****

MSc Business Analytics

Large Scale Optimization

**Part-time program 2018-2020**

**Participants:**

Eirini Kokolaki (P2821813), Prasinos Michalis (P2821823),

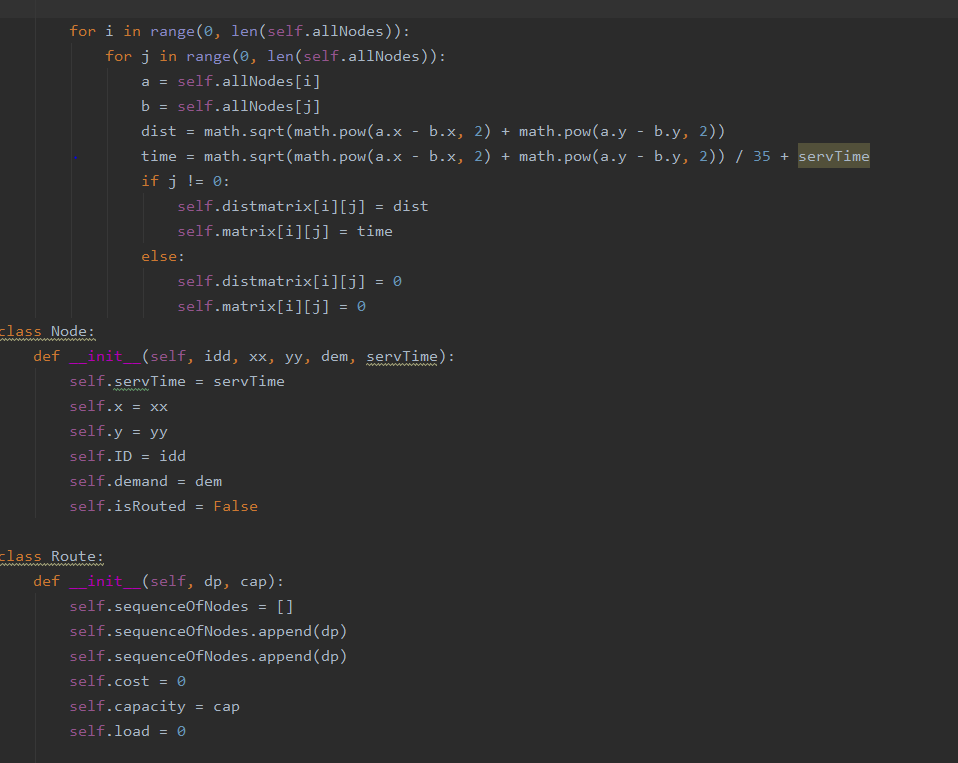
Mirsini Zourou (P2128828), Giorgos Vogiatzis (P2821827)

***1. Να παρέχει την υποδομή για την επίλυση του εξεταζόμενου προβλήματος και να δημιουργεί το input***

***a. Κλάσεις (π.χ. Διαδρομή, Σημεία κτλ.)***

***b. Πίνακες (Κόστος μεταξύ σημείων, Χρονικές Αποστάσεις…)***





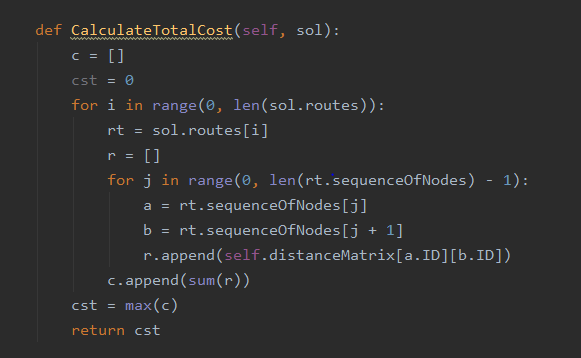
Για την λύση του προβλήματος χρησιμοποιούμε ένα VRP Model.

Φτιάξαμε τις κλάσεις model, node (τοποθεσίες) και route (οι διαδρομές/φορτηγά). Στην κλάση model δημιουργείται το def Build Model που περιλαμβάνει τον αρχικό node (0 / αποθήκη), το συνολικό capacity κάθε φορτηγού 3000 κιλά και τα 200 συνολικά σημεία εξυπηρέτησης, τον πίνακα των σημείων ορίζοντας για κάθε σημείο τις συντεταγμένες του, την ζήτηση και τον χρόνο εξυπηρέτησης που είναι ο χρόνος μεταξύ δυο σημείων συν δεκαπέντε λεπτά για την εκφόρτωση. Έτσι, δημιουργούνται δυο πίνακες, ο πίνακας των αποστάσεων distmatrix (ο οποίος υπολογίζεται με τη ευκλείδεια απόσταση μεταξύ σημείων) και ο πίνακας του χρόνου (ο οποίος υπολογίζεται από την απόσταση διαιρεμένη με το 35 χλμ/ώρα συν τα δεκαπέντε λεπτά της εκφόρτωσης. Επειδή στο πρόβλημα μας δεν μας ενδιαφέρει η απόσταση από το τελικό σημείο προς την αποθήκη, στους δυο σχετικούς πίνακες έχουμε ορίσει όλα τα σημεία προς την αποθήκη ίσον με το μηδέν. Για το πρόβλημά μας θα χρησιμοποιήσουμε τον πίνακα που αφορά των χρόνο εξυπηρέτησης.

Η κλάση node περιλαμβάνει τον servtime (χρόνο εξυπηρέτησης), το x και y (τις συντεταγμένες), το id του Node, την ζήτηση (demand), το isRouted (κάνει flag εάν ο Node έχει συμπεριληφθεί στην διαδρομή ή όχι).

Η κλάση Route περιλαμβάνει μια λίστα με την σειρά των nodes, το κόστος της κάθε διαδρομής, το Capacity του φορτηγού είναι το όριο 3000 κιλά που μπορεί να εξυπηρετήσει το κάθε φορτηγό και το Load είναι η ποσότητα που αφορά το πόσο γεμάτο είναι το κάθε φορτηγό, το οποίο αλλάζει κάθε φορά που προστίθεται ένα νέο node στην διαδρομή.

***2. Υλοποιείστε μία μέθοδο που θα δέχεται σαν όρισμα μία οποιαδήποτε λύση του προβλήματος και θα υπολογίζει την τιμή της αντικειμενικής συνάρτησης αυτής της λύσης.***



Η λύση του προβλήματός μας θα είναι μια λίστα με 25 διατάξεις σημείων, όπου κάθε διάταξη αποτελεί την διαδρομή ενός φορτηγού.

Επομένως, η αντικειμενική μας συνάρτηση (calculate total cost) σύμφωνα με τον ορισμό του προβλήματος θα μας επιστρέφει τον χρόνο που χρειάζεται για να πραγματοποιηθεί η μεγαλύτερη διαδρομή από άποψη χρόνου.

Στόχος είναι η ελαχιστοποίηση του χρόνου της μεγαλύτερης διαδρομής.

Στοιχείο της λύσης είναι η προσθήκη ενός Node σε μια διαδρομή μέχρι η ζήτηση να μην υπερβαίνει την χωρητικότητα των οχημάτων.

***3. Κατασκευάστε έναν κατασκευαστικό αλγόριθμο, ο οποίος να δημιουργεί μία αρχική λύση.***

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Η διαδικασία που ακολουθεί ο κατασκευαστικός αλγόριθμος είναι η εξής:

* Self.**SetRoutedFlagToFalseForAllpoints** () :  
  Θέτει σε όλα τα σημεία την τιμή False, δηλαδή ότι όλα σημεία δεν έχουν δρομολογηθεί ακόμα.

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

* Self.**MinimumInsertions** () :   
  Δημιουργεί μια κενή αρχική λύση (μια κενή λίστα με 0 κόστος).

Ταξινομεί σε αύξουσα σειρά όλα τα σημεία σε σχέση με την απόστασή τους από την αποθήκη (depot) και παίρνει τα πρώτα 25 σημεία και τα βάζει σε 25 διαφορετικά φορτηγά.   
Επομένως στο πρώτο βήμα δημιουργεί 25 routes που ξεκινάνε και τελειώνουν στην αποθήκη και έχουν ένα σημείο ενδιάμεσα.

Με την μέθοδο roundrobin, για όλα τα σημεία που δεν έχουν δρομολογηθεί ακόμα, υπολογίζει επαναληπτικά το κόστος από την προσθήκη του επόμενου σημείο σε κάθε διαδρομή και τοποθετεί το σημείο αυτό σαν στοιχείο της λύσης στην διαδρομή με το μικρότερο κόστος , λαμβάνοντας ταυτόχρονα υπόψιν τους περιορισμούς του προβλήματός. Ο αλγόριθμος επαναλαμβάνεται μέχρι να τοποθετηθούν όλα τα σημεία.

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, καθιστός

Περιγραφή που δημιουργήθηκε αυτόματα

* self.**TestSolution** ():   
  Για κάθε route που έχει δημιουργηθεί ελέγχει ξανά τους περιορισμούς του προβλήματος και μας εμφανίζει στο τέλος εάν η λύση μας είναι αποδεκτή.

Εικόνα που περιέχει κείμενο

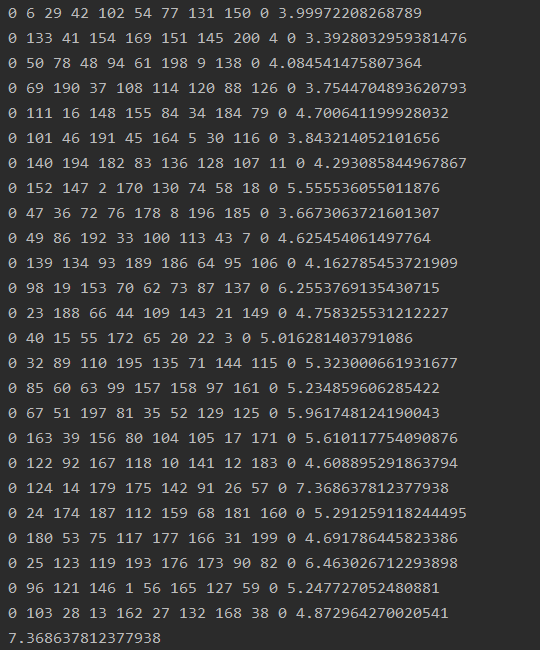
Περιγραφή που δημιουργήθηκε αυτόματα

* Self.**ReportSolution** ():   
  Παίρνει σαν input την λύση που δημιουργήσαμε και επιστρέφει μια λίστα με 25 διαδρομές και το συνολικό κόστος της κάθε διαδρομής, καθώς και το κόστος της μέγιστης διαδρομής από άποψη χρόνου, που είναι η τιμή που μας ενδιαφέρει να ελαχιστοποιήσουμε.

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Το αποτέλεσμα της αντικειμενικής μας συνάρτησης είναι 7.36, το οποίο δείχνει τις ώρες που θα χρειαστούν για να εξυπηρετηθεί το σημείο με την μεγαλύτερη διαδρομή.



***4. Σχεδιάστε ένα αλγόριθμο τοπικής έρευνας ο οποίος να χρησιμοποιεί την κίνηση μετακίνηση ενός πελάτη (relocation)***

***a. Σε πόσες επαναλήψεις η μέθοδός σας παγιδεύεται σε τοπικό ελάχιστο;***

***b. Ποιο είναι η είναι το κόστος του τοπικού ελάχιστου και ποιο το σύνολο των διαδρομών;***

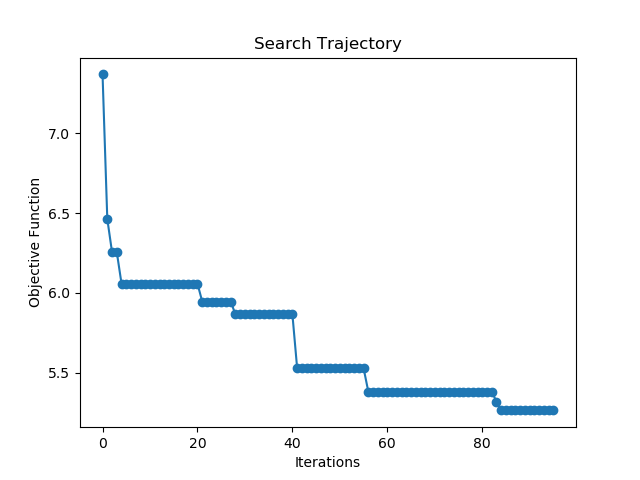
Ο αλγόριθμος τοπικής έρευνας δέχεται σαν input μια λύση του προβλήματος και κάνοντας αλλαγές μεταξύ των σημείων των διαδρομών προσπαθεί να μειώσει τον χρόνο της.

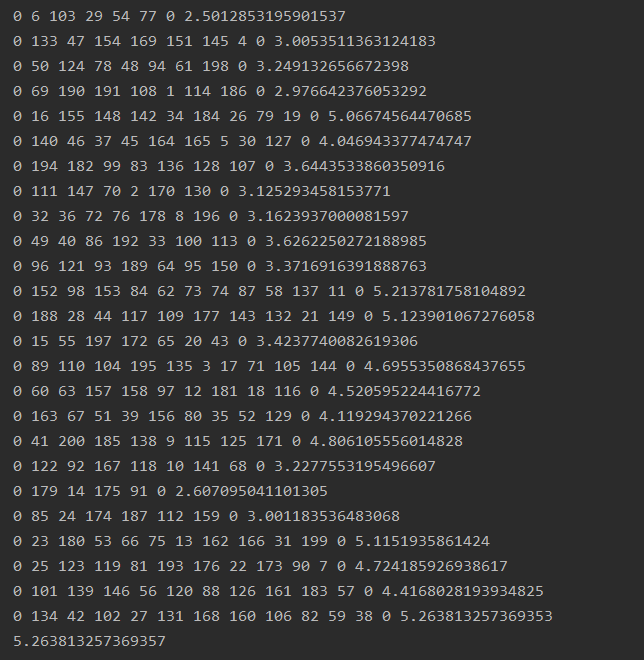
Πιο συγκεκριμένα, με το function **FindBestRelocationMove**, εντοπίζει την καλύτερη κίνηση που μπορεί να κάνει βάσει του συνολικού κόστους που θα προσφέρει η κίνηση του ενός σημείου σε μια άλλη διαδρομή ή στην ίδια.

Εφόσον το κόστος κίνησης αυτό είναι το βέλτιστο και με την προϋπόθεση ότι δεν αυξάνει την τιμή της αντικειμενικής συνάρτησης, τότε αυτή η κίνηση αποθηκεύεται προσωρινά με το function **StoreBestRelocationMove**.

Τέλος, με την συνάρτηση **ApplyRelocationMove**, ελέγχει εάν το συνολικό κόστος κίνησης είναι αρνητικό και τότε η αποθηκευμένη κίνηση εφαρμόζεται στην λύση μας αλλάζοντας την θέση των σημείων, τις διαδρομές, το κόστος των διαδρομών και το κόστος της αντικειμενικής συνάρτησης.

Παρατηρώντας το παρακάτω διάγραμμα “Search Trajectory”, από τις 7 ώρες και 36 λεπτά που διαρκούσε η μεγαλύτερη διαδρομή με την αρχική μας λύση , φαίνεται ότι με τον local search έχουμε φτάσει στις 5 ώρες και 26 λεπτά (κόστος) μετά από 83 επαναλήψεις και πραγματοποιώντας 25 διαδρομές. Σε αυτό το σημείο η λύση μας παγιδεύεται σε τοπικό ελάχιστο.





***5. Σχεδιάστε ένα VND αλγόριθμο ο οποίος θα χρησιμοποιεί τρεις τελεστές τοπικής έρευνας***

***(1)***