Τεχνητή Νοημοσύνη - Αναφορά Δεύτερου Project

Ηλέκτρα Σκεπετάρη - ΑΜ: 03113074

Α. Γενικός Σχεδιασμός Συστήματος

Στο συγκεκριμένο θέμα κληθήκαμε να κατασκευάσουμε, μέσω ενός προγράμματος Java, τον κορμό μιας ευφυούς υπηρεσίας εξυπηρέτησης πελατών ταξί. Σκοπός του θέματος δηλαδή είναι να εξομοιωθεί μια υπηρεσία που για έναν πελάτη που βρίσκεται σε συγκεκριμένες γεωγραφικές συντεταγμένες, επιλέγει το ταξί που μπορεί να μεταβεί πιο γρήγορα στη θέση του μεταξύ των διαθέσιμων ταξί που βρίσκονται σε ένα αρχείο τη στιγμή της υποτιθέμενης κλήσης από τον πελάτη, λαμβάνοντας υπ'όψιν και άλλα κριτήρια που καθιστούν το ταξί κατάλληλο.

Το πρόγραμμα "μαθαίνει" τους κανόνες που ορίζουν τι είναι επιτρεπτό και τι όχι στον κόσμο μας απο τον Didaktor (Didaktor.java) που παίρνει τους κανόνες που έχουμε ορίσει σε ένα πρόγραμμα prolog που γνωρίζει τις ιδιότητες του κόσμου μας, τον kosmognwsti (kosmognwstis.pl). Σε αυτό το πρόγραμμα έχουν οριστεί οι εξής κανόνες που ρυθμίζουν τον κόσμο μας:

- ★ Αν ο δρόμος έχει τα γνωρίσματα bus_guideway, escape, raceway δεν είναι για ΙΧ.
- ★ Το ΙΧ έχει σαν μέση ταχύτητα το 90% του ορίου του δρόμου. Έτσι τελικά μπορούμε να λάβουμε υπόψιν τον χρόνο που θα χρειαστεί, και όχι τόσο την απόσταση καθώς σε κεντρικούς δρόμους και λεωφόρους τα όρια είναι μεγαλύτερα.
- ★ Η κίνηση στους δρόμους έχει βάρος, 1 αν δεν υπάρχει, 0.9 αν είναι μέτρια και 0.7 αν είναι υψηλή. Έτσι η τελική συνάρτηση της προτεραιότητας θα επιστρέψει χειρότερα αποτελέσματα αν υπάρχει κίνηση.
- ★ Ο φωτισμός τον δρόμων έχει βάρος 1 αν είναι καλός, 0.9 αν είναι μέτριος και 0.8 αν είναι χαμηλός. Έτσι η τελική συνάρτηση της προτεραιότητας θα επιστρέψει χειρότερα αποτελέσματα αν δεν υπάρχει φωτισμός.
- ★ Ο φωτισμός τον δρόμων είναι καλός τις πρωινές και μεσημεριανές ώρες, μέτριος τις βραδυνές, σε περίπτωση που υπάρχει φωτισμός στον συγκεκριμένο δρόμο και χαμηλός τις βραδινές ώρες σε περίπτωση που δεν υπάρχει φωτισμός. Έτσι επηρεάζεται το αποτέλεσμα εκεί που υπάρχει ανάγκη, ενώ το πρωί δεν αλλάζει κάτι.
- ★ Τα αυτοκίνητα έχουν πρόσβαση σε δρόμους που δεν έχουν τα γνωρίσματα service, pedestrian, track, bus_guideway, escape, raceway, footway, bridleway, steps, path, cycleway, waterway, railway και ταυτόχρονα έχουν το γνώρισμα access. Αυτό συμβαίνει γιατί το ταξί θα πρέπει να περνάει απο δρόμους που το επιτρέπει ο κ.ο.κ.
- ★ Η προτεραιότητα των αυτοκινήτων ορίζεται απο το γινόμενο που προκύπτει απο τον συντελεστή του φωτισμού, της μέσης ταχύτητας, του συντελεστή κίνησης και του επιτρεπτού ορίου ταχύτητας, ενώ είναι 0 σε περίπτωση που τα αμάξια δεν έχουν πρόσβαση στον δρόμο.

Αναλυτικά τα ορίσματα των κανόνων του κοσμογνώστη …

```
average_speed(0.9).

traffic_weight(R, T, 1) :- not(traffic(R, T, _)).
traffic_weight(R, T, 0.9) :- traffic(R, T, medium).
traffic_weight(R, T, 0.7) :- traffic(R, T, high).

lighting_weight(good, 1).
lighting_weight(medium, 0.9).
lighting_weight(low, 0.8).

lighting(_, morning, good).
lighting(_, noon, good).
lighting(R, afternoon, medium) :- lit(R).
lighting(R, afternoon, low) :- not(lit(R)).

forCars(R) :- not(service(R)), not(pedestrian(R)), not(track(R)), not(bus_guideway(R)),
not(escape(R)), not(raceway(R)), not(footway(R)), not(bridleway(R)), not(steps(R)),
not(path(R)), not(cycleway(R)), not(waterway(R)), not(railway(R)).
accessible(R) :- access(R), forCars(R).

priority(R, T, Z) :- accessible(R), maxspeed(R, MS), average_speed(A), traffic_weight(R, T,
TW), lighting(R, T, L), lighting_weight(L, LW), Z is MS * A * TW * LW.
priority(R, _, 0) :- not(accessible(R)).
```

Πίνακας 1: Κοσμογνώστης

Το πρόγραμμα σε java, Didaktor διαβάζει το αρχείο εισόδου με τις πληροφορίες για τον χάρτη nodes.csv και μέσω της μηχανής του JIProlog που μας δώθηκε στέλνει τα δεδομένα στην prolog για να φτιαχτούν οι κανόνες. Τελικά μαθαίνει τον κόσμο και φτίαχνει όλους τους κανόνες που τον ορίζουν και τους αποθηκεύει στο αρχείο kosmos.pl.

Στην τελική φάση του προγράμματος η εφαρμογή που χρησιμοποιούν οι πελάτες είναι γραμμένη στο Tarifbeat.java. Εκεί διαβάζουμε απο τα αρχεία εισόδου client.csv, taxis.csv τις πληροφορίες για την θέση και τα χαρακτηριστικά του πελάτη και των ταξί της υπηρεσίας. Για τις θέσεις λοιπόν τρέχει τον αλγόριθμο Astar ο οποίος σαν ευριστική συνάρτηση πλέον παίρνει το αποτέλεσμα που θα δώσει ο κανόνας priority, όπως ορίστηκε παραπάνω.

Β.Συγκριτικά με την προηγούμενη έκδοση της εφαρμογής.

Τα σημεία που έχουν αλλάξει είναι τα εξής:

- ★ Αρχικά στον αλγόριθμο Α* στέλνουμε πλέον ερώτημα στην prolog και μας επιστρέφει αυτή το αποτέλεσμα της ευριστικής συνάρτησης.
- ★ Πρίν γίνει η αναζήτηση με τον Α*, αποκλείουμε όλα τα ταξί που δεν πληρούν τις προϋποθέσεις του πελάτη (πχ σε σχέση με τον αριθμό ατόμων, τις αποσκευές, το μήκος διαδρομής)
- ★ Στη συνέχεια κρατάμε τα 5 καλύτερα αποτελέσματα απο αυτά που δίνει ο Α*-που πλέον επιλέγει με βάση το πόσο γρήγορα θα φτάσει στον πελάτη- και τα δείχνουμε στον πελάτη.
- ★ Τέλος γίνεται ταξινόμηση με βάση τη βαθμολογία των υπολοίπων πελατών και δίνεται η τελική κατάταξη.

Αναλυτικά οι παραδοχές που έχουμε πάρει για το πρώτο φιλτράρισμα των ταξί…

```
//θα μπορέσει να συνεννοηθεί ο πελάτης με τον οδηγό?
	public Boolean canSpeak(String language) {
	   return this.languages.containsKey(language);
	}
//Είναι ελεύθερο το συγκεκριμένο ταξί?
	public Boolean isAvailable() {
	   return this.available;
	}
//Χωράνε οι επιβάτες στο συγκεκριμένο αυτοκίνητο?
	public Boolean canFitPersons(int persons) {
	   return (persons >= this.minCapacity && persons <= this.maxCapacity);
	}
//Χωράνε οι αποσκευές στο συγκεκριμένο αυτοκίνητο?
	public Boolean canFitLuggage(int luggage) {
	   if(this.type.equals("minivan") && luggage <= 5)
	      return true;
	   if(this.type.equals("large") && luggage <= 3)
	      return true;
	   if(this.type.equals("compact") && luggage <= 2)
	      return true;
	   if(this.type.equals("subcompact") && luggage <= 1)
	      return true;
	   return false;
	}
//Μπορεί να κάνει μεγάλη απόσταση το ταξί?
	public Boolean longDistance() {
	   return this.long_distance;
	}
```

Πίνακας 2: Tarifbeat.java methods

Αποσκευές ανά είδος Ι.Χ.	
minivan	5
large	3
compact	2
subcompact	1

Πίνακας 3: Παραδοχή σχετικά με τις αποσκευές ανα είδος Ι.Χ.

Max Speed ανά είδος δρόμου (εάν δεν δίνεται)	
motorway	130
trunk	110
primary	90
secondary	70
tertiary	70
unclassified	50
residential	50
living_street	20
road	50
motorway_link	80
trunk_link	50
primary_link	50
secondary_link	50
tertiary_link	50

Πίνακας 4: Παραδοχή σχετικά με τις μέγιστες ταχύτητες ανά είδος δρόμου όπως ορίζονται απο τον ελληνικό κ.ο.κ..

C.Test Cases

Input CSV	Output KML	Output HTML Map
client.csv	client-original-output.kml	originalclientresult.html
client.csv	client-original-fullroutes-output.kml	originalclientresult-fullroutes.html
client-galatsi.csv	client-galatsi-output.kml	cliengalatsitresult.html
client-koukaki.csv	client-koukaki-output.kml	clientkoukakiresult.html
client-reverse.csv	client-reverse-output.kml	reverseclientresult.html

Όλες οι δοκιμές πραγματοποιήθηκαν με τα υπόλοιπα input files όπως δόθηκαν στο θέμα.

D. Screenshots απο την εκτέλεση

```
Preparing Map ......Complete!
Gathering Taxi Data ......Complete!
Gathering Client Information ......Complete!
Searching for taxis...
 ----------------1st Taxi Sorting  ----------------
1. Taxi 100 ----------------
Smart Function Result = 3.527559699590316
2. Taxi 140 ----------------
Smart Function Result = 3.8613945735771953
3. Taxi 170 ----------------
Smart Function Result = 4.652236082164348
4. Taxi 220 ----------------
Smart Function Result = 4.991008567684562
5. Taxi 210 ----------------
Smart Function Result = 5.023770108919045
 ----------------2nd Taxi Sorting  ----------------
1. Taxi 100 ----------------
Smart Function Result = 9.2
2. Taxi 140 ----------------
Smart Function Result = 7.3
3. Taxi 220 ----------------
Smart Function Result = 7.2
4. Taxi 170 ----------------
Smart Function Result = 7.1
Please choose your taxi.
100
Success!!!mbptoutselectra:final electrasjavac -cp jiprolog-4.1.4.1.jar Tarifbeat2.java
```

```
mbptoutselectra:final electraskepetari$ java -cp .:jiprolog-4.1.4.1.jar Tarifbeat2
Preparing Map ......Complete!
Gathering Taxi Data ......Complete!
Gathering Client Information ......Complete!
Searching for taxis...
 ----------------1st Taxi Sorting  ----------------
1. Taxi 230 ----------------
Smart Function Result = 2.103898782731732
2. Taxi 210 ----------------
Smart Function Result = 2.211499487603854
3. Taxi 220 ----------------
Smart Function Result = 2.640073892788514
4. Taxi 120 ----------------
Smart Function Result = 2.8280427674771405
5. Taxi 170 ----------------
Smart Function Result = 2.9146389425369614
 ----------------2nd Taxi Sorting  ----------------
1. Taxi 230 ----------------
Smart Function Result = 9.8
2. Taxi 210 ----------------
Smart Function Result = 9.2
3. Taxi 120 ----------------
Smart Function Result = 8.0
4. Taxi 220 ----------------
Smart Function Result = 7.2
5. Taxi 170 ----------------
Smart Function Result = 7.1
Please choose your taxi.
230
mbptoutselectra:final electraskepetari$
```