ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη αλγορίθμων δημιουργίας δρομολογίων απορριμματοφόρων σε πολυεπεξεργαστικά υπολογιστικά συστήματα.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΙΤΗ
ΔΡΟΣΟΣ ΔΗΜΗΤΡΙΟΣ

ΕΠΙΒΛΕΠΩΝ: Παναγιώτης Αλεφραγκής – Καθηγητής Εφαρμογών

Πίνακας περιεχομένων

Περίληψη

Η πτυχιακή έχει ως στόχο την σχεδίαση και ανάπτυξη εφαρμογής για την επίλυση και γραφική αποτύπωση των προτεινόμενων διαδρομών οχημάτων αποκομιδής σκουπιδιών σε μητροπολιτικό δήμο. Στο πλαίσιο της μελετήθηκαν οι δυνατότητες χρήσης βιβλιοθηκών ανοιχτού κώδικα (or-tools, open street maps, graphhoper) για την ανάπτυξη λογισμικού το οποίο μπορεί να επιλύσει και να αναπαραστήσει γραφικά τις διαδρομές των οχημάτων αποκομιδής σκουπιδιών πάνω σε χάρτη με την χρήση δεδομένων από το έργο Dynacargo.

Η βιβλιοθήκη or-tools της Google χρησιμοποιήθηκε για την εύρεση των διαδρομών των οχημάτων χρησιμοποιώντας δεδομένα από το έργο Dynacargo. Δεδομένα που έχουν συλλεχθεί από κάθε κάδο απορριμματοφόρων της Ναυπάκτου και των γύρω χωριών της με στόχο την πρόβλεψη του πότε κάθε κάδος θα φτάσει στο μέγιστο της χωρητικότητας του ανάλογα με την χρήση των κατοίκων της περιοχής.

Τα δεδομένα αυτά επεξεργάζονται με κατάλληλη μοντελοποίηση από την βιβλιοθήκη σε γλώσσα κώδικα java και προετοιμάζονται διαδρομές ανα ημέρα και αριθμό οχημάτων ανάλογα με την επιλογή του χρήστη με κάδους οι οποίοι σύμφωνα με τα στατιστικά του dynacargo χρειάζονται άδειασμα.

Στην συνέχεια ένας τοπικός graphhoper server χρησιμοποιώντας ένα κομμάτι του χάρτη με την περιοχή των διαδρομών αποκομμένος από το open street maps δημιουργεί μια γραφική αποτύπωση της εν λόγο διαδρομής σύμφωνα με τις επιλογές του χρήστη.

Κύριός στόχος της εφαρμογής είναι η βέλτιστη χρήση του στόλου οχημάτων του δήμου Ναυπάκτου εξυπηρετώντας μόνο τους κάδους που πλησιάζουν την μέγιστη χωρητικότητα τους με συνέπεια την μείωση άσκοπων δαπανών σε καύσιμα. Το πρόβλημα αυτό ανήκει στην
ομάδα προβλημάτων του πλανόδιου πωλητή (Traveling Salesman Problem), ενα από τα ποιο διάσημα προβλήματα στο πεδίο επιστήμης υπολογιστικών συστημάτων. Στην παρούσα εργασία αναπτύσσετε εφαρμογή για την μοντελοποίηση και λύση του στην περιοχή της Ναυπάκτου και των χωριών που έλαβαν μέρος στο Dynacargo.

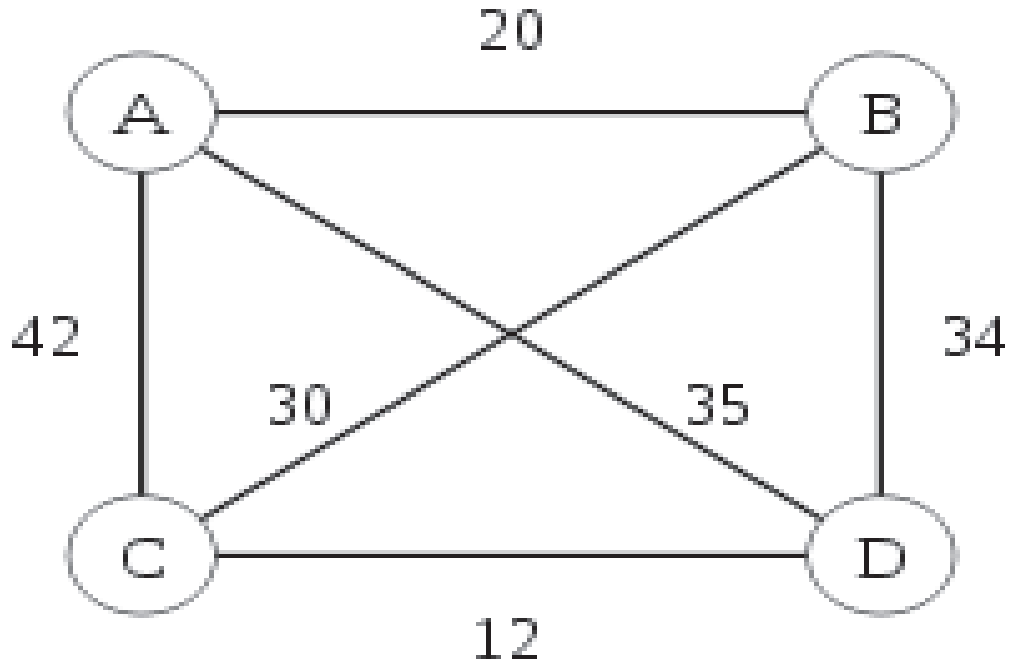
Κεφάλαιο 1.ΕΙΣΑΓΩΓΗ

Εισαγωγή στο TSP(traveling salesman problem).

1.1.Το πρόβλημα του πλανόδιου πωλητή

Το πρόβλημα του πλανόδιου πωλητή είναι πολλών ετών, και χαρακτηρίζετε από το πρόβλημα που αντιμετώπιζαν οι πλανόδιοι πωλητές που έπρεπε να ταξιδέψουν από πόλη σε πόλη για να πουλήσουν το εμπόρευμα τους. Ακολουθώντας μια γενικευμένη λογική, όσο καλύτερα προγραμματισμένη ήταν οι διαδρομή τους ως προς την μείωση της απόστασης που έπρεπε να διανύσουν, τόσο το καλύτερο.
Παρόλο που ένα πρόβλημα της τάξης των 10 πόλεων μπορεί να λυθεί σχετικά εύκολα από έναν άνθρωπο, το να βρεθεί η μικρότερη διαδρομή ανάμεσα σε μια ομάδα τοποθεσιών αποτελεί ένα εκθετικά δύσκολο πρόβλημα καθιστώντας ένα πρόβλημα βέλτιστης διαδρομής μεταξύ 20 πόλεων περισσότερο από δυο φορές δυσκολότερο των 10.

Μαθηματικά το πρόβλημα του πλανόδιου πωλητή μπορεί να χαρακτηριστεί σαν γραφική παράσταση. Ο στόχος είναι να βρεθεί



το μονοπάτι που θα επισκέπτεται όλους τους κόμβους με το μικρότερο άθροισμα βαρών μεταξύ τους.

1.2 Πρόβλημα δρομολόγησης οχημάτων

Το πρόβλημα δρομολόγησης οχημάτων πηγάζει από το πρόβλημα του πλανόδιου πωλητή στοχευμένο στην εύρεση πολλαπλών διαδρομών για ένα στόλο οχημάτων που θέλει να εξυπηρετήσει μια ομάδα κόμβων. Στην προκειμένη περίπτωση η εξυπηρέτηση συχνά αναφέρετε στην απλή επίσκεψη τους μιας και δεν περιλαμβάνετε στο πρόβλημα κάποια μεταβλητή αποθήκευσης ή παράδοσης κάποιου προϊόντος στους κόμβους του κάθε οχήματος.

Ο στόχος της εργασίας παραπέμπει σε ένα ακόμα παρακλάδι του προβλήματος του πλανόδιου πωλητή και του προβλήματος δρομολόγησης οχημάτων, το πρόβλημα δρομολόγησης οχημάτων με ιδιότητες περιορισμού.

Το πρόβλημα δρομολόγησης οχημάτων με ιδιότητες περιορισμού εμπλουτίζει το πρόβλημα δρομολόγησης οχημάτων, με περιορισμούς που πρέπει να τηρηθούν κατά την διεξαγωγή της λύσης του. Ενώ στο απλό πρόβλημα δρομολόγησης οχημάτων ο μόνος περιορισμός ήταν ότι όλοι οι κόμβοι πρέπει να επισκεφθούν από ένα όχημα, στο παρόν πρόβλημα προσθέτονται και άλλες περιοριστικές μεταβλητές ώστε να αποτυπώνουν αληθινά προβλήματα πιο ρεαλιστικά.

Τέτοιοι περιορισμοί μπορεί να είναι τα διαθέσιμα καύσιμα του κάθε οχήματος, και ο περιορισμός καθ' αυτός είναι να μην τελειώσουν κατά την διάρκεια εξυπηρέτησης κάποιου κόμβου από το όχημα.

Σε περίπτωση μεταφοράς κάποιου φορτίου που λαμβάνετε από κάθε επισκεπτόμενο κόμβο να υπάρχει αρκετός χώρος έτσι ώστε το όχημα να μπορέσει να συλλέξει φορτίο από όλους τους κόμβους που έχει προγραμματιστεί να εξυπηρετήσει χωρίς να υπερβεί το όριο του επιτρεπόμενου φορτίου του.

Η μοντελοποίηση του προβλήματος μπορεί να περιλαμβάνει περιορισμούς χρόνου. Δηλαδή ένα χρονικό όριο στο οποίο πρέπει η εξυπηρέτηση των κόμβων να έχει ολοκληρωθεί πριν αυτό τελειώσει. Τέτοιο είδος περιορισμός παραπέμπει σε προβλήματα δρομολόγησης οχημάτων που αφορούν τις δημόσιες και μη μαζικές συγκοινωνίες. Παραδείγματος χάρη τον στόλο λεωφορείων αστικής συγκοινωνίας του δήμου Ναυπάκτου και σε άλλες περιπτώσεις εταιριών ταχυμεταφορών.

Η λύση τέτοιων προβλημάτων αποφέρει επεκτάσιμα ανάλογα με τον όγκο τους πλεονεκτήματα που μπορούν να επιφέρουν μεγάλα κέρδη στους οργανισμούς που θα αφιερώσουν πόρους για την βελτιστοποίηση τους. Αρκετές φορές τα προβλήματα αυτά είναι τόσο περίπλοκα ώστε η πλήρη λύση τους δεν είναι εφικτή ή ακόμα και άκρος επιθυμητή σε περίπτωση που η επεξεργαστική ισχύς του υπολογιστικού συστήματός που την υπολογίζει δεν είναι επαρκείς ώστε να είναι επιτυχές μέσα στα επιθυμητά χρονικά περιθώρια του χρήστη ή ακόμα και η υλοποίηση του κώδικα του αλγορίθμου να μην αποφέρει την μέγιστη ταχύτητα λύσης με αποτέλεσμα μια απλά βελτιωμένη λύση του προβλήματος να είναι επαρκείς. Αν και μη βέλτιστη, τα αποτελέσματα σε σχέση με μια πρωταρχική μη βελτιστοποιημένη λύση θα αποφέρει μεγάλα πλεονεκτήματα. Βιβλιοθήκες αλγόριθμων όπως αυτή του

or-tools χρησιμοποιείτε σε ένα ιδικό είδος προγραμματισμού που ονομάζετε περιοριστικός προγραμματισμός(Constraint programming). Παρόλο που η έννοια του OOP(object oriented programming) συνεχίζει να ισχύει λόγο της φύσης των γλωσσών προγραμματισμού που χρησιμοποιούνται(στην περίπτωση της άσκησης η Java) ο περιοριστικός προγραμματισμός κάνει πιο εύκολη την αποτύπωση προβλημάτων όπως αυτό του πλανόδιου πωλητή και των ειδικεύσεων του διότι δίνει στην περίπτωση μας την δυνατότητα της έκφρασής των διάφορων μεταβλητών του προβλήματος σαν συλλογές από αντικείμενα.

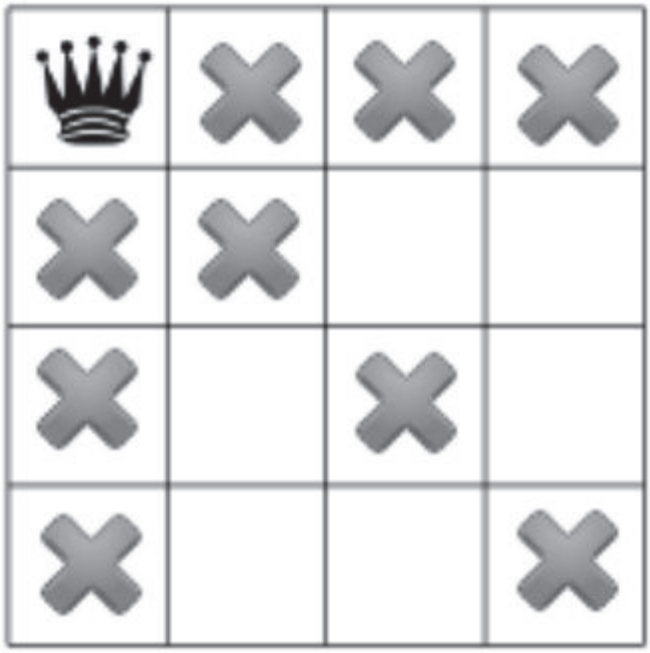
Η διαφορά του από τον OOP είναι η χρησιμοποίηση περιορισμών που δρουν διαφορετικά από τα απλά θεμελιακά στοιχεία με την έννοια ότι δεν περιγράφουν βήματα ή μια αλληλουχία τους, αλλά ιδιότητες για την λύση ενός προβλήματος. Τέτοιες ιδιότητες δημιουργούνται για την ικανοποίηση διάφορων περιορισμών που έχουν τεθεί σε αυτό.

1.3 Παράδειγμα τεσσάρων βασιλισσών

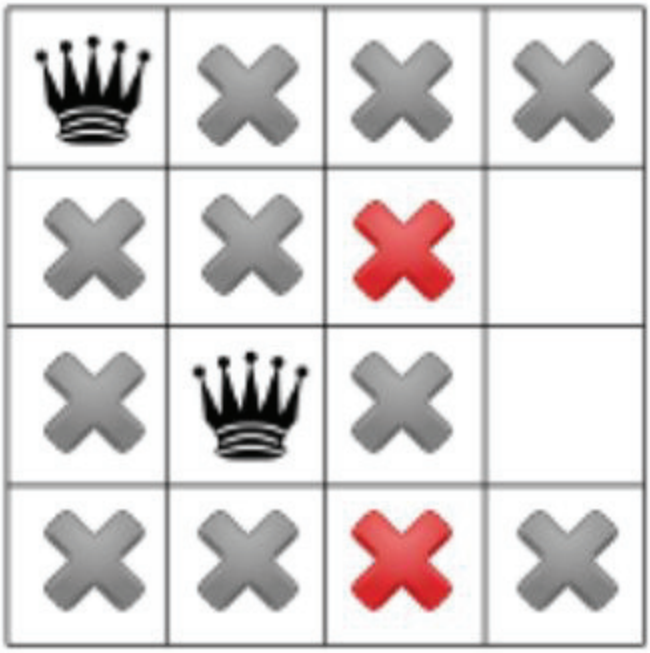
Σε αυτό το κεφάλαιο θα αναλυθεί το πως δουλεύει ο περιοριστικός προγραμματισμός με το παράδειγμα των τεσσάρων βασιλισσών.

Έστω οτι έχουμε μια σκακιέρα και θέλουμε να τοποθετήσουμε τέσσερις βασίλισσες σε αυτήν με τέτοιο τρόπο ώστε καμία να μην απειλεί τις άλλες. Με αυτό το παράδειγμα ουσιαστικά επιστρέφουμε πίσω στον περιοριστικό προγραμματισμό, άρα βάση των περιορισμών που θα τεθούν θα γίνει προσπάθεια να λυθεί το παραπάνω πρόβλημα. Σε αυτή την περίπτωση του προβλήματος οι περιορισμοί είναι ότι μόνο μια βασίλισσα πρέπει να βρίσκεται σε μια σειρά, κολόνα, και διαγώνια γραμμή.

Ο αλγόριθμος θα ξεκινήσει βάζοντας την πρώτη βασίλισσα στην πρώτη σειρά της πρώτης στήλης.



Σύμφωνα με τους περιορισμούς που έχουν τεθεί στον αλγόριθμο καμία άλλη βασίλισσα δεν μπορεί να τοποθετηθεί στην διαγώνιο, στην οριζόντια και στην κάθετη γραμμή. Άρα ο αλγόριθμος θα συνεχίσει βάζοντας την επόμενη βασίλισσα στην αμέσως επόμενη ελεύθερη συντεταγμένη στην σκακιέρα.

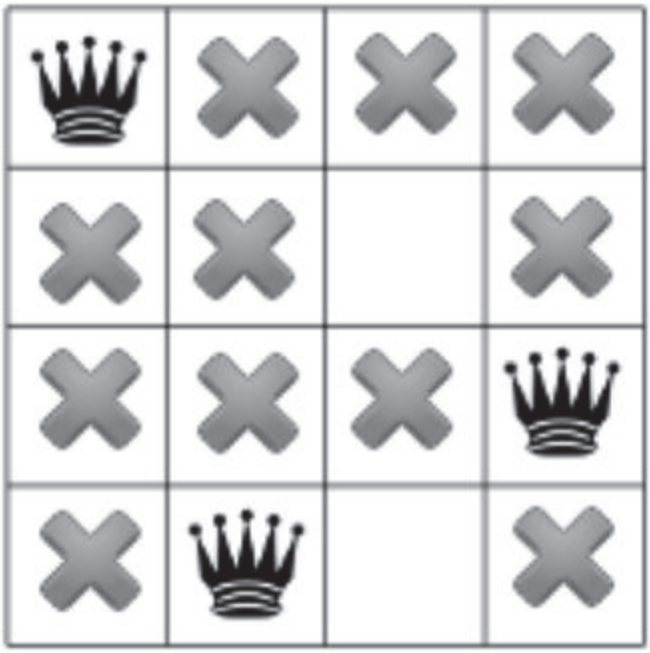


Τα δύο νέα σημεία που καλύπτονται από την καινούργια βασίλισσα διακρίνονται με κόκκινα Χ σύμφωνα με τους περιορισμούς που έχουν τεθεί. Εδώ φτάνουμε σε ένα σημείο όπου ο αλγόριθμος αποτυγχάνει να λύσει το πρόβλημα(*εδώ θα τυπωνόταν το failure άμα ήταν ενεργοποιημένη η επιλογή setLogSearch μέσα στον κώδικα που θα*

αναλυθεί στην συνέχεια) γιατί πλέον δεν μπορούν να τοποθετηθούν άλλες δυο βασίλισσες και να συνεχίζουν να ικανοποιούνται οι περιορισμοί.

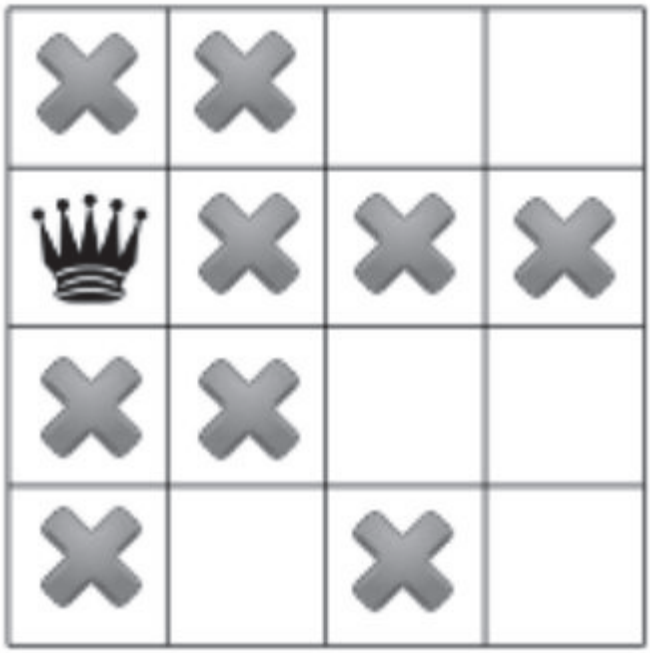
Σε αυτή τη περίπτωση ο αλγόριθμος κάνει backtracking δηλαδή σε περίπτωση που έχει κολλήσει επειδή έχει αναθέσει τιμές με τις οποίες δεν μπορεί να ικανοποιηθεί ο κύριος περιορισμός του προβλήματος και στόχος του, τότε επιστρέφει σε προηγούμενα βήματα του και απολύει το συγκεκριμένο μονοπάτι βημάτων που το οδήγησαν στο να κάνει backtracking. *Να σημειωθεί ότι αυτή η διαδικασία γίνεται και όταν ο αλγόριθμος έχει επιτύχει στο βρει αρχική λύση συνεχίζοντας με στόχο να βρει και άλλες καλύτερες.*

Λόγο του backtracking ο αλγόριθμος θα τοποθετήσει την δεύτερη βασίλισσα ξανά στο αμέσως επόμενο κάθετα τετράγωνο της σκακιέρας και αφού τοποθετήσει την τρίτη βασίλισσα τότε θα έρθει πάλι σε κατάσταση backtracking γιατί απέτυχε ακόμα μια φορά στο να βρει την λύση.

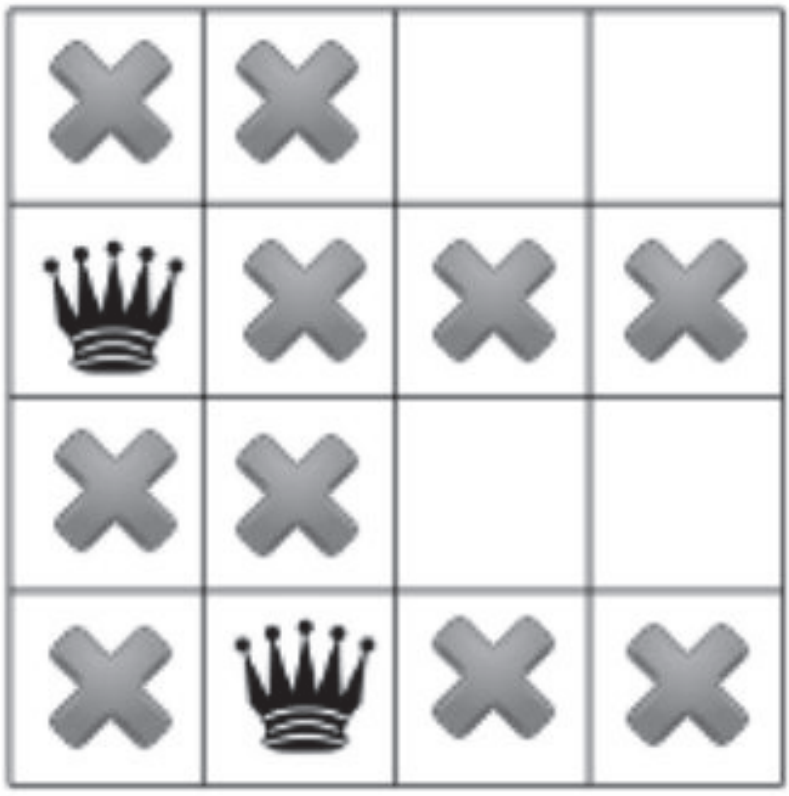


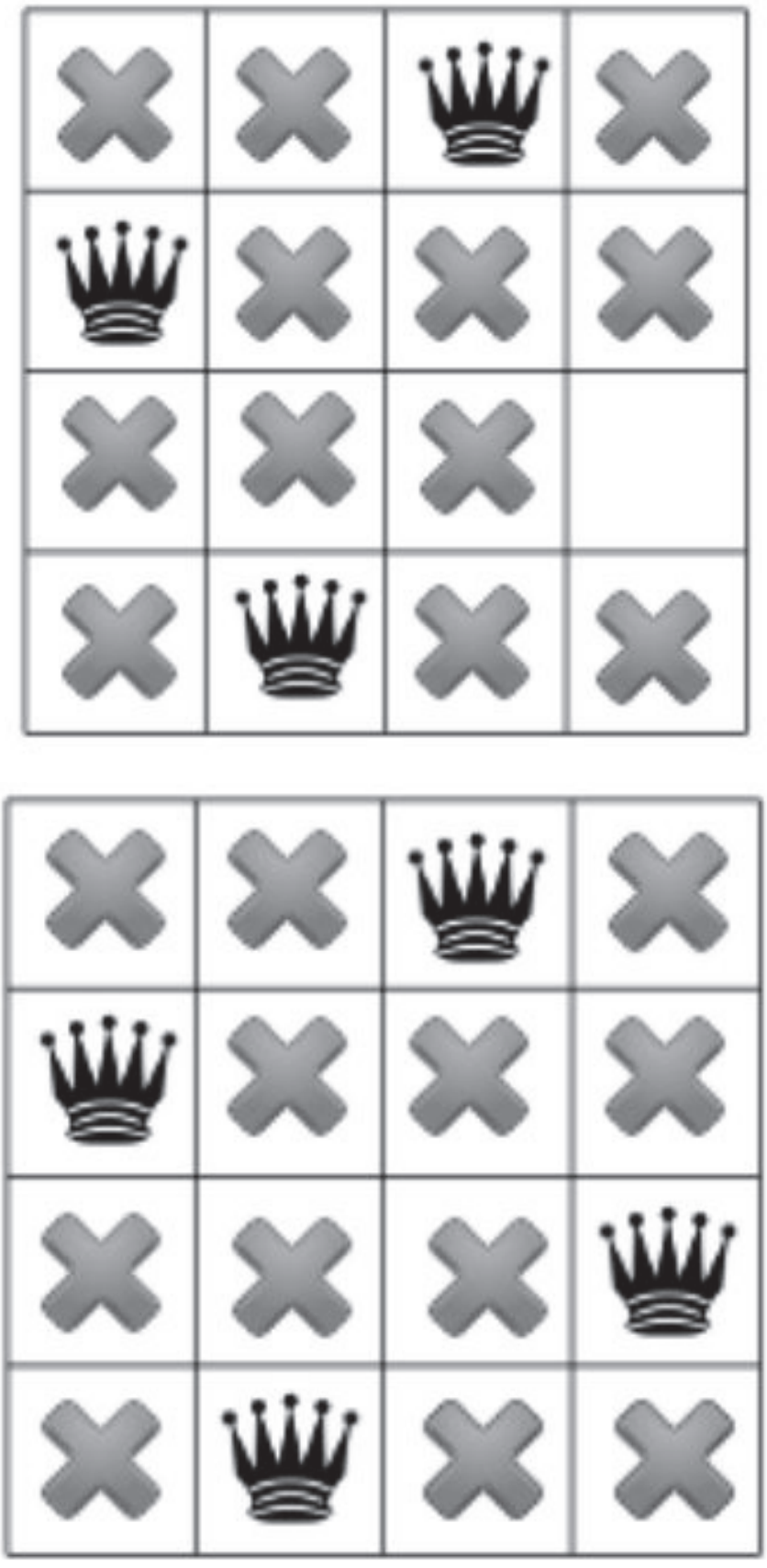
Όπως φαίνεται στην φωτογραφία δεν υπάρχει μέρος που να μπορεί να τοποθετηθεί η τέταρτη βασίλισσα και να ικανοποιούνται οι περιορισμοί.

Αφού δεν μπόρεσε πάλι ο αλγόριθμος να βρει κάποια λύση τότε αυτή τη φορά θα επιχειρήσει να αλλάξει θέση στην πρώτη βασίλισσα, πηγαίνοντας πίσω στο πρώτο βήμα της λύσης. Αυτή τη φορά το πρώτο βήμα της λύσης είναι το ακόλουθο.



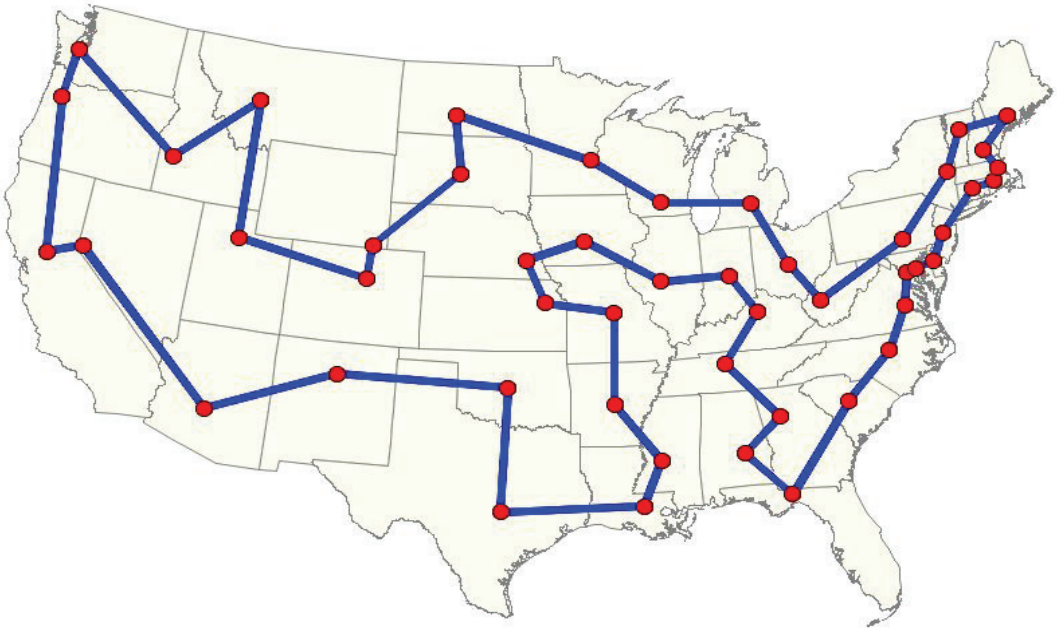
Εκτελώντας την ίδια ακολουθία ενεργειών βρίσκουμε τελική λύση.





Τελική λύση προβλήματος.

Αυτός είναι ο τρόπος με τον όποιο λειτουργεί ο αλγόριθμος or-tools και γενικότερα ο περιοριστικός προγραμματισμός. Με τον ίδιο τρόπο λειτουργεί ο αλγόριθμος και στο πρόβλημα της άσκησης.



Αντί για σημεία στην σκακιέρα έχουμε σημεία στον χάρτη που πρέπει να τα επισκεφτεί **ένα** όχημα και ο αποθηκευτικός χώρος των οχημάτων δεν πρέπει γεμίσει πριν τελειώσει η διαδρομή. Ο αλγόριθμος θα επιλέξει τις διαδρομές μεταξύ κόμβων που έχουν ως άθροισμα τα λιγότερα χιλιόμετρα για να βρει την μικρότερη διαδρομή.

Κεφάλαιο 2. Εργαλεία που χρησιμοποιήθηκαν

Open Street Maps

Το Open Street Maps αποτελεί ένα μεγάλο παράδειγμα της φιλοσοφίας του ανοιχτού κώδικα και μία ισχυρή εναλλακτική ανάμεσα στις υλοποιήσεις των μεγάλων εταιριών όπως Google και Apple. Ο λόγος που επιλέχτηκε για την αποτύπωση των αποτελεσμάτων της υλοποίησης του προβλήματος της άσκησης είναι η προσβασιμότητα του στον σπουδαστή, και σε πολλούς developers όπου χρησιμοποιώντας το API του έχουν υλοποιήσει προγράμματα αποτύπωσης δρομολογίων σε χάρτες που καθιστούν ευκολότερη την ενσωμάτωση τους λόγο της άδειας ανοιχτού κώδικα.

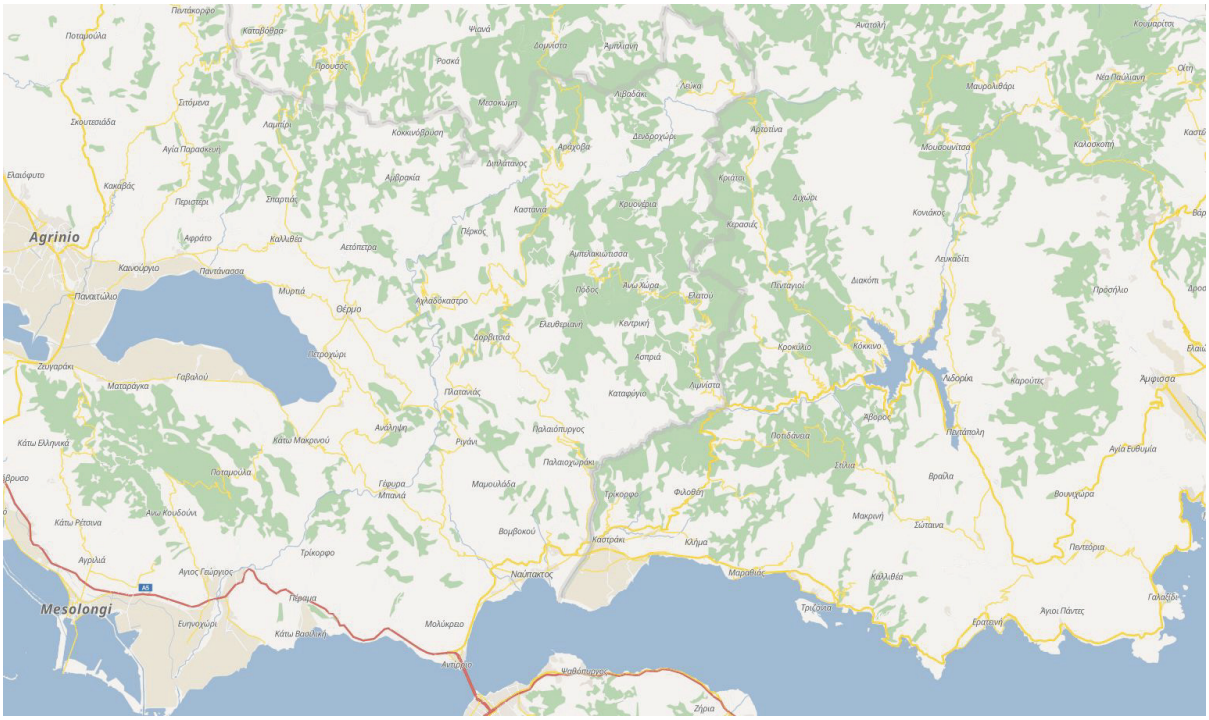
Graphhopper

Το Graphhoper παρέχει και αυτό την δυνατότητα δρομολόγησης οχημάτων μέσα από το site τους. Στην περίπτωση της εργασίας χρησιμοποιείτε μόνο μέρος του κώδικα τους για την απεικόνιση των δρομολογίων. Το Graphhoper API ειναι ένα εργαλείο αποτύπωσης και εύρεσης διαδρομών με παρόμοιες λειτουργίες με αυτές του προγράμματος της άσκησης που χρησιμοποιεί το API του Open Street Maps και επιλέχτηκε να χρησιμοποιηθεί για την αποτύπωση των αποτελεσμάτων της βιβλιοθήκης or tools.

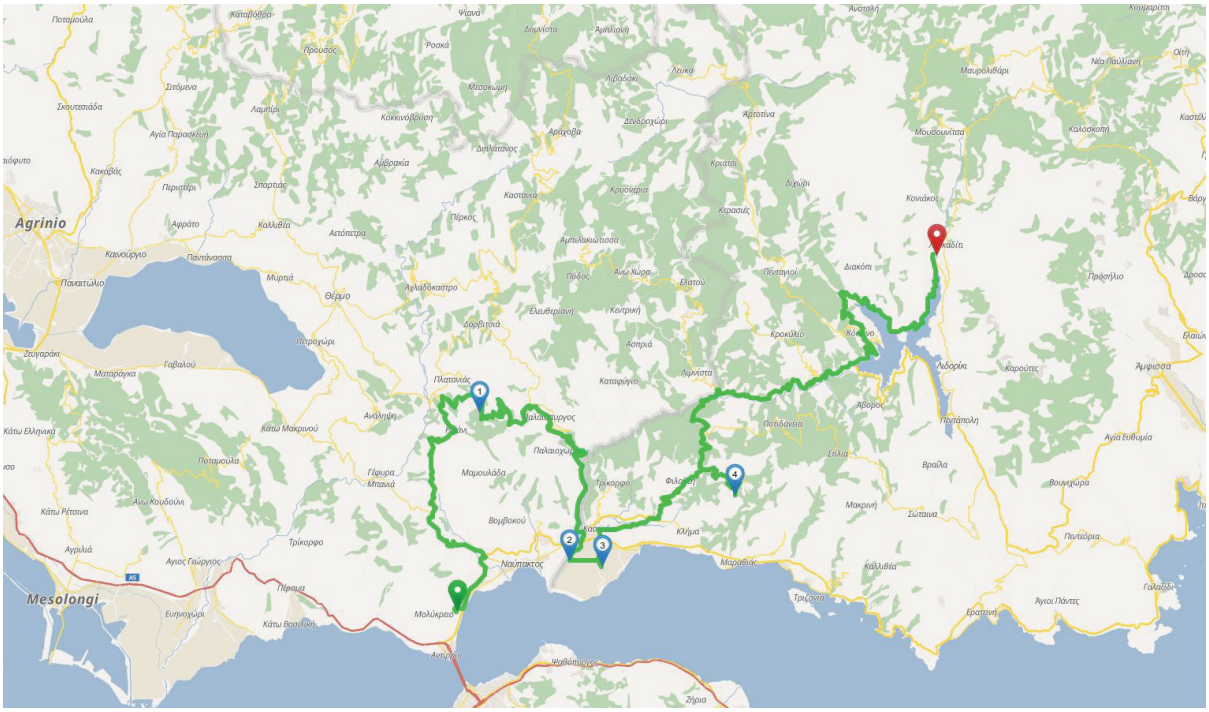
Το Graphhopper Map είναι μια ιδανική λύση για την αποτύπωση των διαδρομών στον χάρτη. Προσφέρει τοπικό server όπου χρησιμοποιώντας τα αρχεία .txt που εξήγαγε το or-tools δίνει την δυνατότητα αποτύπωσης των αποτελεσμάτων.

Αρχικά χρησιμοποιήθηκε το OpenStreetMap για να γίνει περικοπή του χάρτη της Ναυπάκτου. Το αρχείο αυτό χρησιμοποιείται από τον τοπικό Graphhopper server για να υπολογίσει ακριβές διαδρομές από τα σημεία που αναφέρουν τα αρχεία αποτελεσμάτων. Το αρχείο είναι μορφής .osm το οποίο διαμορφώθηκε σε αρχείο .pbf για να μειωθεί ο όγκος του αρχείου. Χαρακτηριστικά το osm αρχείο της Ναυπάκτου είναι 101MB ενώ το ανάλογο σε pbf μορφή είναι 4MB.

Η παραγωγή γραφικών αποτελεσμάτων γίνεται με την χρήση του graphhoper client όπου τρέχει τοπικά στον browser. Το μόνο που χρειάζεται για να ξεκινήσει ενας τοπικός server είναι να τρέξει το συμπεριλαμβανόμενο script και να ανοιχτεί ένα παράθυρο στο localhost:8989 το οποίο γίνεται αυτόματα από την εφαρμογή της διπλωματικής. Μέσα από τον .config αρχείο του μπορούν να γίνουν ρυθμίσεις για των αριθμό των νημάτων που θα έχει στην διάθεση του ο server, την stack και heap μνήμη που μπορεί να χρησιμοποιήσει.



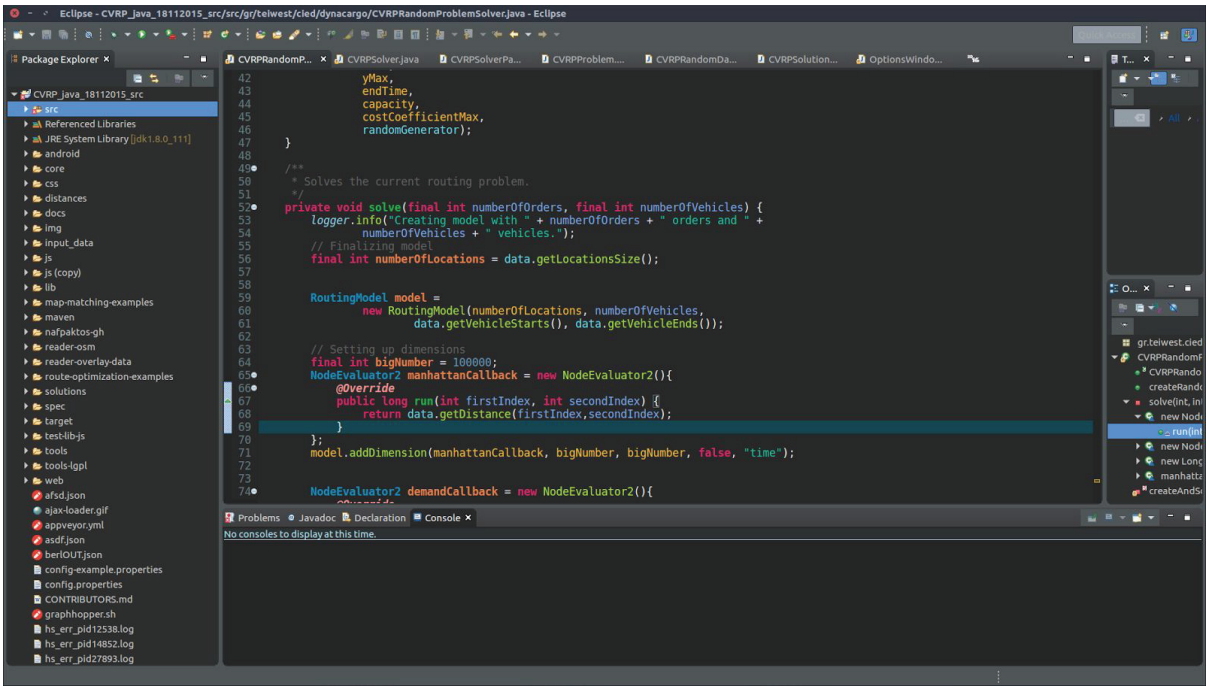
Χάρτης της περιοχής ενδιαφέροντος μέσα από τον τοπικό server.



Απλή διαδραστικότητα που προσφέρει ο χάρτης μέσω του server.

Eclipse IDE

Το eclipse ειναι το πρόγραμμα περιβάλλοντος προγραμματισμού που χρησιμοποιήθηκε. Υποστηρίζει πολλές γλώσσες προγραμματισμού και με αυτό γράφτηκε όλως ο κώδικα της εφαρμογής.



Or-Tools

Το or-tools αποτελεί μια σουίτα συνδυαστικής βελτιστοποίησης της Google. Μπορεί να λύσει προβλήματα περιοριστικού προγραμματισμού και στην περίπτωση μας λειτουργεί σαν αλγόριθμος γράφων, δηλαδή είναι το κομμάτι του κώδικα που λύνει το πρόβλημα που έχουμε αποτυπώσει με την χρήση περιοριστικού προγραμματισμού όπως στο παράδειγμα των τεσσάρων βασιλισσών.

Ένα από τα πιο δύσκολα και επικίνδυνα για λάθη μέρη του προβλήματος είναι η αποτύπωση του σε μορφή κώδικα. Δηλαδή με απλά λόγια, να εξηγήσουμε στον υπολογιστή το πρόβλημα έτσι ώστε αυτός να το λύσει. Παρόλο που αυτό ισχύει γενικότερα σαν έννοια στον προγραμματισμό. Στην περίπτωση του περιοριστικού προγραμματισμού φέρει ιδιαίτερη βαρύτητα, μιας και εν μέρη λανθασμένη αποτύπωση μπορεί να αλλάξει τελείως το τελικό αποτέλεσμα.

Το ίδιο ισχύει και για την βιβλιοθήκη or-tools. Καλώντας συγκεκριμένα κομμάτια κώδικα της παρουσιάζουμε σταδιακά το πρόβλημα στον αλγόριθμο όπου

στην συνέχεια θα το λύσει για εμάς. Λόγω της φύσης του προβλήματος όπως προαναφέρθηκε, ο στόχος δεν είναι η βέλτιστη διαδρομή, αλλά μια βελτιωμένη, που παρόλα αυτά θα έχει πολύ μεγάλη διαφορά από οποιαδήποτε αρχική που δεν έχει υποστεί επεξεργασία τέτοιας μορφής. Μπορεί να υπάρξει επιχείρημα ότι ίσως οι οδηγοί των οχημάτων αυτών λόγο εμπειρίας γνωρίζουν ποιοι κάδοι φέρουν την μεγαλύτερη κίνηση από τους κάτοικους των περιοχών αλλά σε καμία περίπτωση δεν μπορούν να ξεπεράσουν με εμπειρικά μοντέλα την αποτελεσματικότητα των υπολογισμών βιβλιοθηκών όπως η or-tools.

Αρχικά ο κώδικας δέχεται τα δεδομένα του Dynacargo μέσω των διαθέσιμων μελετών σε μορφή αρχείων .txt. Το format των αρχείων αυτών είναι json, μια πολύ καλή διαμόρφωση για αναπαράσταση μεγάλου όγκου δεδομένων. Στον κώδικα θα χρησιμοποιηθεί μια ειδική βιβλιοθήκη για ανάγνωση τέτοιας διαμόρφωσης, η JSON.simple. Αυτή εχει την δυνατότητα να διαβάσει το “δέντρο’’ της json στο αρχείο και να δημιουργήσει συλλογές αντικειμένων που θα χρησιμοποιηθούν από το or-tools για τον υπολογισμό διαδρόμων. Συγκεκριμένα θα δημιουργηθούν λίστες από αντικείμενα οχημάτων και κάδων με τα στοιχεία τους όπως προαναφέρθηκαν.

Μαζί με τα οχήματα και τους κάδους δημιουργούνται και άλλα είδη αντικειμένων που χρησιμοποιούνται κυρίως σαν δοχεία για αυτά τα δυο, αλλά και εισάγοντας καινούργια στοιχεία και μεθόδους με στόχο την ολοκληρωμένη διατύπωση του προβλήματος σε μορφή κώδικα. Η βιβλιοθήκη or-tools περιέχει πολύ μεγάλη γκάμα ρυθμίσεων για το πως μπορεί να επιλέξει ο χρήστης της το πως θα “δουλέψει’’ ο αλγόριθμος.

Μερικές που αξίζει να σημειωθούν είναι:

setLogSearch - άμα αυτή η επιλογή είναι ενεργοποιημένη ο κώδικας του or tools θα τυπώνει πολλά χρήσιμα στατιστικά για την κάθε λύση που βρίσκει, όπως το πόση ώρα πέρασε από την στιγμή που ξεκίνησε να ψάχνει για λύσεις, πόσες φορές απέτυχε, πόσο βαθιά έψαξε να βρει μια λύση κ.α

setTimeLimitMs - πόση ώρα επιθυμεί ο χρήστης να τρέξει ο αλγόριθμος

setSolutionLimit - όριο αριθμού λύσεων του αλγόριθμου, αμα παραδειγματος χάρη τεθεί το setSolutionLimit(3), τότε ο αλγόριθμος θα βρει τρεις λύσεις(άμα υπάρχουν) και μετά θα σταματήσει

setOptimizationStep - σε περίπτωση που καλεστεί αυτή η επιλογή με έναν αριθμό N, τότε ο αλγόριθμος καθώς έχει βρει μια αρχική λύση και σαν επόμενο βήμα ψάχνει για επόμενη καλύτερη, θα θεωρήσει καλύτερη λύση μόνο κάποια η οποία έχει διαφορά N από την προηγούμενη.

Από τις προηγούμενες επιλογές, όπως προαναφέρθηκε λόγο της φύσης του προβλήματος δεν ψάχνουμε για βέλτιστη λύση και χρησιμοποιούμε μόνο την επιλογή setTimeLimitMs.

Αφού η βιβλιοθήκη or-tools εκτελέσει τους υπολογισμούς της και δημιουργηθούν διαδρομές για τον αριθμό των οχημάτων και τις μέρες που έχουν επιλεχθεί από τον χρήστη το πρόγραμμα θα εξάγει αρχεία .txt σε json format για κάθε όχημα ανα ήμερα. Τα αρχεία περιέχουν στατιστικά για το όνομα του οχήματος, τα χιλιόμετρα τα όποια διένυσε στην διαδρομή, πόσα κιλά απορριμάτων σύλλεξε από τους κάδους που εξυπηρέτησε και τα λίτρα καυσίμου που χρειάστηκε.


```
{
"vehicleid":21,
"totalKMs":42.51,
"vehile_capacity":1500,
"total_waste":1065.0,
"total_fuel_consumption":5.1011999999999995,
"bins":[
    {
        "sid":164,
        "lon":21.83415589,
        "lat":38.40025867,
        "estLoad":0,
        "threshold":2147483647
    },
    {
        "sid":326,
        "lon":21.83284633,
        "lat":38.39748803,
        "estLoad":30,
        "threshold":60
    },
```


Κάτω από τα στατιστικά του οχήματος ακολουθεί μια λίστα με τους κάδους που επισκέφθηκε. Παρατηρείται ότι ο πρώτος κάδος είναι το node 164, δηλαδή ο τόπος στάθμευσης του στόλου οχημάτων. Το estLoad του είναι μηδέν γιατί στην πραγματικότητα δεν είναι κάδος και το threshold του είναι ένα πολύ μεγάλο νούμερο σε σχέση με όλα τα άλλα nodes. Αυτό γίνεται εσκεμμένα, το threshold είναι το “penalty’’ που έχει κάθε κόμβος, δηλαδή οι επιπτώσεις του οχήματος που θα το εξυπηρετήσει και έχει οριστεί έτσι ώστε ο αλγόριθμος να μην το λάβει υπόψιν του σαν επιλογή μέχρι να τελειώσουν όλοι οι κάδοι κόμβοι και να μην έχει άλλον κόμβο από τον αρχικό ώστε να γυρίσει πίσω στον χώρο στάθμευσης.

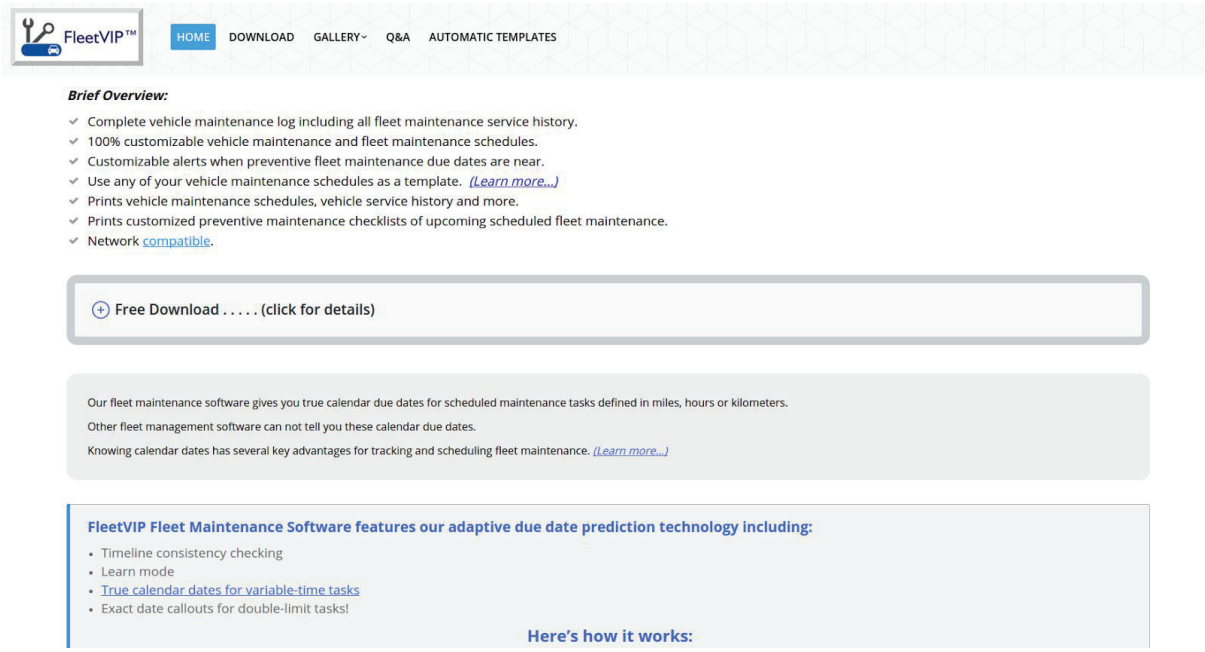
Project Dynacargo

Το project Dynacargo(Dynamic Routing on-the-go) που ξεκίνησε τον Σεπτέμβριο του 2013 και έληξε τον Ιούνιο του 2015, κατάφερε να μαζέψει στατιστικά με τα οποία μπορούμε να καθορίσουμε σταθερές για τον κάθε κάδο στην και γύρω από την περιοχή της Ναυπάκτου. Το project είχε στόχο να δημιουργήσει ένα τεχνολογικά εξελιγμένο σύστημα συλλογής πληροφοριών με την δυνατότητα δυναμικής βελτιστοποίησης δρομολογίων των απορριμματοφόρων της πόλης χρησιμοποιώντας αισθητήρες διαμοιρασμένους σε σημεία ενδιαφέροντος της πόλης. Η στατιστική για τους κάδους απορριμάτων συλλέχτηκε μέσω οχημάτων εξοπλισμένα με δέκτες οι όποιοι καθώς το όχημα περνούσε κοντά τους έπαιρνε πληροφορίες όπως το ποσοστό πληρότητας τους. Το πρόγραμμα της άσκησης χρησιμοποιεί αυτά τα δεδομένα σε μορφή αρχείων .txt όπου μέσα τους αναγράφονται πληροφορίες για τον κάθε κάδο ξεχωριστά. Οι πληροφορίες είναι ο ID αριθμός του κάθε κάδου, οι συντεταγμένες του στον χάρτη(γεωγραφικό πλάτος/μήκος), η πρόβλεψη πλήρωσης χωρητικότητας του κάδου σε ποσοστό της εκατό, η πρόβλεψη βάρους του κάδου, τα άνω και κάτω όρια του κάδου, και άλλα τα οποία θα καλυφθούν κατά την ανάπτυξη του μοντέλου.

2.1 Παρόμοια Εργαλεία Fleet Management

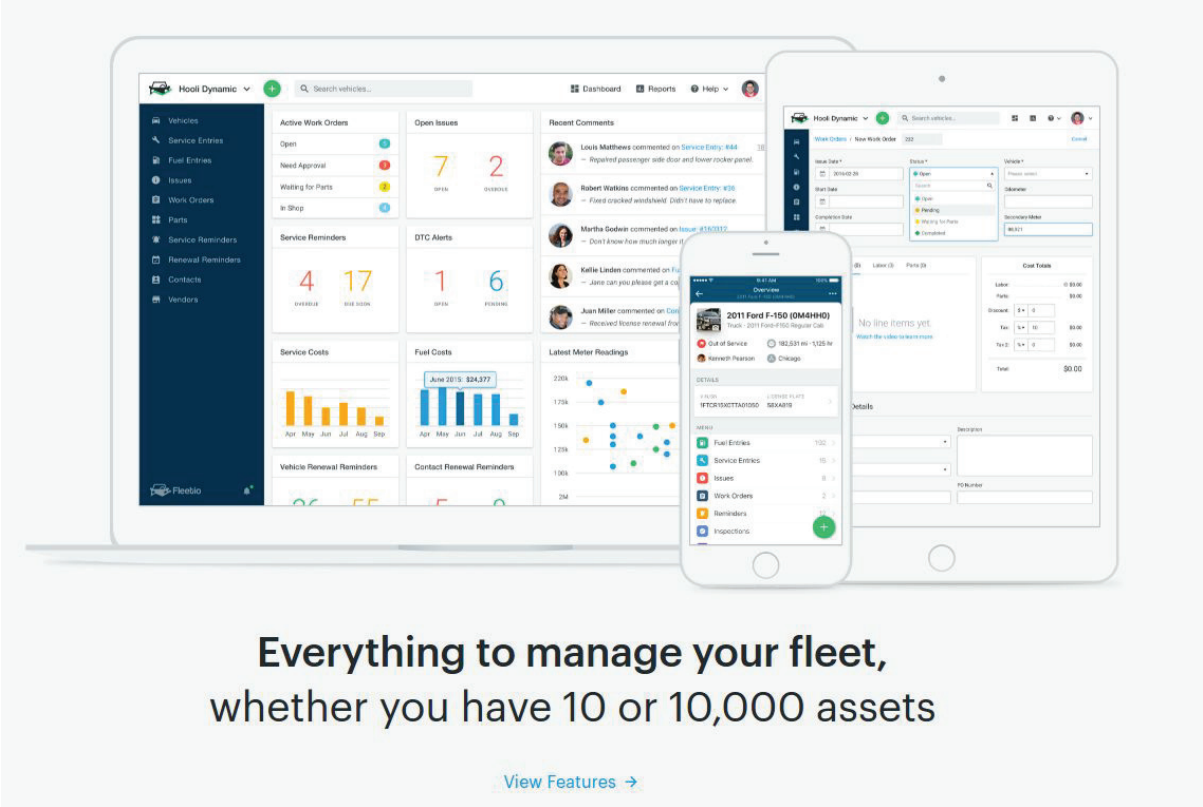
Υπάρχουν αρκετά projects τα οποία προσφέρουν παρόμοια λειτουργικότητα με αυτή της εφαρμογής της πτυχιακής αν και πολλά από αυτά δεν προσφέρουν την δημιουργία διαδρομών έχοντας σαν κύρια λειτουργία την καταγραφή των διαδρομών οχημάτων.

Fleet VIP



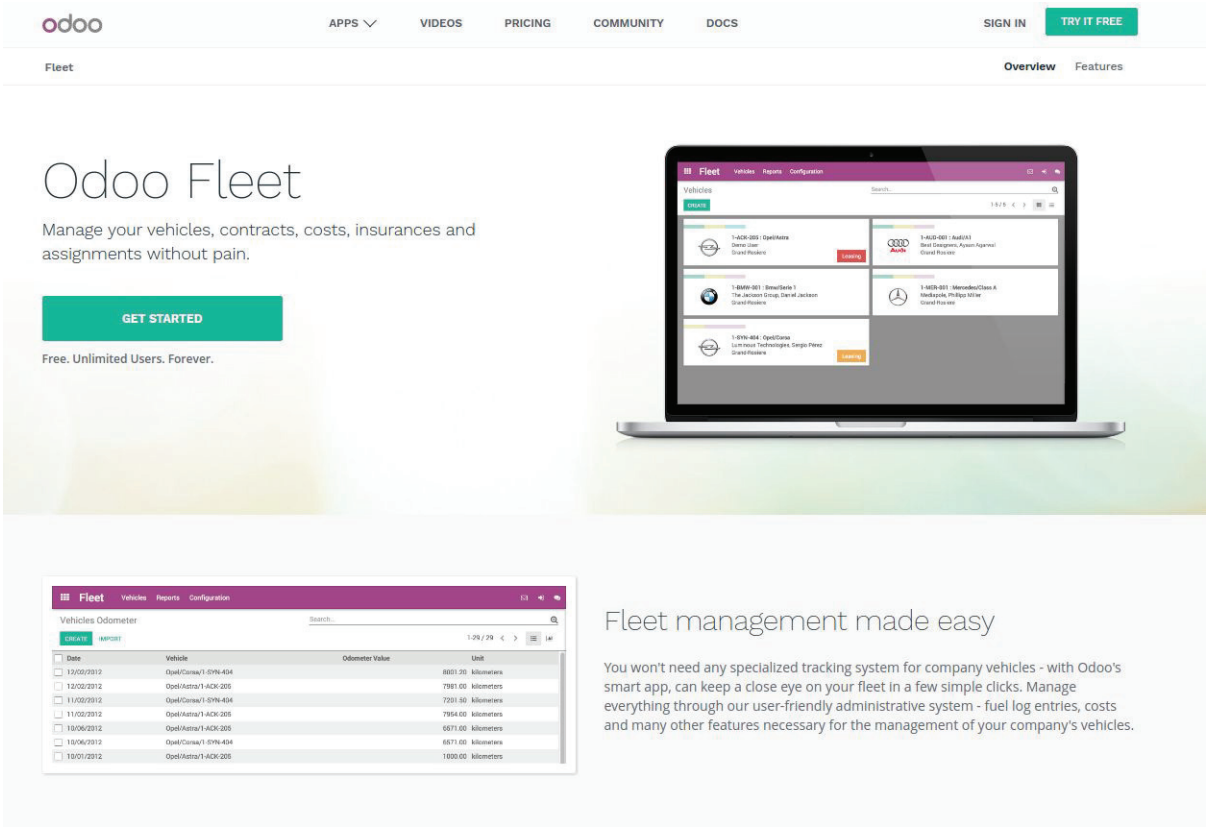
Το Fleet Vip προσφέρει στην free εκδοσή του υπηρεσίες για δυο οχήματα. Προσφέρουν GPS Tracking, καταγραφή διάνισης χιλιομέτρων για τα οχήματα και υπενθύμιση για προληπτική συντήρηση των οχημάτων.

Fleetio



Προσφέρουν ένα φιλικό προς τον χρήστη περιβάλλον και η free έκδοση περιλαμβάνει μέχρι 10 οχήματα. Προσφέρουν τις βασικές υπηρεσίες όπως GPS Tracking, καταγραφή διάνισης χιλιομέτρων και καυσίμου και σαν extra την δυνατότητα εξαγωγής των δεδομένων σε spreadsheet.

Odoo Fleet Management

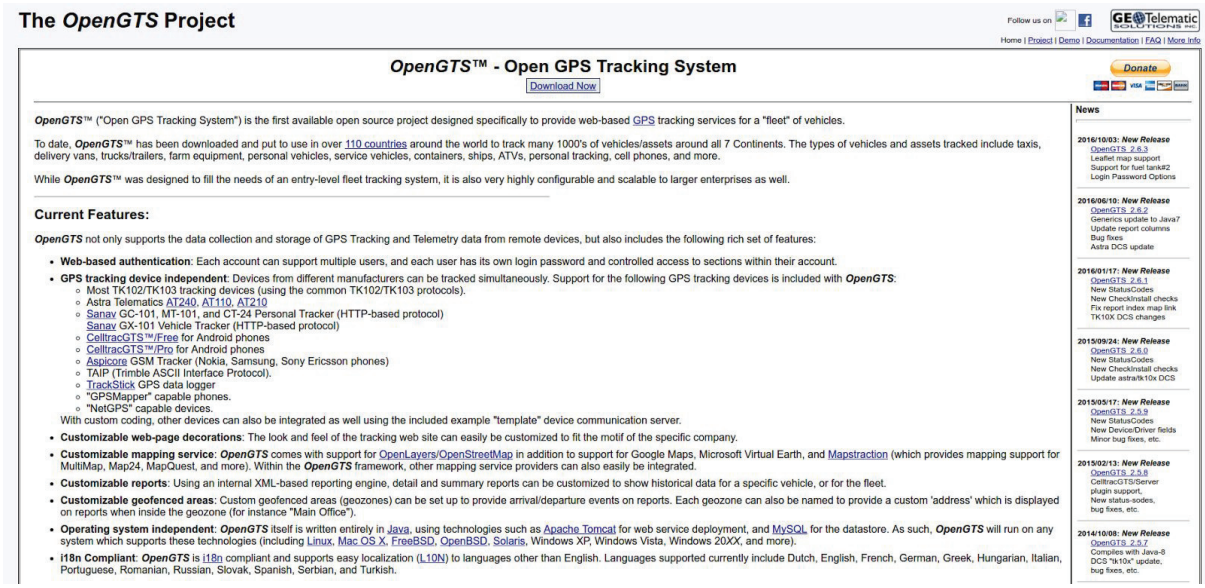


Και αυτό σαν τα προηγούμενα παρέχει κυρίως λογιστικές υπηρεσίες αλλά το φάσμα του επεκτείνεται και σε άλλους τομείς εκτός το fleet management. Πέρα από τις βασικές υπηρεσίες καταγραφής καυσίμων και διάνισης χιλιομέτρων προσφέρουν υπενθυμίσεις για λήξη συμβολαίων.

GPS Wox

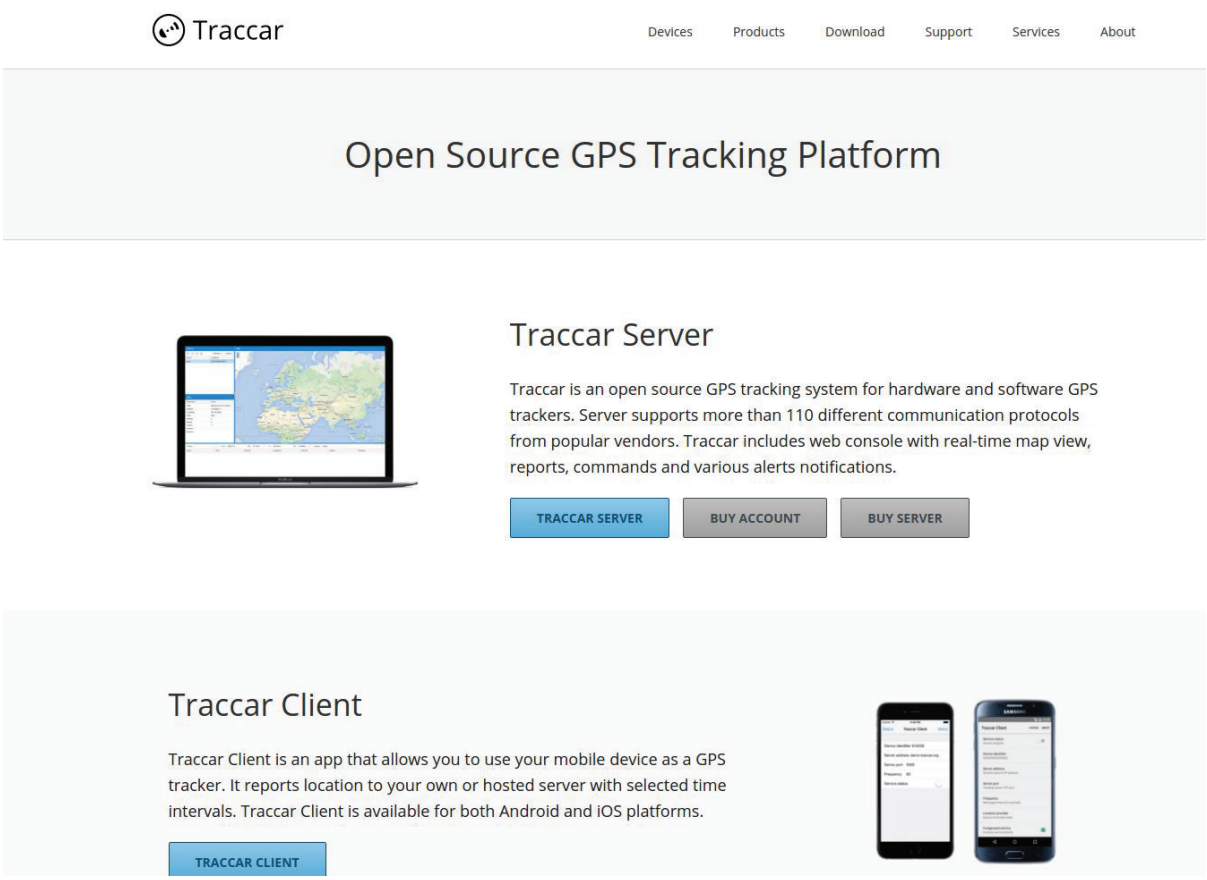
Το GPS Wox είναι και αυτό ένα fleet management λογισμίκο που φέρει χρησιμότητα κυρίως σε μικρές επιχειρήσεις. Διαθέτει δωρεάν χρήση μονο για ενα όχημα και εχει την δυνατότητα real time tracking.

Open GTS Project



Το open gts project διαφέρει απο τα προηγούμενα προγράμματα, διότι σου δίνει την δυνατότητα μέσω της άδειας ανοιχτού κώδικα την εμπλουτίση του ηδη βασικού προγράμματος. Η βασική λειτουργία του είναι ανίχνευση τοποθεσίας οχημάτων.

Traccar



Το Traccar αποτελεί και αυτό μια δωρεάν εφαρμογή GPS Tracking. Σαν το Open GTS, παρέχει μόνο την βασική λειτουργία πρόσβασης της τοποθεσίας των οχημάτων. Παρέχει επίσης server-based έκδοση αλλά και για λειτουργικά συστήματα κινητών τηλεφώνων.

Γενικά η εφαρμογή που δημιουργήθηκε για την διπλωματική άσκηση δεν ανήκει ακριβώς στην κατηγορία fleet management διότι προσφέρει την ικανότητα βελτιστοποίησης διαδρόμων των οχημάτων. Υπηρεσία σαν αυτή έχει αποσπαστεί σε

δικό της τομέα έξω από τις λογιστικές υπηρεσίες που προσφέρουν τα FM προγράμματα και υπηρεσίες.

Κεφάλαιο 3. Μοντελοποιηση Προβληματος

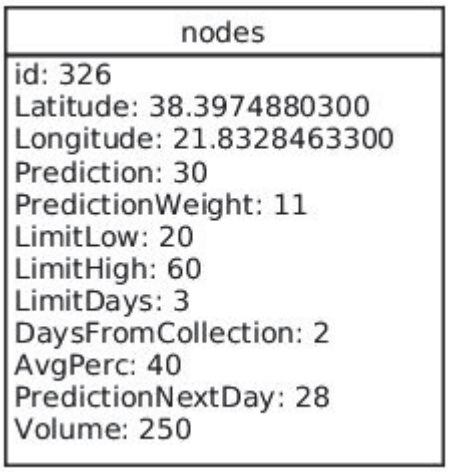
Σε αυτό το κεφάλαιο θα γίνει διατύπωση και ανάλυση του προβλήματος της άσκησης.

Το πρόβλημα που μας απασχολεί στην πτυχιακή άσκηση είναι η μοντελοποίηση και λύση του *Capacitated Vehicle Routing Problem* για τον στόλο απορριμματοφόρων του δήμου Ναυπάκτου και όσες περιοχές εξυπηρετούνται από αυτόν.

Αρχικά πρέπει να γίνει ανάλυση των κάδων για την σωστή αποτύπωση τον χαρακτηριστικών του ως μέρος του κώδικα. Διαβάζοντας ένα από τα διαθέσιμα αρχεία απο το project Dynacargo έχουμε τα εξής δεδομένα για κάθε κάδο.


```
{
    "groups": [{
                "GroupId": 1,
                "nodes": [{
                    "id": 326,
                    "Latitude": 38.3974880300,
                    "Longitude": 21.8328463300,
                    "Prediction": 30,
                    "PredictionWeight": 11,
                    "LimitLow": 20,
                    "LimitHigh": 60,
                    "LimitDays": 3,
                    "DaysFromCollection": 2,
                    "AvgPerc": 40,
                    "PredictionNextDay": 28,
                    "Volume": 250
                }, {
                    "id": 327,
                    "Latitude": 38.3956191500,
                    "Longitude": 21.8280950300,
                    "Prediction": 27,
                    "PredictionWeight": 10,
```

Από την ανάγνωση του αρχείου μπορούμε να καταλάβουμε ότι υπάρχουν ομάδες κάδων με τα ανάλογα χαρακτηριστικά.
Οπότε το αντικείμενο που θα δημιουργηθεί για την αποτύπωση του κάδου στο πρόβλημα σύμφωνα με το αρχείο είναι το ακόλουθο.



ID: Το αναγνωριστικό, μοναδικό όνομα του κάδου.

Latitude/Longitude: Αποτελούν συντεταγμένες στον χάρτη για την τοποθεσία του.

Prediction: Πρόβλεψη πλήρωσης χωρητικότητας του κάδου.

PredictionWeight: Πρόβλεψη βάρους του κάδου.

LimitLow: Κάτω όριο κάδου.

LimitHigh: Άνω όριο κάδου.

LimitDays: Όριο ημερών κάδου.

DaysFromCollection: Μέρες μετά την τελευταία εξυπηρέτηση.

AvgPerc: Μέσο ποσοστό της εκατό.

PredictionNextDay: Πρόβλεψη για την επόμενη ημέρα.

Volume: Χωρητικότητα κάδου.

Κάποια από αυτά τα στοιχειά θα χρησιμοποιηθούν άμεσα σαν περιορισμοί του αλγόριθμου για εύρεση βελτιστοποιημένων διαδρομών.

Στοιχεία όπως το LimitLow, LimitHight αποτελούν τα κάτω και άνω όρια για το τι προτεραιότητα θα δώσει ο αλγόριθμος στον κάθε κάδο. Παραδείγματος χάρη όταν ο κάδος βρίσκετε κοντά στο LimitLow η προτεραιότητα του θα είναι χαμηλή. Αυτό δεν σημαίνει ότι δεν υπάρχει περίπτωση να τον διαλέξει ο αλγόριθμος προς εξυπηρέτηση, αλλά το ότι κάποιος άλλος κάδος που έχει ποσοστό πληρότητας που είναι πιο κοντά στο LimitHight έχει πολύ περισσότερες πιθανότητες να επιλεχτεί.

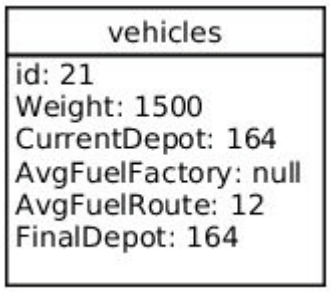
Άλλο περιοριστικό στοιχείο είναι το LimitDays, το όποιο είναι ένα χρονικό όριο για την συχνότητα εξυπηρέτησης του κάδου. Άμα το LimitDays για έναν κάδο είναι 3, τότε αυτός ο κάδος έχει και σαν περιορισμό ότι πρέπει να εξυπηρετείται έστω μία φορά κάθε τρεις μέρες.
Μια μεταβλητή που συνδέεται άμεσα με το LimitDays είναι το DaysFromCollection όπου κάθε μέρα που ο κάδος δεν εξυπηρετείτε ο αριθμός του μεγαλώνει κατά ένα, και σύμφωνα με το προηγούμενο περιορισμό δεν πρέπει να ξεπεράσει το 3 στην προκείμενη περίπτωση.

Ως τώρα ο στόχος του αλγόριθμου βελτιστοποίησης δρομολογίων είναι να βρει τις κοντινότερες διαδρομές μεταξύ κάδων χωρίς να μείνει ανικανοποίητος κάποιος περιορισμός όπως αυτοί των κάδων απορριμάτων που αναλύθηκαν ως τώρα.

Διαβάζοντας το περιεχόμενο του αρχείο βρίσκουμε πληροφορίες και για τα οχήματα, που σε αυτή την περίπτωση είναι τα απορριμματοφόρα της Ναυπάκτου.


```
"vehicles": [{
    "id": 21,
    "Weight": 1500,
    "CurrentDepot": 164,
    "AvgFuelFactory": null,
    "AvgFuelRoute": 12,
    "FinalDepot": 164
}]
```


Παρατηρούμε ότι τα στοιχεία του αντικειμένου είναι παρόμοια σε φύση με αυτά των κάδων απορριμάτων, τα όποια είναι επαρκή για την κατασκευή ενός αντικείμενου που θα χρησιμοποιηθεί στην ανάλυση του προβλήματος.



Στην συγκεκριμένη περίπτωση η ομάδα οχημάτων αποτελείτε από ένα μόνο όχημα. Γνωρίζοντας ότι υπάρχουν και άλλα οχήματα με τιμές στο Weight της τάξης των 8000 από το υπόλοιπο το αρχείου. Πρόκειται για το όχημα που συλλέγει τους κάδους απορριμάτων οι οποίοι δεν είναι προσβάσιμοι στα μεγαλύτερα οχήματα λόγο της τοποθεσίας τους. Η άλλη ομάδα οχημάτων αποτελείτε από τα κανονικά απορριμματοφόρα που εξυπηρετούν όλους τους υπόλοιπους προσβάσιμους κάδους.


```
"vehicles": [{
    "id": 10,
    "Weight": 9000,
    "CurrentDepot": 164,
    "AvgFuelFactory": null,
    "AvgFuelRoute": 54,
    "FinalDepot": 164
}, {
    "id": 11,
    "Weight": 9000,
    "CurrentDepot": 164,
    "AvgFuelFactory": null,
    "AvgFuelRoute": 59,
    "FinalDepot": 164
}, {
    "id": 12,
    "Weight": 8000,
    "CurrentDepot": 164,
    "AvgFuelFactory": null,
    "AvgFuelRoute": 38,
    "FinalDepot": 164
```


ID: Μοναδικό όνομα του κάθε οχήματος.

Weight: Χωρητικότητα οχήματος. *-Από την διαφορά των 9000 και 1500 καταλαβαίνουμε ότι το πρώτο όχημα που αναφέρθηκε φέρει την προαναφερθέντα ιδιότητα σε σχέση με τα άλλα.*

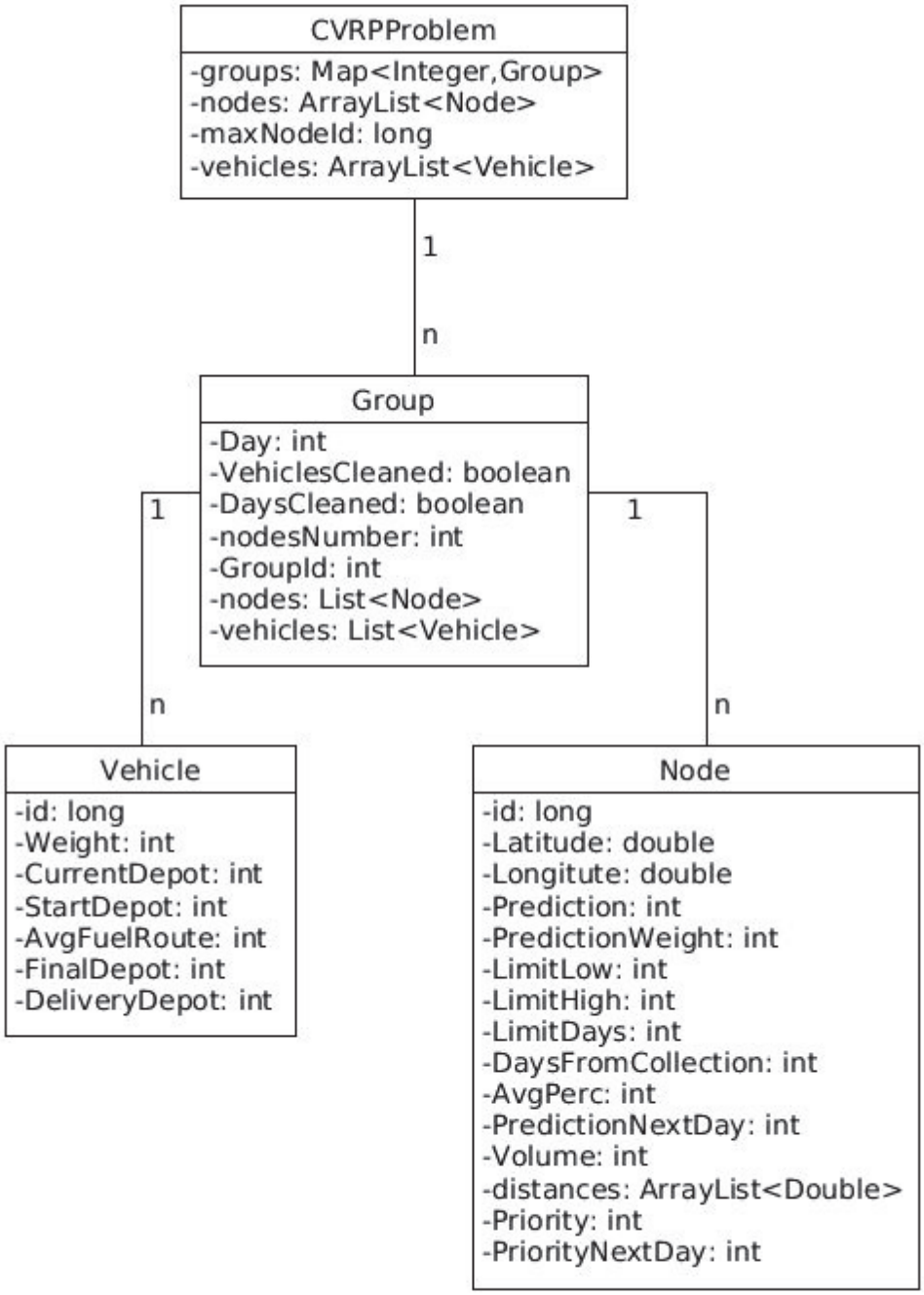
CurrentDepot: Ο κόμβος στον οποίον βρίσκεται το όχημα. Ο συγκεκριμένος είναι ο αρχικός κόμβος όπου όλα τα οχήματα ξεκινάνε την διαδρομή τους.

AvgFuelFactory: Κατανάλωση καυσίμου ανα km, ο οποίος βρίσκεται μέσα στην λογική του προγράμματος.

AvgFuelRoute: Ο μέσος όρος κατανάλωσης καυσίμου ανα διαδρομή.

FinalDepot: Ο κόμβος που θα επισκεφθεί το όχημα στο τέλος κάθε διαδρομής Στην περίπτωση μας είναι ο ίδιος με τον κόμβο εκκίνησης οπότε μπορούμε να υποθέσουμε ότι ο κόμβος 164 βρίσκεται στον χώρο στάθμευσης του στόλου.

Γνωρίζοντας αυτά μπορούμε να σχεδιάσουμε πιο ειδικευμένα τα βασικά κομμάτια μοντελοποίησης του προβλήματος.



Έτσι αρχικά μπορούμε να πούμε ότι έχουμε ένα πρόβλημα βελτιστοποίησης που αποτελείται από groups τα οποία αποτελούνται από οχήματα και κόμβους που πρέπει να εξυπηρετηθούν.

Κόμβος 164

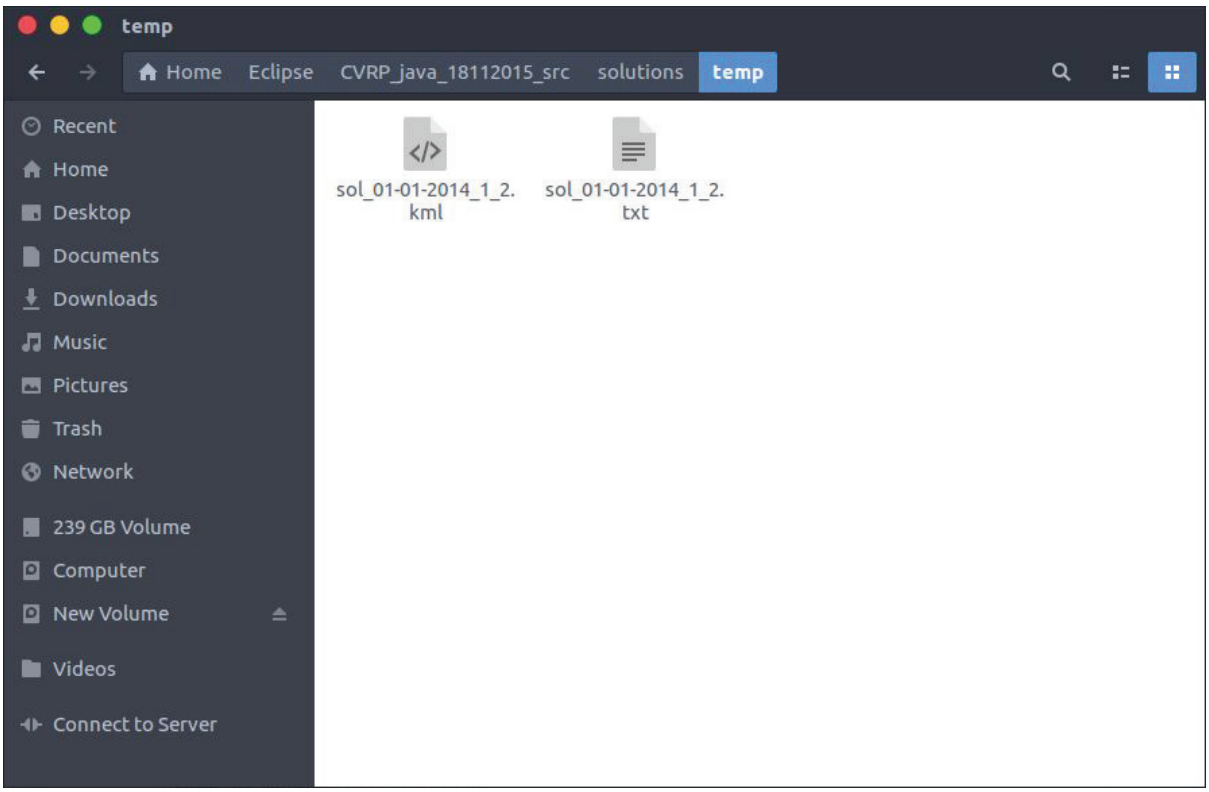
Κεφάλαιο 4.Τροποποιήσεις Εργαλείων

Στο πρόβλημα που λύνει η πτυχιακή χρησιμοποιούνται εργαλεία όπως το or-tools και το graphhopper για την λύση και αναπαράσταση του. Τα οποία προφανώς, δεν μπορούν να χρησιμοποιηθούν χωρίς κάποια παραμετροποίηση. Στην συνέχεια θα αναπτυχθούν οι τρόποι με τους οποίους τροποποιήθηκαν ώστε να γίνει δυνατή η απεικόνιση των λύσεων του προγράμματος.

4.1Τροποποιήσεις στο κύριο κώδικα

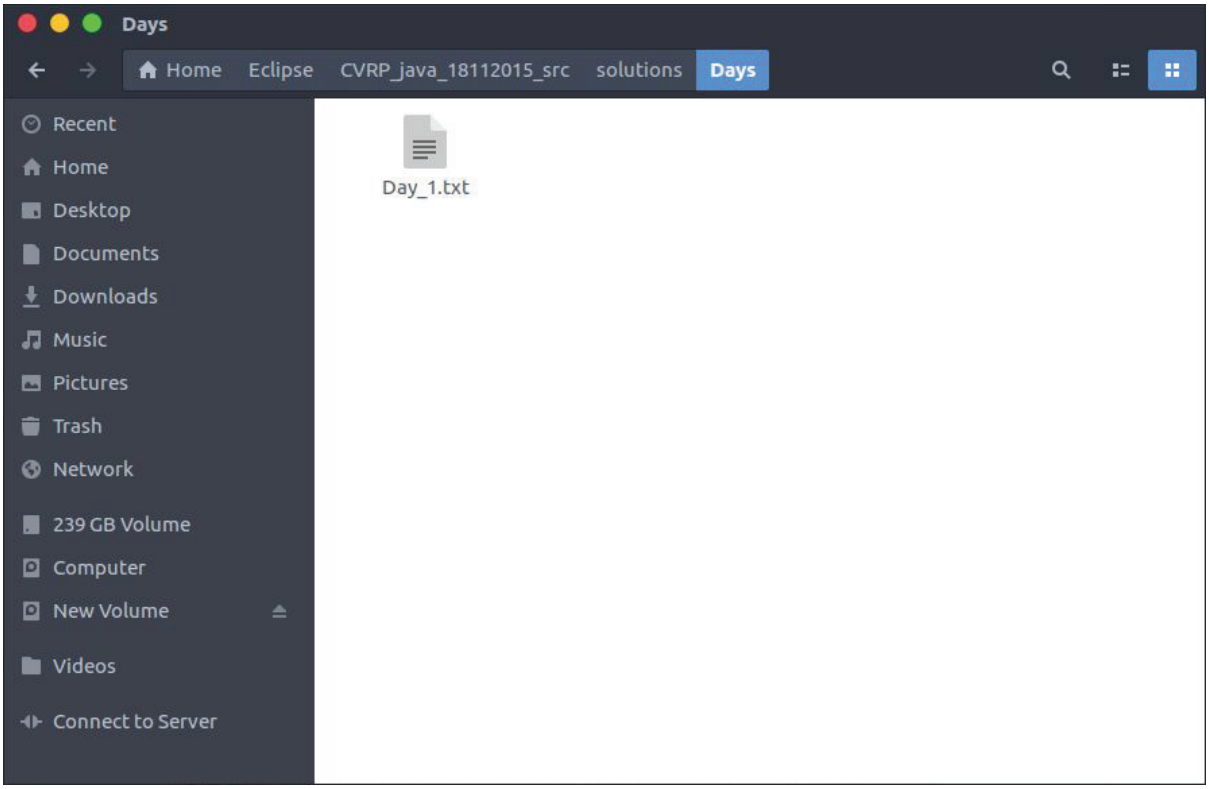
Ο αρχικός κώδικας του προγράμματος δημιουργούσε αρχεία με βάση των επιλεγμένο αριθμό οχημάτων, δηλαδή κάθε αρχείο ήταν η διαδρομή ενός οχήματος. Έχοντας όμως ως στόχο την απεικόνιση όλων τον οχημάτων ανά ημέρα μέσα στο γραφικό περιβάλλων τροποποιήθηκε ο κύριος κώδικας που λύνει το πρόβλημα ως εξής.

Στάδιο 1



Αρχικά δημιουργούνται τα αρχεία των λύσεων της μιας ημέρας και τοποθετούνται σε 2 φακέλους, στον φάκελο temp και vehicles. Πριν την δημιουργία κάποιου αρχείου στον φάκελο temp το πρόγραμμα διαγράφει ο,τι αρχείο υπάρχει μέσα σε αυτό που έμεινε από προηγούμενη εκκίνηση του προγράμματος.

Στάδιο 2



Στην συνέχεια το πρόγραμμα διαβάζει τα αρχεία που έχουν δημιουργηθεί και δημιουργεί ένα ενιαίο αρχείο που περιέχει ολα τα οχήματα που διάβασε απο τον φάκελο temp και το τοποθετεί στον φάκελο days. Όπου αυτό στην συνέχεια θα μπορεί να προβληθεί στην όψη per day view του γραφικού περιβάλλοντος. Μόλις ολοκληρωθεί και αυτη η διαδικασία τοτε τα αρχεία στον φάκελο temp θα διαγραφούν για να δημιουργηθούν τα αρχεία της επόμενης ημέρας, και θα επαναληφθεί η διαδικασία μέχρι ο αριθμός ημερών να φτάσει τις επιλεγμένες μέρες του χρήστη.

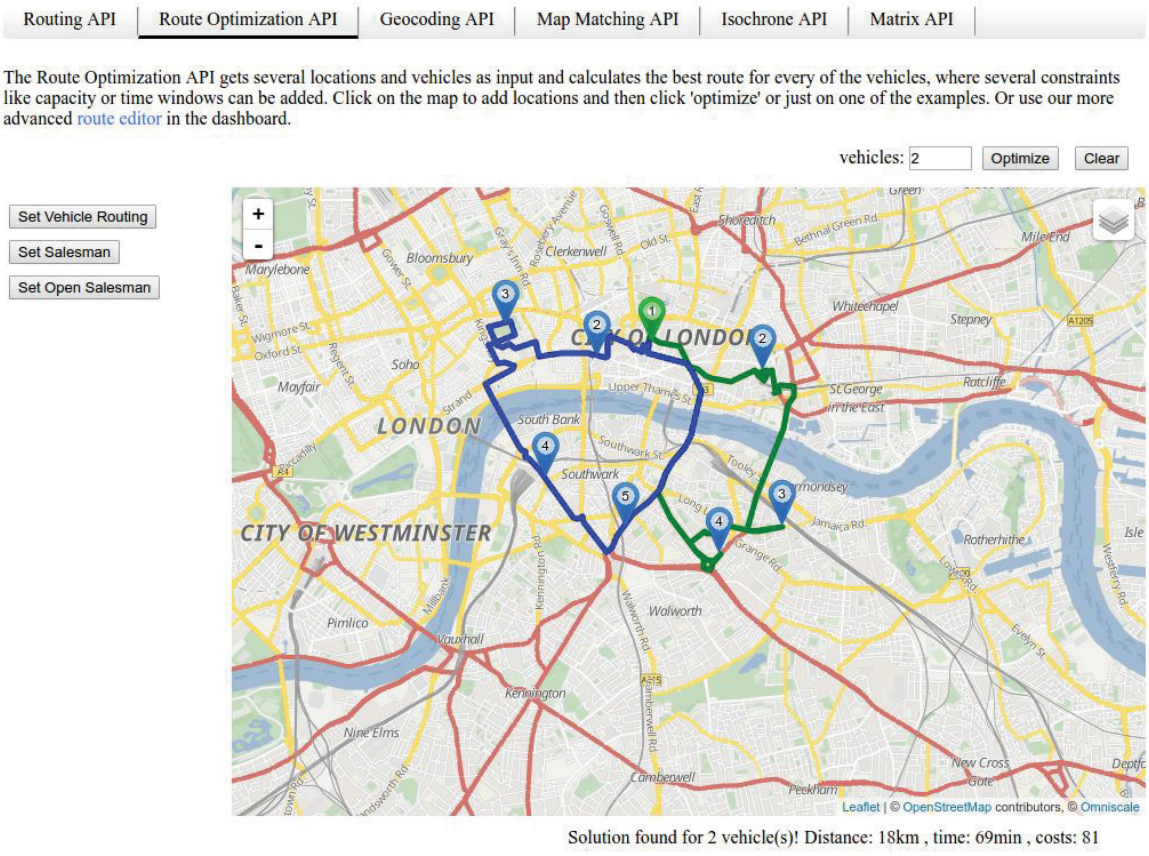
Το τελικό στάδιο που επιτρέπει την εμφάνιση στατιστικών στην όψη “Analytics”
ενεργοποιείται όταν ο χρήστης επιλέξει το κουμπί Analytics. Τότε το πρόγραμμα διαβάζει στατιστικά από τα αρχεία των φακέλων days και vehicles και δημιουργεί τα charts.

4.2Τροποποιήσεις στο Graphhopper

Παρόλο που το graphhopper παρέχει παρόμοιες λειτουργίες με αυτές του προγράμματος της πτυχιακής εργασίας. Το μόνο μέρος του κώδικα που χρησιμοποιήθηκε είναι αυτό της απεικόνισης σε χάρτη. Άλλες επιλογές ως προς την απεικόνιση των διαδρομών θα καθιστούσαν την σύνδεση στο ιντερνετ απαραίτητη ενώ με το grapphhopper δημιουργείτε αυτόματα ενας τοπικός εξυπηρετητής που μπορεί να φέρει εις πέρας την απεικόνιση των διαδρομών.

Η πρώτη τροποποίηση στον κώδικα του graphhopper έγινε στο μέγεθος του server buffer ο οποίος λάμβανε εντολές για την τοποθέτηση των nodes/κάδων στον χάρτη. Λόγο του μεγάλου αριθμού nodes έπρεπε να μεγαλώσει αυτος ο αριθμός ώστε να μπορέσει να αναγνώσει όλο το API Call.

Επόμενη τροποποίηση έγινε στο javascript κώδικα των html αρχειων του server. Στην όψη ανα όχημα δεν χρειάστηκε περαιτέρω τροποποίηση, αλλά η όψη ανα ημέρα δεν ήταν δυνατή με τον παρεχόμενο κώδικα.



Ο παρεχόμενος κώδικας έπαιρνε σημεία με το κλικ του ποντικιού και πατώντας το κουμπί optimize έστελνε ενα api call στον graphhoper server του website του για να λύσει και να αναπαραστήσει το πρόβλημα. Καθώς το πρόβλημα έχει λυθεί τοπικά από την βιβλιοθήκη or-tools ο κώδικας javascript τροποποιήθηκε έτσι ώστε να διαβάζει τοπικά ενα json αρχειο με την επιλεγμένη διαδρομή και στην συνέχεια να την προβάλει στον χάρτη. Αξίζει να σημειωθεί οτι λόγο της ασύγχρονης φύσης της javascript η εντολή για αναπαράσταση της διαδρομής στον χάρτη αποτύγχανε διότι αργούσε να ολοκληρωθεί η ανάγνωση του json αρχείου με αποτέλεσμα ο κώδικας να συνεχίσει να εκτελείτε και οταν φτάσει στην γραμμή οπου προβάλει τα αποτελέσματα η λίστα να είναι κενή. Αυτό λύθηκε με επιβολή συγχρονισμού κατά την εκτέλεση της ανάγνωσης του αρχείου ώστε οταν μονο τελειώσει η ανάγνωση να μπορεί να συνεχίσει η ροή του προγράμματός.

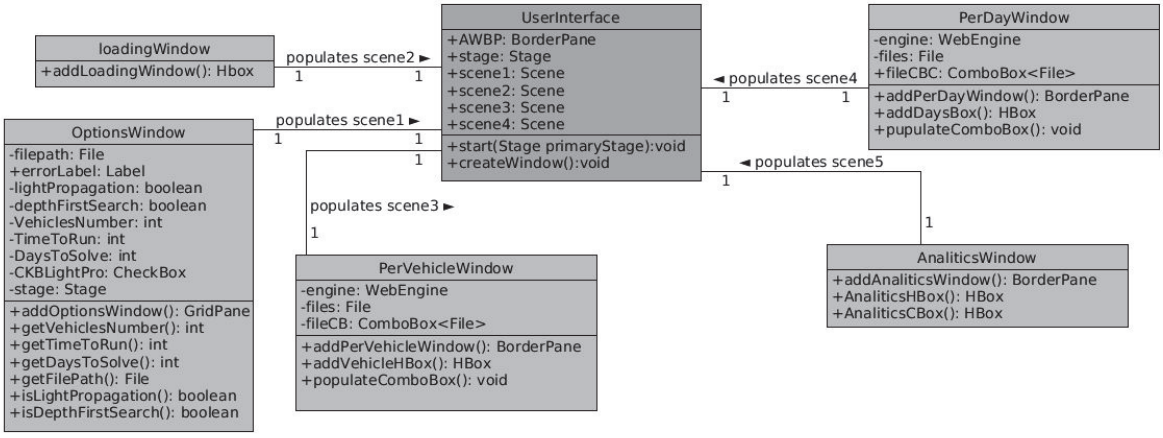
Δημιουργήθηκε επίσης μια συνάρτηση όπου αλλάζει τυχαία τα χρώματα των διαδρομών κάθε φορά που ανανεώνεται η σελίδα δίνοντας την δυνατότητα στον χρήστη να αλλάξει τα χρώματα των διαδρόμων με το κουμπί “Change Colors” άμα αυτός το επιλέξει. Τέλος υπάρχει μια ακόμα html σελίδα που παρέχει όλα τα παραπάνω συν την απεικόνιση των κάδων αριθμημένα με το κουμπί “Toggle Points”.

Κεφάλαιο 5.Σχεδιασμός Γραφικού Περιβάλλοντος

Σε αυτό το κεφάλαιο θα αναπτυχθεί το μέρος του σχεδιασμού του γραφικού περιβάλλοντος. Η βιβλιοθήκη που χρησιμοποιήθηκε ονομάζεται JavaFX και είναι μέρος της έκδοσης java της oracle.

Στην JavaFX τα συστατικά που απαρτίζουν ένα γραφικό περιβάλλων ξεκινάνε με το Stage, που είναι το περίγραμμα του παραθύρου. Στην συνέχεια προσθέτονται μέσα σε αυτό τα Scenes, τα όποια είναι το περιεχόμενο του παραθύρου. Κάθε scene μπορεί να αποτελείτε από δοχεία όπως HBox, VBox, BorderPane κ.α τα οποία κρατάνε στοιχεία όπως κουμπιά, κείμενα, εικόνες ή ακόμα και άλλα δοχεία μέσα στα δοχεία και ξεχωρίζουν μεταξύ τους με τον τρόπο με τον οποίο κατατάσσουν τα στοιχεία τους. Για παράδειγμα το HBox κατατάσσει τα στοιχεία του οριζόντια, ανάλογα με την σειρά με την όποια προστίθενται στο δοχείο. Το VBox παρόμοια αλλά κάθετα. Το BorderPane χρησιμοποιεί τα σημεία του ορίζοντα για να ορίσει την τοποθεσία των στοιχείων του. Όταν θέλουμε να αλλάξουμε παράθυρο στην εφαρμογή μας τότε απλά ενεργοποιούμε το Scene που μας ενδιαφέρει, το οποίο περιέχει τα δικά του δοχεία και στοιχεία.

5.1 Κλάσεις Γραφικού Περιβάλλοντος



Στην φωτογραφία αναγράφονται οι κλάσεις που απαρτίζουν το γραφικό περιβάλλον της εφαρμογής. Στην κύρια κλάση UserInterface υπάρχει το Stage και

όλα τα Scenes τα οποία αλλάζουν μεταξύ τους θέση μέσα στο Stage ανάλογα με το παράθυρο που έχει επιλέξει ο χρήστης.

Στην προκειμένη περίπτωση κάθε άλλη κλάση εκτός της UserInterface λειτουργεί σαν ένα δοχείο όπου επιστρέφει στο κατάλληλο Scene κάποιο δοχείο με στοιχεία όπως hbox, vbox, borderpane και gridpane. Καθώς το γραφικό περιβάλλον δημιουργείται με την έναρξη της εφαρμογής, στοιχεία όπως το combobox τα οποία αναγράφουν τα αποτελέσματα επεξεργασίας του προγράμματος που δημιουργούνται κατά την διάρκεια λειτουργίας της. Γιαυτό το λόγο δημιουργήθηκαν συναρτήσεις οι οποίες ανανεώνουν το περιεχόμενο των combobox αφότου τελειώσουν οι υπολογισμοί ώστε να μπορούν να εμφανιστούν σαν επιλογές για ανάλυση από το γραφικό περιβάλλων. Παρόμοια συμπεριφορά φέρει και η κλάση AnaliticsWindow, όπου το Scene της (scene5) δημιουργείτε όλο αφότου τελειώσουν οι υπολογισμοί μαζί με τα combobox.

5.2 Γραφικό Περιβάλλον

Σε αυτό το κεφάλαιο θα αναπτυχθεί το μέρος του γραφικού περιβάλλοντος του προγράμματος το οποίο χωρίζεται σε δυο μέρη. Το αρχικό παράθυρο όπου βρίσκονται οι ρυθμίσεις το προγράμματος και το παράθυρο που εμφανίζονται τα αποτελέσματα μετά την εκτέλεση του αλγόριθμου.

5.2.1 Αρχικό παράθυρο

Για το αρχικό παράθυρο έχει χρησιμοποιηθεί η βιβλιοθήκη JavaFX. Το παράθυρο προσφέρει τρεις βασικές ρυθμίσεις οι οποίες θα χρησιμοποιηθούν ως παράμετροι εκκίνησής το αλγόριθμου και 2 προαιρετικές. Με την σειρα που εμφανίζονται στο γραφικό περιβάλλον είναι:

1. Αριθμός οχημάτων
2. Χρόνος λειτουργίας
3. Ημέρες
4. Light Propagation
5. Depth First Search
6. Επιλογή αρχείου

1. Η πρώτη επιλογή δίνει την δυνατότητα στον χρήστη να επιλέξει τον αριθμό οχημάτων που έχει στην διάθεση του ο αλγόριθμος για την εύρεση διαδρόμων. Σε αρκετές περιπτώσεις ο αλγόριθμος χρησιμοποιεί περισσότερα από δύο οχήματα(όπως προαναφέρθηκε, ένα για μικρών διαστάσεων για τους στενούς δρόμους της Ναυπάκτου και ένα κανονικών) άμα του δοθεί η δυνατότητα.

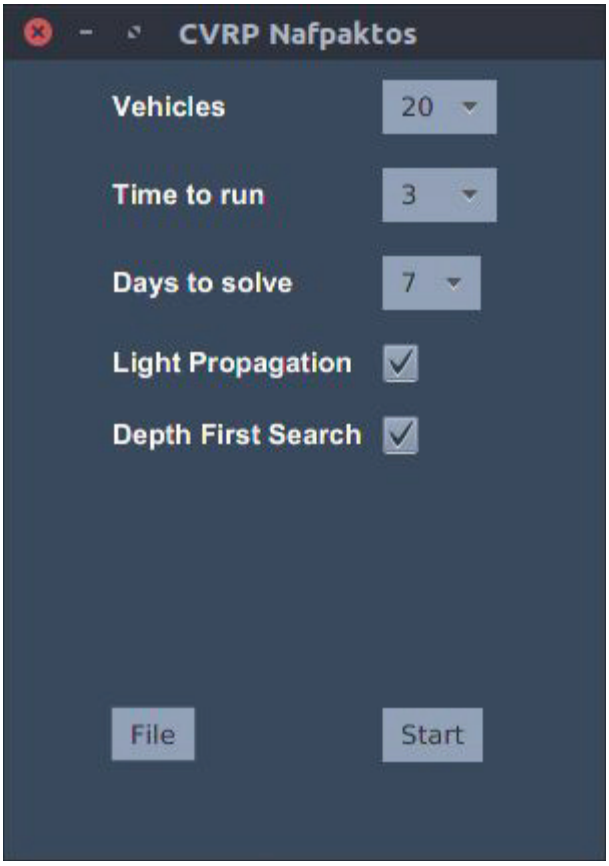
2. Δεύτερη επιλογή απευθύνετε στην διαθέσιμη ώρα που έχει ο αλγόριθμος ώστε να λύσει το πρόβλημα. Η επιλογή είναι συνδεδεμένη με την setTimeLimitMs που αναφέρθηκε στο κεφάλαιο 2.

3. Το πρόγραμμα έχει την δυνατότητα χρησιμοποιώντας τα δεδομένα Dynacargo να προβλέψει την ποσότητα κάθε κάδου και για τις ακόλουθες μέρες. Αυτή η επιλογή δείχνει πόσες ημέρες θα προβλεφθούν και θα λυθούν.

4. Light Propagation είναι μια συμπληρωματική επιλογή που μπορεί να χρησιμοποιηθεί μόνο μαζί με το Depth First Search. Αυξάνει την ποιότητα των αποτελεσμάτων αλλά κάνει την διαδικασία αναζήτησης πιο αργή και καταναλώνει περισσοτέρους επεξεργαστικούς πόρους.

5. Το DFS είναι μία τεχνική αναζήτησης που κάνει τον αλγόριθμο να εκτελεί μεγαλύτερες αναζητήσεις για κάθε επιλογή βέλτιστης διαδρομής πριν προχωρήσει στην επόμενη, αυξάνοντας τις πιθανότητες εύρεσης καλύτερων διαδρομών.

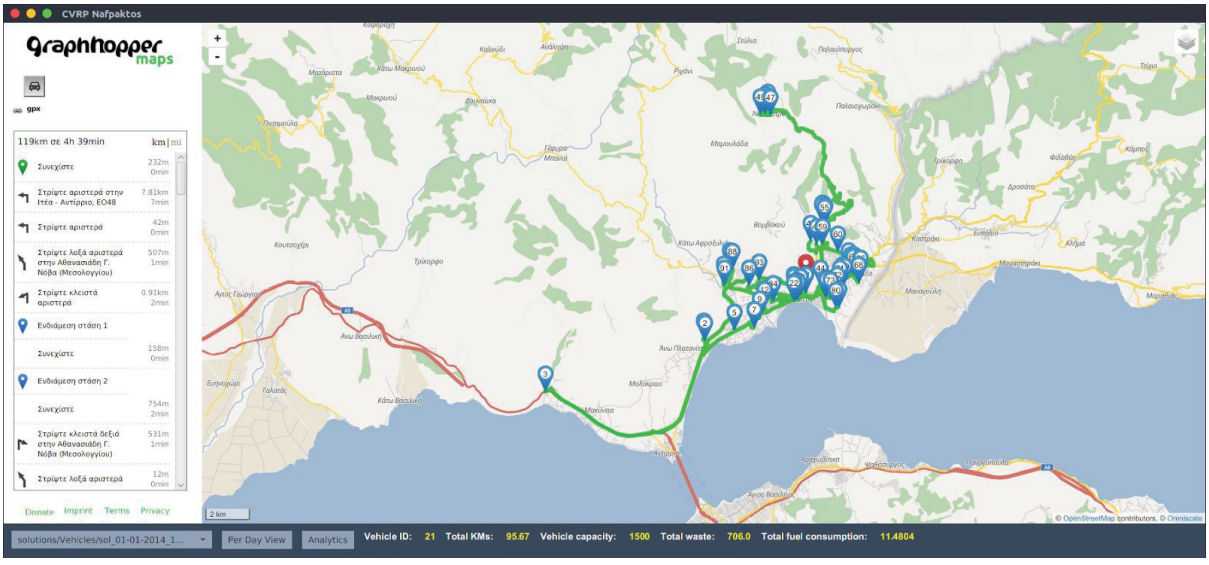
6. Η τελευταία επιλογή αναφέρετε στην επιλογή του αρχείου που θα λύσει το πρόγραμμα. Αυτά τα αρχεία είναι τα αρχεία μετρήσεων του project Dynacargo. Το τελευταίο στοιχείο του πρώτου παραθύρου είναι το κουμπί εκκίνησης εκτέλεσης του αλγόριθμου, που όταν τελειώσει θα εμφανίσει το δεύτερο παράθυρο.

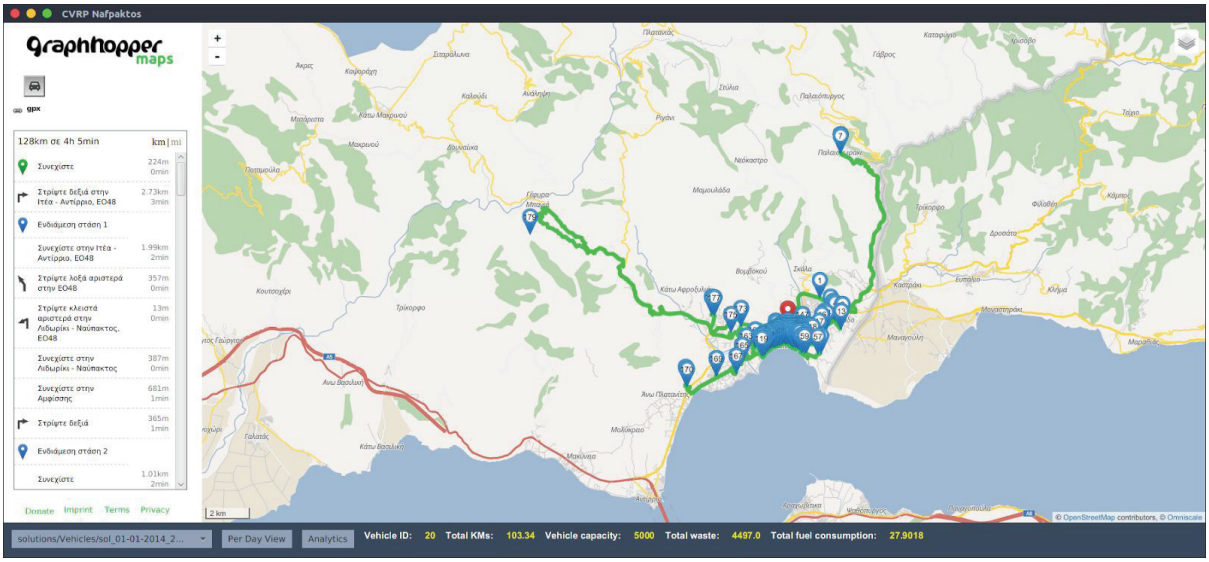


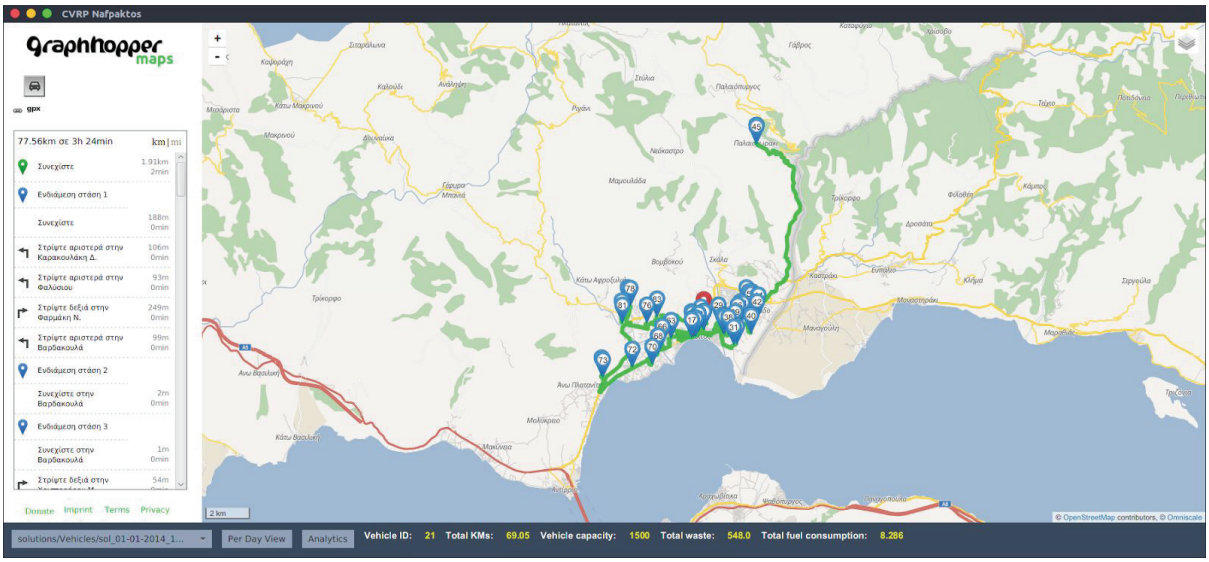
Αρχικό παράθυρο ρυθμίσεων εκκίνησης του αλγόριθμου.

5.2.2 Παράθυρο αποτελεσμάτων

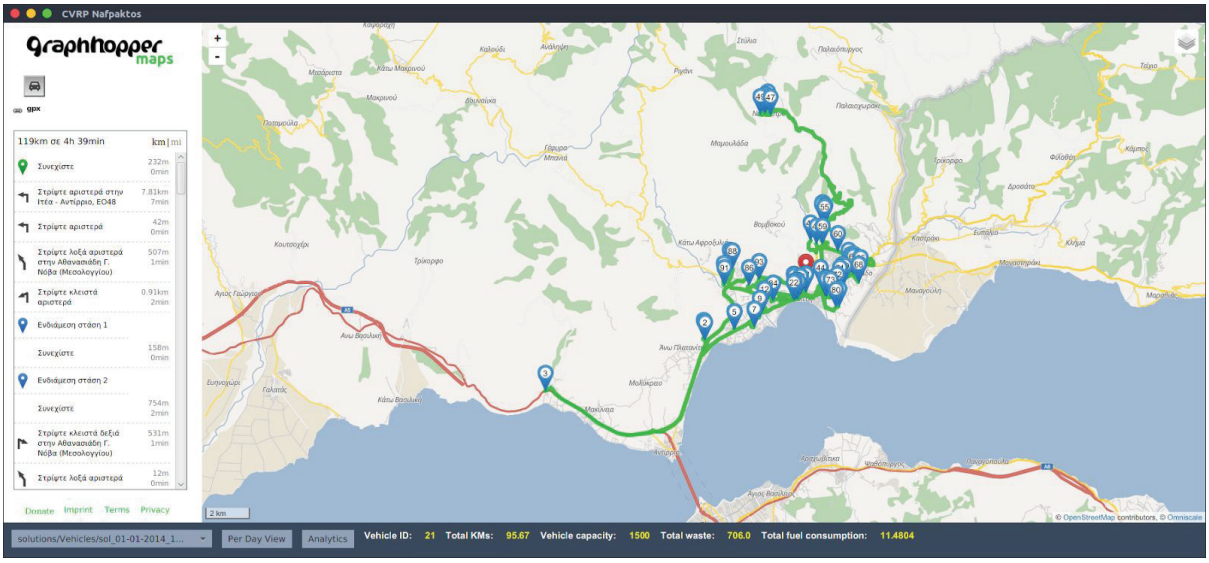
Αφού ολοκληρωθεί η δημιουργία διαδρομών τότε εμφανίζετε το παράθυρό αποτελεσμάτων με τα στατιστικά για κάθε αποτέλεσμα.

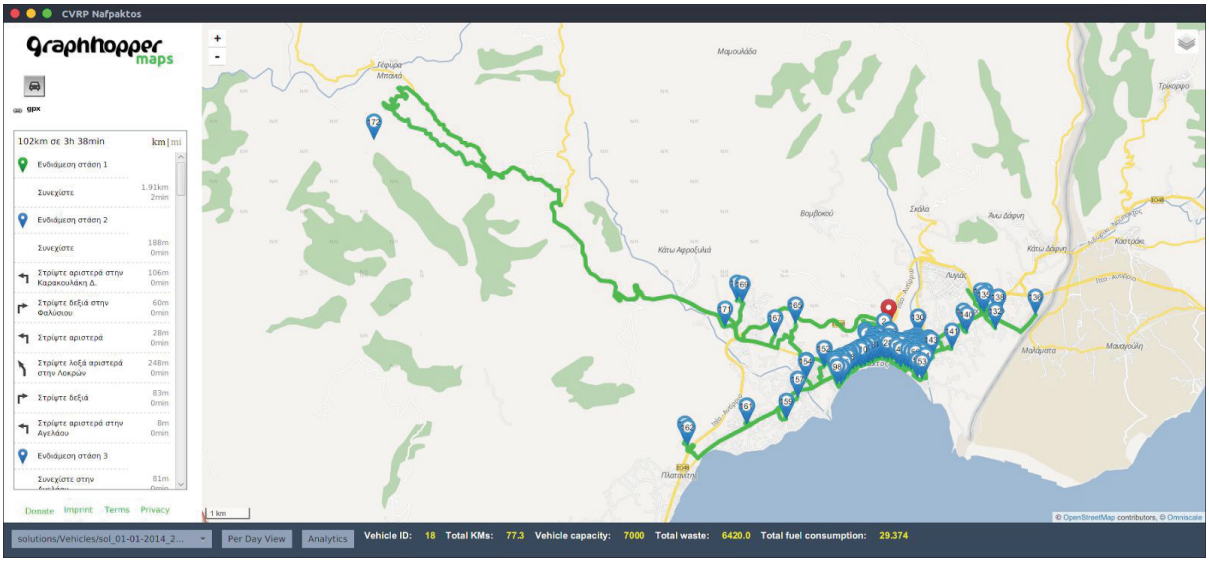




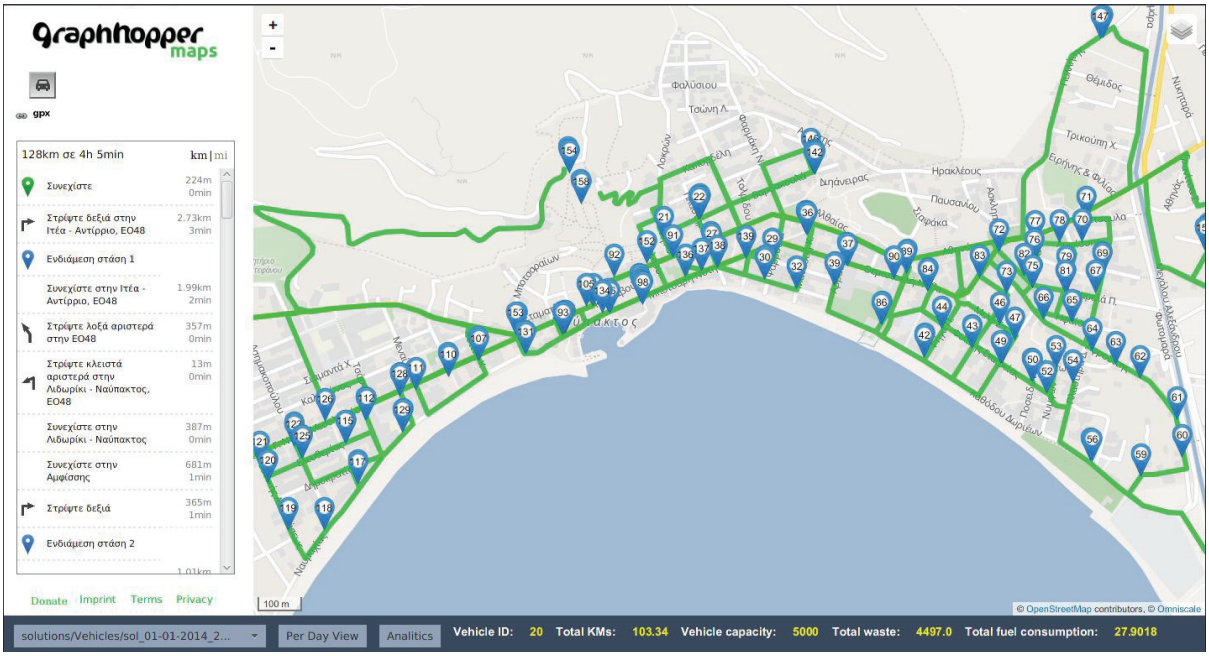


Το παράθυρο αυτό κάνει χρήση του graphhoper API στον τοπικό server που έχει ξεκινήσει αυτόματα μαζί με το πρόγραμμα για να σχεδιάσει την κάθε διαδρομή ξεχωριστά.





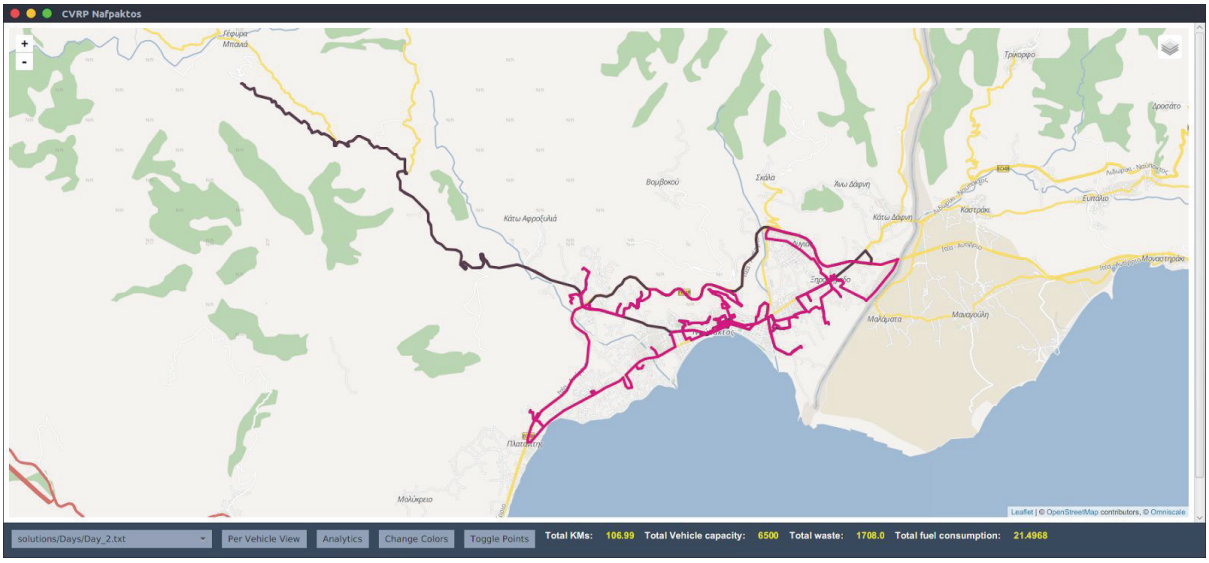
Κάτω αριστερά υπάρχει επιλογή για τον χρήστη να διαλέξει πια διαδρομή θέλει να εμφανιστεί στον χάρτη. Επίσης αναγράφονται στατιστικά για το όχημα της κάθε διαδρομής, τα χιλιόμετρα που διένυσε το όχημα, την χωρητικότητα του, τα κιλά σκουπιδιών που μάζεψε από τους κάδους και τα λίτρα κατανάλωσης καυσίμου.

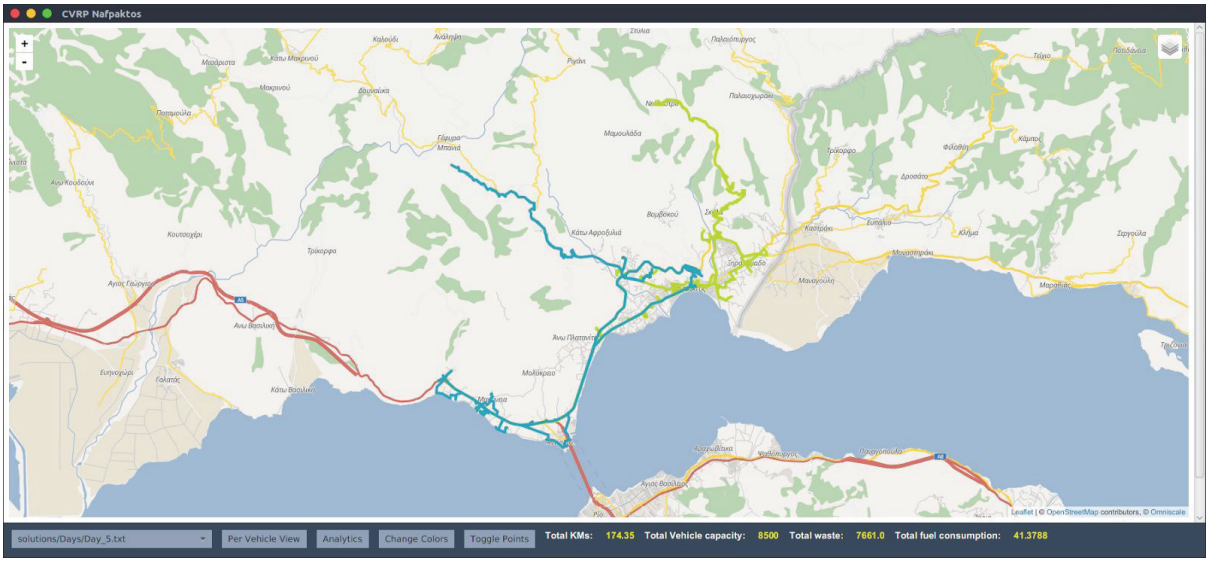


Στο παράθυρο επίσης εμφανίζονται όλοι οι κάδοι οι οποίοι έχουν επιλεχθεί να εξυπηρετηθούν και με πια σειρά, που αναγράφεται στο νούμερο που έχει πάνω το κάθε στοιχείο του χάρτη. Στα αριστερά αναγράφονται οδηγίες για την διαδρομή. Δεξιά της επιλογής διαδρομών υπάρχει το κουμπί που αναγράφει “Per Day View”.

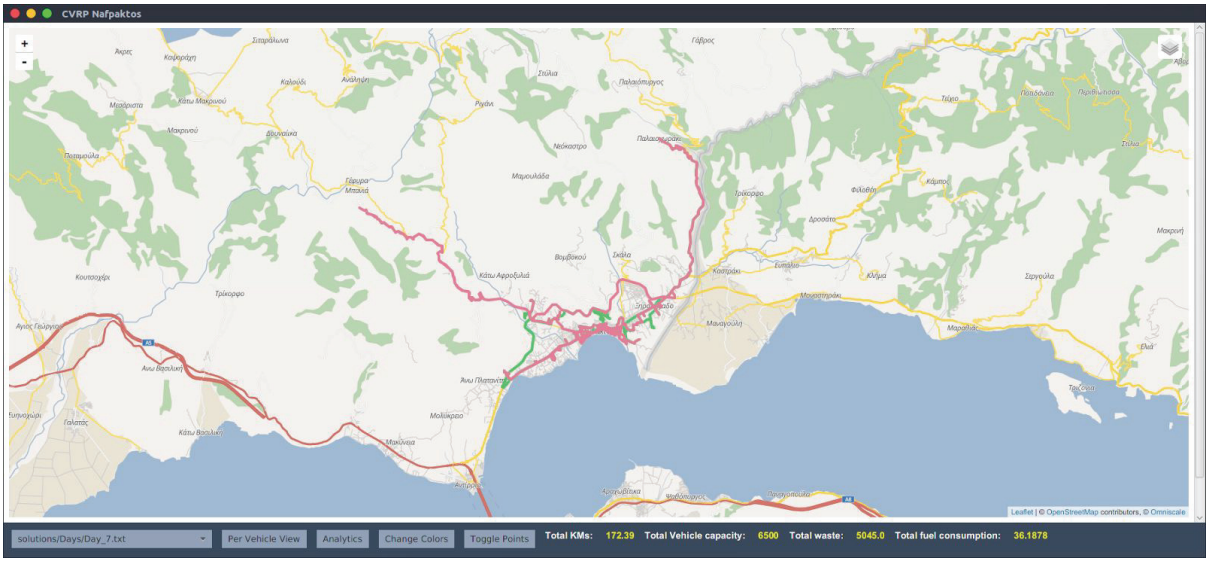
Αυτό εμφανίζει μια άλλη όψη του χάρτη όπου εδώ αναγράφετε η διαδρομή ανα ημέρα. Στον χάρτη εμφανίζονται όλες οι διαδρομές της επιλεγμένης ημέρας από όλα τα οχήματα που χρησιμοποιήθηκαν και στο κάτω μέρος τα στατιστικά των διαδρομών αυτών αθροιστικά. Συγκεκριμένα τα συνολικά χιλιόμετρα που διανύθηκαν από όλα τα οχήματα την ήμερα αυτή, το συνολικό διαθέσιμο σε κιλά φορτίο, πόσο χρησιμοποιήθηκε και τα λίτρα κατανάλωσης σε καύσιμο.

Το κουμπί “Per Vehicle View” αλλάζει την όψη το προγράμματος πίσω σε ανάλυση ανα διαδρομή.

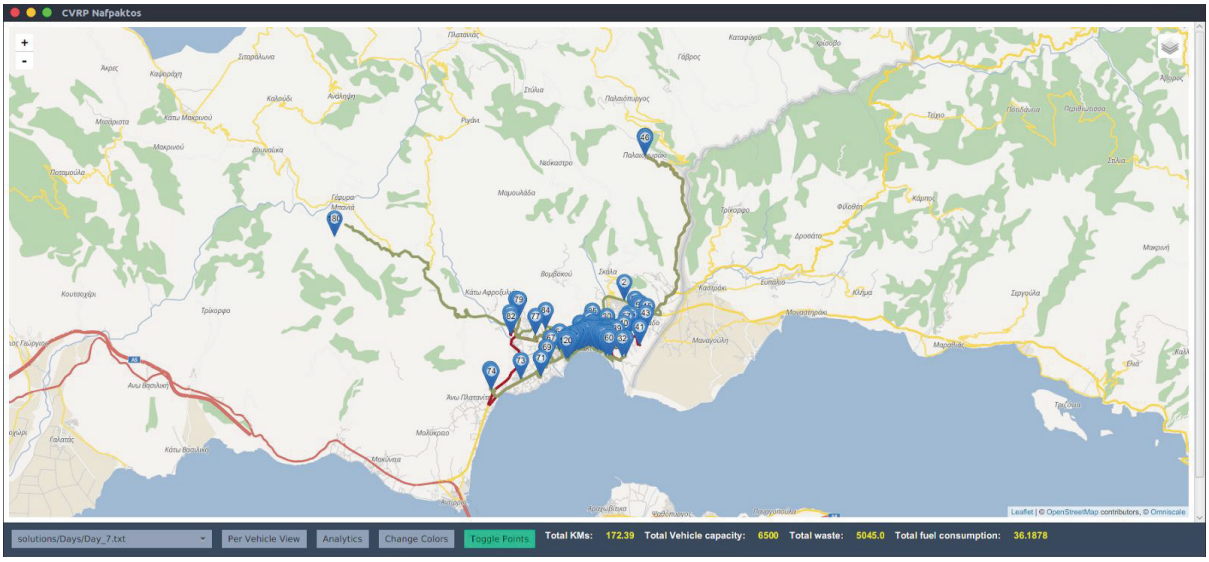




Το κουμπί “Toggle Points” εμφανίζει τους κάδους σε μορφή σημείων όπως και στην όψη ανα διαδρομή. Το κουμπί “Change Colors” αλλάζει τα χρώματα αναπαράστασης της διαδρομής στον χάρτη.



Toggle Points απενεργοποιημένο



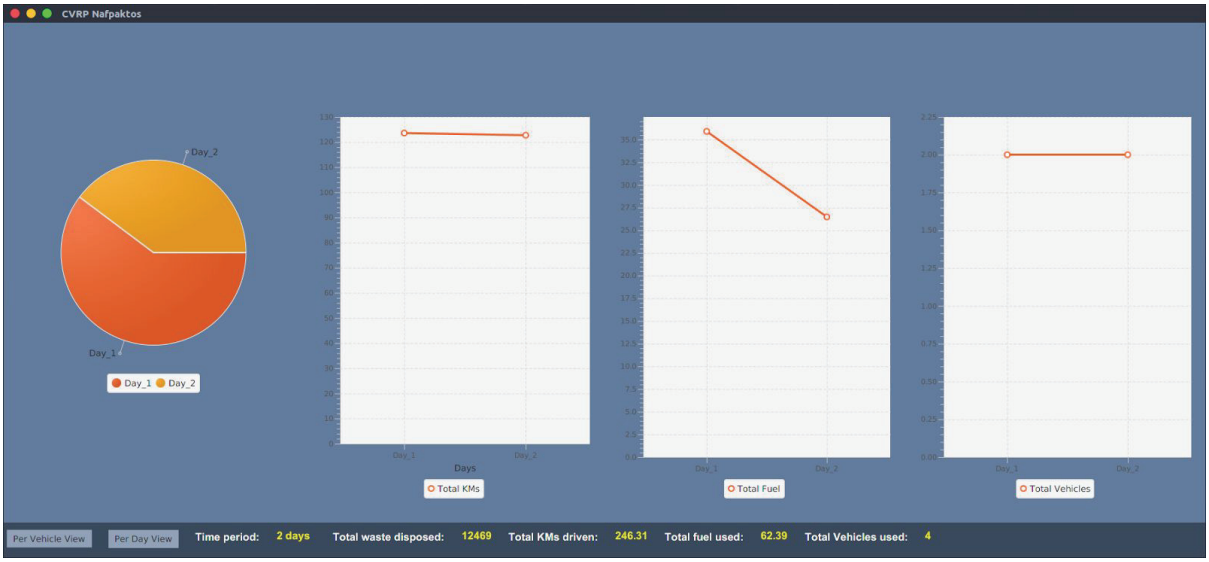
Toggle Points ενεργοποιημένο

Το κουμπί “Analytics” εμφανίζει τα στατιστικά όλων των ημερών σε μορφές γραφημάτων για σύγκριση.

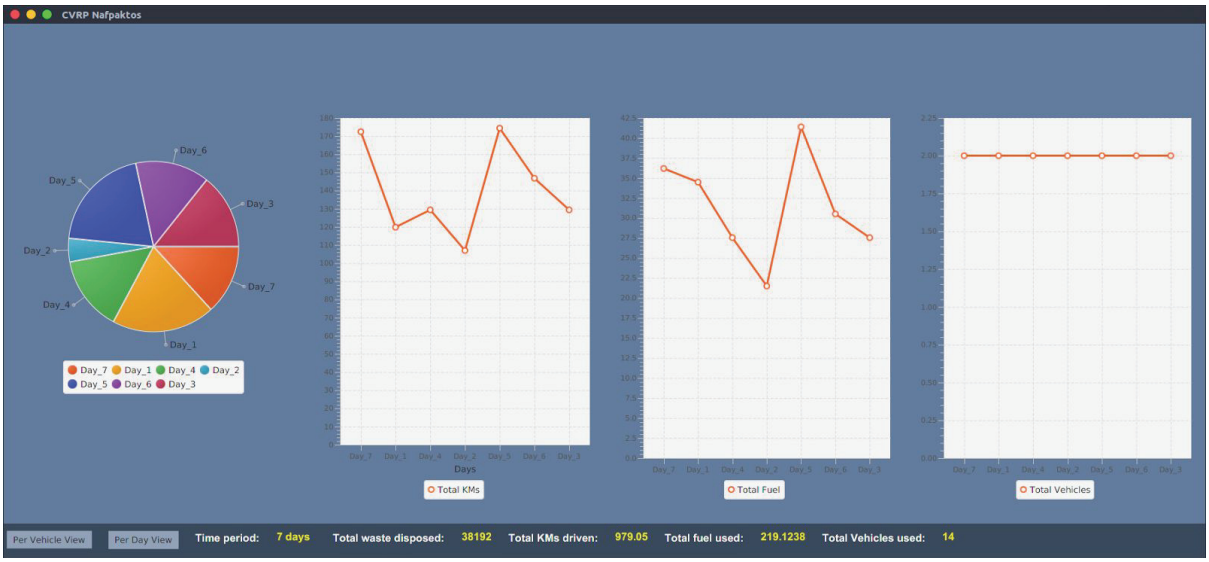
Ξεκινώντας από τα αριστερά αρχικά εμφανίζεται ένα pie chart, με τα ποσοστά αποκομιδής σκουπιδιών σε κιλά ανα ημέρα. Ο χρήστης μπορεί να κάνει κλικ σε κάθε κομμάτι του pie chart για να εμφανίσει τα κιλά σκουπιδιών αυτής. Ακολουθεί ένα γράφημα δίανυσης χιλιομέτρων ανά ημέρα απο τα οχήματα συνολικά της κάθε ημέρας. Στην συνέχεια υπάρχει ένα γράφημα για την κατανάλωση καυσίμων σε λίτρα και τέλος ένα γράφημα για τα οχήματα που χρησιμοποιήθηκαν ανα ημέρα.

Στο κάτω μέρος εμφανίζονται αναλυτικά, οι ημέρες που αναλύθηκαν από τον αλγόριθμο, και συνολικά για αυτές τις ήμερες τα κιλά σκουπιδιών που συλλέχθηκαν

τα χιλιόμετρα που διανύθηκαν, τα λίτρα που καταναλώθηκαν από τα οχήματα, και τέλος πόσα οχήματα χρησιμοποιήθηκαν για όλες τις ημέρες.



Analytics όψη δυο ημερών



Analytics όψη επτά ημερών

Κεφάλαιο 6. Συμπεράσματα και μελλοντικές επεκτάσεις

Η εφαρμογή της πληροφορικής στην διαχείριση στόλων αποφέρει μεγάλα κέρδη στους οργανισμούς που χρησιμοποιούν αλγορίθμους βελτιστοποίησης διαδρομών. Ακόμα και όταν οι διαδρομές δεν είναι βέλτιστες η διαφορά μεταξύ απλών διαδρομών προγραμματισμένων χωρίς την βοήθεια τέτοιων προγραμμάτων είναι πολύ μεγάλη. Εκτός της αποκομιδής σκουπιδιών τέτοιοι αλγόριθμοι μπορούν, και έχουν εφαρμοστεί και σε πολλούς άλλους τομείς, όπως οι μεταφορές υλικών μέσω όλων των μέσων μεταφοράς. Με την απεικόνιση των διαδρομών και οδηγιών γίνεται ευκολότερη η δουλειά των οδηγών τέτοιων οχημάτων που στο μέλλον ίσως αντικατασταθούν από συστήματα αυτόνομης οδήγησης που θα ανταλλάσσουν πληροφορίες απευθείας με προγράμματα βελτιστοποίησης διαδρομών και θα μπορούν να αλλάζουν τις διαδρομές εν κινήση ανάλογα με τις συνεχώς εναλλασσόμενες συνθήκες της διαδρομής για εξοικονόμηση χρόνου και μέγιστη ασφάλεια του ωφέλιμου φορτίου.

Καθώς ο έλεγχος των αυτοκινήτων αφαιρείται σταδιακά από τον χρήστη με την δημιουργία συστημάτων ολοκληρωτικής πλοήγησης για ιδιωτικά αμάξια. Προγράμματα fleet management σαν αυτό της διπλωματικής θα αρχίσουν να επεκτείνονται έξω από την εμπορική χρήση τους και θα αρχίσουν να επηρεάζουν πιο άμεσα τους καταναλωτές. Τα αμάξια θα συλλέγουν τα ίδια, δεδομένα για την τοποθεσία τους τα οποία θα επεξεργάζονται από ειδικούς εξυπηρετητές που θα λαμβάνουν υπόψιν τυχόν ατυχήματα που υπάρχουν στους δρόμους, δημόσια έργα, ή την παρουσία μεγάλου αριθμού αμαξιών σε κρίσιμους κόμβους, και θα δημιουργούν βέλτιστες διαδρομές τις οποίες τα αυτοκίνητα θα χρησιμοποιούν για να μεταφέρουν τον επιβάτη στην τοποθεσία που έχει επιλέξει χωρίς περαιτέρω αλληλεπίδραση από την μεριά του. Τα αποτελέσματα χρήσης τέτοιας τεχνολογίας θα επιφέρουν ελαχιστοποίηση ατυχημάτων αλλά και τεράστια μείωση μποτιλιαρίσματος στους δρόμους.

Θα μπορούσε να ειπωθεί ότι οι πρώτες ενέργειες βελτιστοποίησης διαδρομών έγιναν με τις πρώτες προσπάθειές του ανθρώπου να εξερευνήσει το διάστημα. Σε αυτή την περίπτωση μείζων σημασία έφερε η χρήση των καυσίμων καθώς δεν υπήρχε τρόπος ανεφοδιασμού και έπρεπε να γίνουν ακριβείς υπολογισμοί ώστε να μπορέσει η ρουκέτα να δραπετεύσει από την βαρύτητα της γης αλλά και να μείνει σε σταθερή τροχιά γύρω από αυτή. Καθώς η υπολογιστική ισχύ αυξάνεται, σειρά παίρνουν τα ποιο καθημερινά προβλήματα από βελτιστοποίηση διαδρόμων αεροσκαφών μέχρι και μελλοντική ολική αυτοματοποίηση των ιδιωτικών και μαζικών μέσων μεταφοράς.