Μάθημα: ΜΕΒΕΔΕ

Διαγωνιστική Ομαδική Εργασία: 3-5 Άτομα

Βαθμολογία: 50% Τελικού Βαθμού

Διορία: 07/01/2019 - 15.00

Στα πλαίσια της άσκησης θα επιλύσουμε ένα πρόβλημα δρομολόγησης οχημάτων, το οποίο σχετίζεται με την παροχή βοήθειας-υλικών σε διάφορα γεωγραφικά σημεία μετά από φυσικές καταστροφές.

Το επιχειρησιακό σενάριο έχει ως εξής:

1. Μετά από μία μεγάλη φυσική καταστροφή, καλούμαστε να μεταφέρουμε όσο το δυνατό γρηγορότερα, απαραίτητες προμήθειες σε ένα σύνολο 200 γεωγραφικών σημείων.
2. Κάθε γεωγραφικό σημείο καλύπτει διαφορετικά τμήματα του πληθυσμού, έτσι ώστε κάθε ένα από αυτά να έχει διαφορετικές ανάγκες προμηθειών
3. Για τη μεταφορά των προμηθειών χρησιμοποιούνται 25 ομογενή φορτηγά αυτοκίνητα με μέγιστο φορτίο 3 tn.
4. Κάθε φορτηγό ξεκινά τη διαδρομή του από τη κεντρική αποθήκη $d = \{0\}$ και επισκέπτεται διαδοχικά κάποια από τα 200 σημεία εξυπηρέτησης $N = \{1,2,3, \dots, n\}$
5. Οι διαδρομές όλων των φορτηγών ξεκινούν ταυτόχρονα
6. Κάθε σημείο εξυπηρέτησης ικανοποιείται από μία μόνο επίσκεψη ενός αποκλειστικά οχήματος. Επομένως, όταν ένα όχημα επισκέπτεται ένα σημείο εξυπηρέτησης, παραδίδει σε αυτό το σύνολο των απαιτούμενων προμηθειών.
7. Τα οχήματα ταξιδεύουν στο οδικό δίκτυο με περίπου με 35 km/hr
8. Σε κάθε σημείο εξυπηρέτησης $i \in N$, απαιτείται ένα χρονικό διάστημα 15 λεπτών για την εκφόρτωση των προμηθειών

Ο σχεδιασμός των διαδρομών πρέπει να ελαχιστοποιεί το χρόνο ολοκλήρωσης της παραλαβής των προϊόντων από το σημείο το οποίο θα εξυπηρετηθεί πιο αργά από όλα τα υπόλοιπα σημεία.

Οι κόμβοι του προβλήματος δημιουργούνται από τον παρακάτω κώδικα ο οποίος πρέπει να ενσωματωθεί στο πρόγραμμά σας. Οι συντεταγμένες δίνονται σε χιλιόμετρα. Ο πίνακας των αποστάσεων προκύπτει από τον υπολογισμό των Ευκλείδειων αποστάσεων μεταξύ των διάφορων κόμβων:

```
public void CreateAllNodesAndServicePointLists() {
      //Create the list with the service points
      servicePoints = new ArrayList();
      Random ran = new Random(1);
      for (int i = 0 ; i < 200; i++)
      {
         Node sp = new Node();
         sp.x = ran.nextInt(100);
         sp.y = ran.nextInt(100);
         sp.demand = 100*(1 + ran.nextInt(5));
         sp.serviceTime = 0.25;
         sp.add(cust);
      }
      //Build the allNodes array and the corresponding distance matrix
      allNodes = new ArrayList();
      depot = new Node();
      depot.x = 50;
      depot.y = 50;
      depot.demand = 0;
      allNodes.add(depot);
      for (int i = 0 ; i < servicePoints.size(); i++)
      {
         Node cust = servicePoints.get(i);
         allNodes.add(cust);
      }

      for (int i = 0 ; i < allNodes.size(); i++)
      {
         Node nd = allNodes.get(i);
         nd.ID = i;
      }
   }
```

Η παραγωγή μίας εφικτής λύσης μέσω ενός πλεονεκτικού αλγορίθμου βαθμολογείται με 6.

Κατόπιν καλείστε να υλοποιήσετε έναν αλγόριθμο για να βελτιώσει την αρχική σας λύση.

Η καλύτερη τελική λύση (προφανώς εφικτή λύση) θα βαθμολογηθεί με 10.
Η χειρότερη τελική λύση (προφανώς εφικτή και βελτιωμένη σε σχέση με την αρχική λύση) θα βαθμολογηθεί με 8.

Οι βαθμοί θα προκύψουν κατόπιν γραμμικά ανάλογα με την ποιότητα της λύσης.