τη στιγμή που πραγματοποιήθηκε το αίτημα. Σ΄ αντίθερη περίπτωση είναι φανερό ότι μπορούμε να αποκλείσουμε το συγκεκριμένο ταξί από την υλοποίηση, όπως θα φανεί και παρακάτω στο κομμάτι της υλοποίησης.

3 Εκτέλεση

Σε πρώτο βήμα, για τα δεδομένα που αφορούν τα ταξί και τον πελάτη που μας δόθηκαν ακολουθούμε τη διαδικασία που έχουμε περιγράψει για να προσδιορίσουμε τα διαθέσιμα ταξί και απ΄ αυτά τα συντομότερα χρονικά μονοπάτια. Επιπλέον, το πρόγραμμά μας για καθ΄ ένα απ΄ αύτα εκτυπώνει και μια βαθμολογία με βάση την συνολική ποιότητα των προσφερόμενων υπηρεσιών. Έτσι, θα πάρουμε τα ακόλουθα αποτελέσματα:

- Ως προς τον χρόνο που απαιτείται για να φτάσουν στον πελάτη έχουμε:
  - Το ταξί με id 150 είναι το ταχύτερο
  - Το ταξί με id 160 είναι 35% πιο αργό από το ταχύτερο
  - Το ταξί με id 110 είναι 102% πιο αργό από το ταχύτερο
  - Το ταξί με id 120 είναι 107% πιο αργό από το ταχύτερο
- Ως προς τη βαθμολογία των παραπάνω ταξί:
  - Το ταξί με id 120 έχει βαθμολογία 9.2
  - Το ταξί με id 150 έχει βαθμολογία 7.3
  - Το ταξί με id 160 έχει βαθμολογία 6.1
  - Το ταξί με id 110 έχει βαθμολογία 5.0

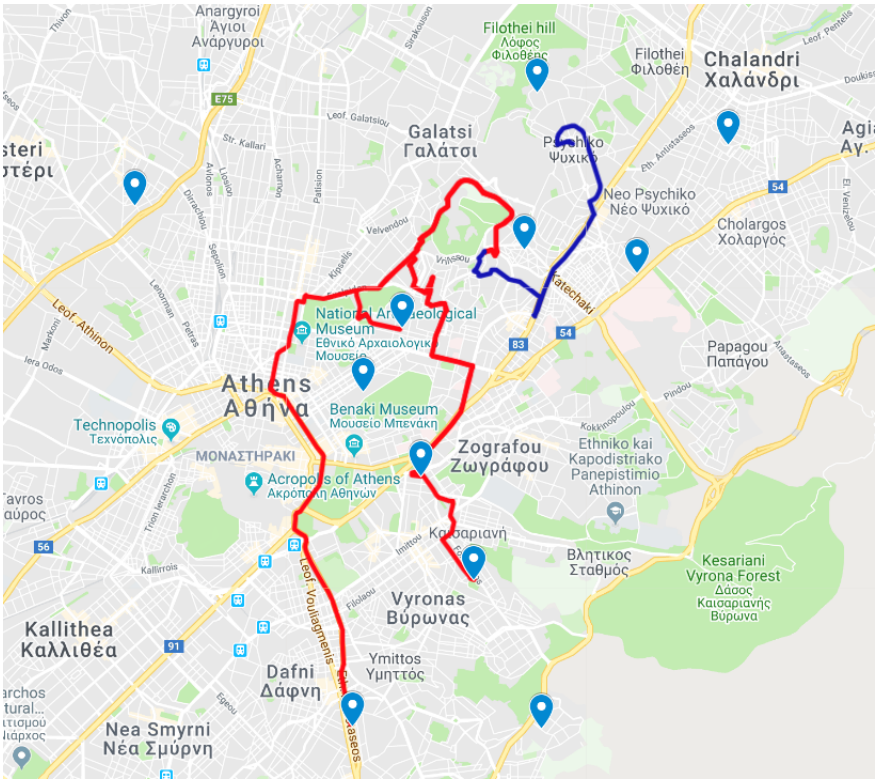
Οι διαδρομές που θα ακολουθήσουν τα παραπάνω ταξί φαίνονται και παρακάτω με χρήση της υπηρεσίας $Google - Maps$, όπου με κόκκινο φαίνονται οι βέλτιστες διαδρομές των ταξί και με μπλε η διαδρομή από τον πελάτη στον προορισμό:



Όπως είναι φανερό, από το σύνολο των ταξί μόνο τα 4 είναι σε θέση να ικανοποιήσουν το αίτημα του πελάτη, όπως φαίνεται και στην παραπάνω αναπαράσταση, όπου τα *markers* αντιστοιχούν στις αρχικές θέσεις των ταξί.

Στη συνέχεια, εισάγουμε δικά μας δεδομένα μέσω των αρχείων *newTaxis.csv* και *newClient.csv* το οποία έχουν ενσωματωθεί στο παραδοτέο. Έτσι, θα πάρουμε τα ακόλουθα αποτελέσματα:

- Ως προς τον χρόνο που απαιτείται για να φτάσουν στον πελάτη έχουμε:
  - Το ταξί με *id* 110 είναι το ταχύτερο
  - Το ταξί με *id* 150 είναι 45% πιο αργό από το ταχύτερο
  - Το ταξί με *id* 160 είναι 66% πιο αργό από το ταχύτερο
  - Το ταξί με *id* 120 είναι 73% πιο αργό από το ταχύτερο
- Ως προς τη βαθμολογία των παραπάνω ταξί:
  - Το ταξί με *id* 160 έχει βαθμολογία 6.1
  - Το ταξί με *id* 120 έχει βαθμολογία 5.5
  - Το ταξί με *id* 110 έχει βαθμολογία 5.2
  - Το ταξί με *id* 150 έχει βαθμολογία 3.3



4 Αναφορές

Για την υλοποίηση της παραπάνω εφαρμογής αξίζει να αναφέρουμε τους παρακάτω ιστιοτόπους:

- *stackoverflow.com*: Για την αποσφαλμάτωση του προγράμματος και τον προσδιορισμό της ευκλίδειας απόστασης δύο σημείων με πεδία γεωμετρικές συντεταγμένες της μορφής *latitude* και *longitude*.

- *docs.oracle.com*: Για την αναζήτηση βιβλιοθηκών που υλοποιούν τις δομές δεδομένων που χρησιμοποιήσαμε.

- *google.com/maps*: Για την οπτικοποίηση των διαδρομών που προκύπτουν από την εκτέλεση του αλγορίθμου.

- *jiprolog.com*: Για την αξιοποίηση της βιβλιοθήκης *jiprolog* που ουσιαστικά καθιστά δυνατή την επικοινωνία του προγράμματος οδηγού σε *Java* με τη βάση δεδομένων της *Prolog*.

- *swi − prolog.org*: Για την αποσφαλμάτωση και την εμπέδωση των βασικών αρχών του λογικού προγραμματισμού.