Report for ΜΕΒΕΔΕ

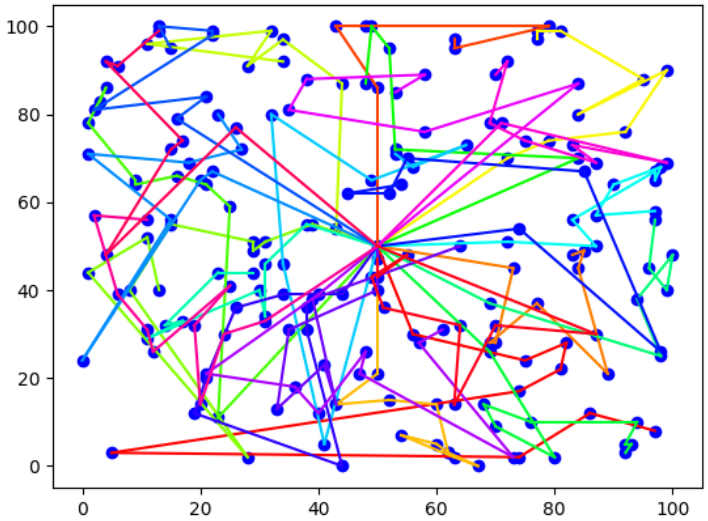
Ομάδα 16
Αθηνά Δάβαρη 8180020
Ναταλία Κατσιάπη 8180040
Αθηνά Πανταζοπούλου 8180089
Αγγελική Παπακωνσταντίνου 8180097

Για την υλοποίηση του παρόντος προβλήματος αρχικά, κατασκευάσαμε το μοντέλο μας με τον κόμβο αρχής και τους κόμβους. Εν συνεχεία, προχωρήσαμε στην υλοποίηση του στοχαστικού greedy αλγόριθμου (minimuminsertions) τον οποίο προσαρμόσαμε κατάλληλα για να υλοποιεί την επίλυση του δικού μας προβλήματος, ώστε να χρησιμοποιούνται εξ αρχής οι 25 διαδρομές. Επιπλέον, τροποποιήσαμε την διαδικασία της μεθόδου έτσι ώστε να επιλέγει κάθε φορά την μικρότερη διαδρομή για να προσθέσει κάθε κόμβο. Μετά το πέρας του Greedy αλγορίθμου, η λύση που προκύπτει με seed=109 και rcl_size=5 είναι 8.68333.

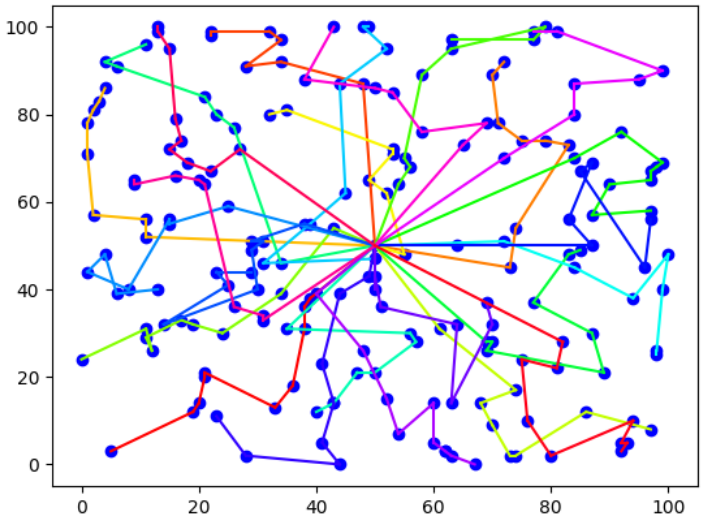
Κατόπιν γίνεται η προσθήκη ενός πλασματικού κόμβου στο τέλος κάθε διαδρομής ο οποίος δεν έχει αντίκτυπο στο χρόνο των διαδρομών.

Εν συνεχεία υλοποιείται η μέθοδος VND() η οποία εναλλάσσει την χρήση των κινήσεων Swap Move, Relocation Move, TwoOpt Move, με σκοπό να βελτιωθεί η λύση του greedy αλγορίθμου. Μετά την υλοποίηση της μεθόδου VND() η λύση διαμορφώνεται στις 4.557142857142857 ώρες.

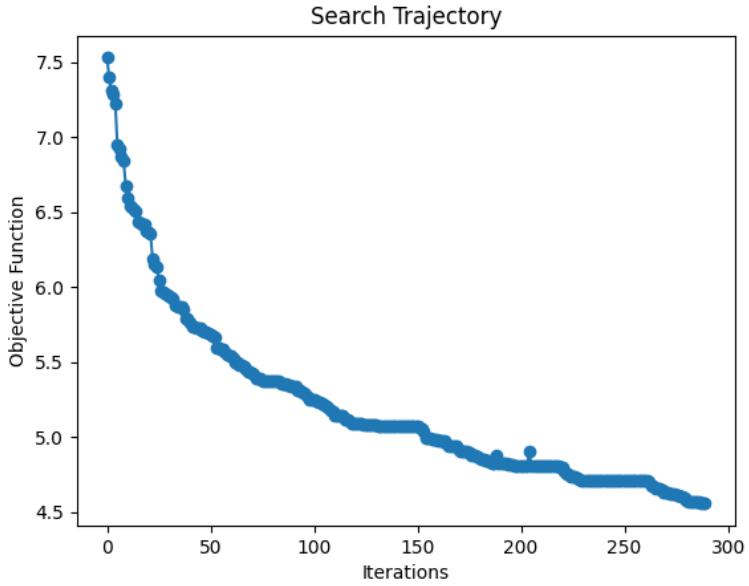
Επιπλέον, κάνουμε χρήση της λογικής απαγορευμένης έρευνας για το Relocation Move με tenure=20, προκειμένου να αποφευχθεί ατέρμων βρόχος.



Εικόνα 1 Λύση greedy αλγορίθμου.



Εικόνα 2 Λύση VND



Εικόνα 3 Μεταβολή λύσης μετά από κάθε επανάληψη του VND

Αλλαγές επίσης έγιναν στις παρακάτω μεθόδους:

Swap Move:

Με αυτή την κίνηση εντοπίζουμε σε κάθε επανάληψη δύο κόμβους τους οποίους μπορούμε να ανταλλάξουμε τις θέσεις τους στην ίδια αλλά και σε διαφορετική διαδρομή και με την προϋπόθεση ότι η αλλαγή αυτή βελτιώνει την τελική λύση. Για κάθε πιθανή κίνηση δημιουργεί ένα αντίγραφο της λύσης (newsol) και την εφαρμόζει σε αυτή και ύστερα την συγκρίνει με την ήδη υπάρχουσα λύση που έχουμε (oldsol).

Relocation Move:

Με αυτή την κίνηση εντοπίζουμε σε κάθε επανάληψη έναν κόμβο ο οποίος συμφέρει να μετακινηθεί είτε σε κάποιο άλλο σημείο στη διαδρομή στην οποία βρίσκεται είτε σε κάποια άλλη διαδρομή. Γίνεται προσπέλαση όλων των διαδρομών και κόμβων, εκτός του τελευταίου (πλασματικός κόμβος), και υπολογίζεται για κάθε συνδυασμό (διαδρομή προέλευσης - διαδρομή τοποθέτησης – θέση προέλευσης – θέση τοποθέτησης). Κατόπιν υπολογίζονται δύο κόστη (moveCosts):

- Ο χρόνος κατά τον οποίο μειώνεται η μέγιστη σε χρόνο διαδρομή, ο οποίος μπορεί είτε να είναι 0 αν η συγκεκριμένη αλλαγή δεν την επηρεάζει είτε < 0 εάν μειώνει το χρόνο της.
- Το χρονικό όφελος που προκύπτει από τις κινήσεις των οποίων η διαδρομή προέλευσης δεν είναι η κρίσιμη.

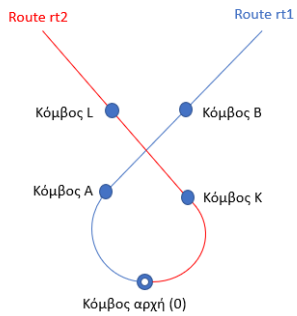
Ύστερα, αποθηκεύονται στο αντικείμενο της κλάσης RelocationMove με τη χρήση των μεθόδων StoreBestMaxRelocationMove() και StoreBestRelocationMove() το moveCostForMax εάν είναι < 0, διαφορετικά το moveCost αν και αυτό είναι < 0 στα αντίστοιχα πεδία. Όταν ολοκληρωθούν οι επαναλήψεις, η ροή επιστρέφει στο VND(), όπου ελέγχεται εάν προέκυψε κάποια λύση που είτε θα μειώσει τη μεγαλύτερη διαδρομή, είτε θα συμβάλει στην καλύτερη κατανομή των σημείων των υπολοίπων διαδρομών και την εφαρμόζει (ApplyRelocationMove()). Στη συγκεκριμένη μέθοδο ελέγχεται εάν θα πρέπει να χρησιμοποιηθούν οι τιμές που έχουν αποθηκευτεί στο αντικείμενο rm που αφορούν τη μέγιστη διαδρομή ή τις υπόλοιπες και ενημερώνει κατάλληλα τη λύση. Εάν η μέθοδος FindBestRelocationMove() δεν καταφέρει να βρει κάποια κίνηση που να βελτιώσει τη λύση, προχωρά στον επόμενο τύπο κίνησης.

TwoOpt Move

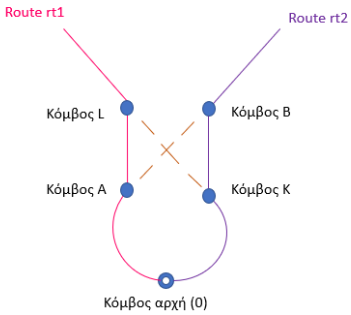
Η βασική ιδέα πίσω από αυτό τον τύπο κίνησης είναι να ακολουθήσουμε δύο διαδρομές οι οποίες διασταυρώνονται και να τις διατάξουμε με τέτοιο τρόπο ώστε να μην πέφτει η μία στην άλλη. Η κίνηση αυτή εφαρμόζεται και για μία διαδρομή όπου μπλέκεται κάποιο σημείο της και δημιουργεί κόμπο.

Στην συγκεκριμένη εφαρμογή αυτού του τύπου κίνησης 2-opt σε κάθε επανάληψη εντοπίζουμε τέσσερεις κόμβους τους οποίους μπορούμε να ανταλλάξουμε ανά δυο τις συνδέσεις τους. Οι κόμβοι αυτοί μπορεί να ανήκουν στην ίδια αλλά και σε διαφορετική διαδρομή όπως φαίνεται στα παρακάτω διαγράμματα. Στις πιθανές λύσεις που παράγονται από κόμβους στην ίδια διαδρομή ο αλγόριθμος κάνει και κάνει reverse μέρος της διαδρομής ώστε να υπολογίσει την σωστή λύση. Απαραίτητη προϋπόθεση για να εφαρμοστεί αυτή η κίνηση είναι ότι η αλλαγή αυτή βελτιώνει την τελική λύση. Για κάθε πιθανή κίνηση δημιουργεί ένα αντίγραφο της λύσης (newsol) και την εφαρμόζει σε αυτή και ύστερα την συγκρίνει με την ήδη υπάρχουσα λύση που έχουμε (oldsol).

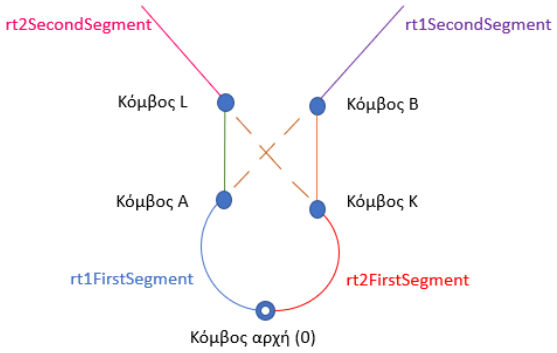
Διαφορετικές διαδρομές:



Εικόνα 4 Οι διαδρομές πριν

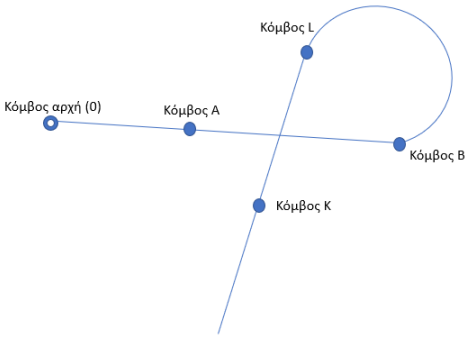


Εικόνα 5 Οι διαδρομές μετά

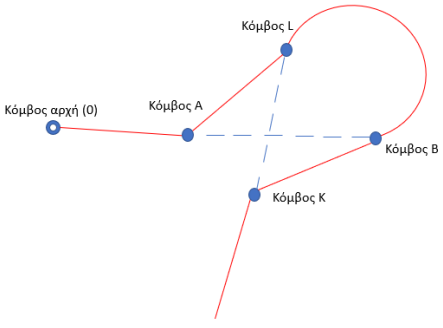


Εικόνα 6 Υπολογισμός χρόνων

Ίδια διαδρομή:



Εικόνα 7 Η διαδρομή πριν



Εικόνα 8 Η διαδρομή μετά

Τέλος, την παραπάνω λύση που βρήκαμε την ελέγξαμε με την χρήση του αρχείου Validation.py.