ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

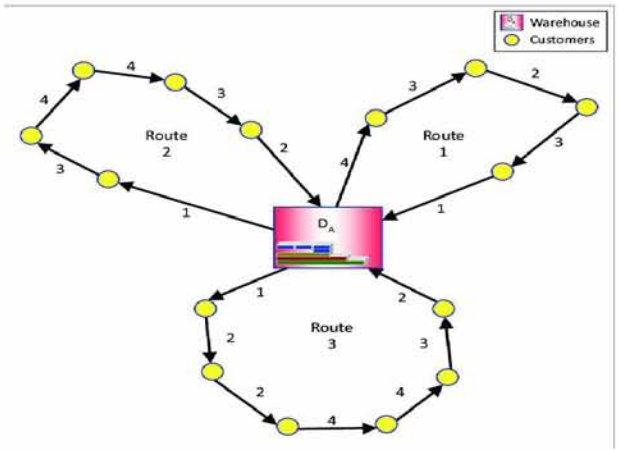
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πρόβλημα Δρομολόγησης οχημάτων περιορισμένης χωρητικότητας με ετερογενή στόλο (Capacitated Vehicle Routing Problem with Heterogeneous fleet)

Υπό

ΜΠΕΛΛΟΣ ΔΗΜΗΤΡΙΟΣ



Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού 2019

Εγκρίθηκε από τα Μέλη της Τριμελούς επιτροπής:

Πρώτος Εξεταστής Δρ. Σαχαρίδης Γεώργιος

(Επιβλέπων) Επίκουρος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών,

Πανεπιστήμιο Θεσσαλίας

Δεύτερος Εξεταστής Δρ. Δημήτριος Παντελής

Αναπληρωτής Καθηγητής

Τμήμα Μηχανολόγων Μηχανικών

Πανεπιστήμιο Θεσσαλίας

Τρίτος Εξεταστής Δρ Γεώργιος Λυμπερόπουλος

Καθηγητής Τμήμα Μηχανολόγων Μηχανικών

Πανεπιστήμιο Θεσσαλίας

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με το πρόβλημα της δρομολόγησης οχημάτων (VRP – Vehicle Routing Problem), και συγκεκριμένα με της περιορισμένης χωρητικότητας οχημάτων που αποτελείται από ετερογενή στόλο (Capacitated Vehicle Routing Problem with heterogenous fleet), αποτελώντας ένα από τα σημαντικότερα προβλήματα της δραστηριότητας «Διακίνηση Προϊόντων» ενός συστήματος Logistics.

Στόχος της εργασίας είναι η μείωση του χρόνου εξυπηρέτησης από την αποθήκη (depot) προς τους πελάτες (customers) από έναν ετερογενή στόλο φορτηγών.

Με τη βοήθεια του λογισμικού βελτιστοποίησης CPLEX και της γλώσσας προγραμματισμού C++, δημιουργήθηκε ένας κώδικας ο οποίος με την κατάλληλη αντικειμενική συνάρτηση καθώς και τους κατάλληλους περιορισμούς όπως επίσης και τα στοιχεία των πελατών και των φορτηγών από αρχεία Excel, υπολογίζονται τα δρομολόγια που θα πρέπει να πραγματοποιήσουν τα φορτηγά για την διεκπεραίωση των παραδόσεων των αποθεμάτων προς τους εκάστοτε πελάτες. Λόγω του μεγάλου όγκου πελατών δημιουργήθηκε ένας αλγόριθμος ομαδοποίησης τους (clustering) ώστε να υπάρχει μια ομαλή εκτέλεση του κώδικα καθώς και των αποτελεσμάτων.

Η διπλωματική εργασία αποτελείται από συνολικά πέντε κεφάλαια. Στο πρώτο και εισαγωγικό κεφάλαιο, δίνονται οι βασικές εκδοχές της δρομολόγησης οχημάτων, όπως επίσης και που εμφανίζονται σε καθημερινή βάση και που τα συναντάμε. Στο δεύτερο κεφάλαιο αναλύεται εκτενώς το πρόβλημα (Capacitated Vehicle Routing Problem with heterogenous fleet) έχοντας γνωστά τα πλήρη στοιχεία των πελατών αλλά και την ιδιαιτερότητα του ετερογενή στόλου..

Στο τρίτο κεφάλαιο γίνεται η παρουσίαση του μαθηματικού μοντέλου όπως αυτό περιλαμβάνει την αντικειμενική συνάρτηση αλλά και τους περιορισμούς, παράλληλα με την ενδελεχή αναφορά στο τι αυτοί σημαίνουν και ποια είναι η χρήση τους στο πρόβλημα.

Το 4ο κεφάλαιο περιέχει πληροφορίες για την μέθοδο της ομαδοποίησης των πελατών μέσα από τον κατάλληλα διαμορφωμένο αλγόριθμο του clustering, καθώς πως αυτή η μέθοδος με την σειρά της εφαρμόζεται πάνω στο πρόβλημα.

Τέλος, στο 5ο κεφάλαιο παρουσιάζεται η εφαρμογή του μοντέλου μαζί με την μέθοδο της ομαδοποίησης και η αναλυτική επίλυση του προβλήματος δρομολόγησης οχημάτων ετερογενούς στόλου. Όπως επίσης στο τέλος παρατίθεται ο κώδικας του CVRPHF μαζί με τον αλγόριθμο του clustering.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα πρώτα απ όλα να ευχαριστήσω τον επίκουρο καθηγητή κ. Σαχαρίδη Γεώργιο για την επίβλεψη και την καθοδήγηση του τόσο κατά την ανάπτυξη των ιδεών του, όσο και στην συγγραφή της παρούσας διπλωματικής εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Φραγκογιό Αντώνιο για την υπερπολύτιμη βοήθεια του στην ολοκλήρωση του κώδικα VRP καθώς και στον αλγόριθμο της ομαδοποίησης που στηρίχτηκε η διπλωματική μου εργασία.

Επιπροσθέτως θα ήθελα να ευχαριστήσω την οικογένεια μου που στάθηκε στο πλευρό μου για την κατανόηση τους ,την αγάπη τους καθώς και τη στήριξη τους προς εμένα όλα αυτά φοιτητικά μου χρόνια όπως και στους γονείς μου που δεν βρίσκονται εν ζωή. Επίσης στον θειο μου Γεώργιο Αμβράζη που με καθοδήγησε σ όλη την πορεία του λυκείου και μου κληροδότησε τις απαραίτητες γνώσεις για να φτάσω στην επιτυχία .

Πίνακας περιεχομένων

ΛΙΣΤΑ ΑΚΡΟΝΥΜΙΩΝ

1. AP : Affinity Propagation,
2. VRP : Vehicle Routing Problem,
3. TSP: Travelling Salesman Problem,
4. CVRP : Capacitated Vehicle Routing Problem,
5. CVRPHF : Capacitated Vehicle Routing Problem Heterogeneous Fleet,
6. HFVRP : Heterogeneous fleet vehicle routing problem,
7. TW : time windows,
8. SVRP : Stochastic vehicle routing problem.
9. PVRP : Periodic Vehicle Routing Problem

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

1ο ΚΕΦΑΛΑΙΟ

1.1 Ιστορική αναδρομή

Η θεωρητική έρευνα και οι πρακτικές εφαρμογές στον τομέα της δρομολόγησης των οχημάτων ξεκίνησε το 1959 με το «πρόβλημα αποστολής φορτηγού» (truck dispatching problem) που τέθηκε από τους Dantzig και Ramser ως εξής: Να βρεθεί «… η βέλτιστη δρομολόγηση ενός στόλου βενζινοκίνητων οχημάτων μεταξύ ενός τελικού σταθμού και ενός μεγάλου αριθμού σταθμών, εξυπηρετούμενων από τον τελικό σταθμό" . Κάνοντας χρήση μιας μεθόδου που βασίζεται στην μοντελοποίηση γραμμικού προγραμματισμού, οι υπολογισμοί που έκαναν με το χέρι έδωσαν μια σχεδόν βέλτιστη λύση, με τέσσερα δρομολόγια για ένα πρόβλημα με δώδεκα σταθμούς που απαιτούσαν εξυπηρέτηση. Οι ερευνητές είχαν δηλώσει πως μέχρι εκείνη την στιγμή δεν είχαν βρεθεί κάποιες πρακτικές εφαρμογές για τη μέθοδο.

Στα επόμενα χρόνια, πολλοί ερευνητές ασχολήθηκαν με το συγκεκριμένο πρόβλημα, με παραλλαγές του αλλά και με εναλλακτικές μεθόδους επίλυσης. Σήμερα, υπάρχει ένας πάρα πολύ μεγάλος όγκος εργασιών που μπορεί να βρει κανείς, που αφορούν στο συγκεκριμένο αντικείμενο και οι έρευνες για καλύτερες μεθόδους επίλυσης, για την αντιμετώπιση μεγάλων προβλημάτων, όπως αυτά που συναντώνται στις πρακτικές εφαρμογές, συνεχίζονται.

1.2 ΔΡΟΜΟΛΟΓΗΣΗ ΟΧΗΜΑΤΩΝ

Το V.R.P. είναι ουσιαστικά το πρόβλημα της εύρεσης των βέλτιστων διαδρομών, ενός στόλου οχημάτων, τα οποία θα πρέπει να εξυπηρετήσουν ένα σύνολο κόμβων, έχοντας ως σημείο εκκίνησης και τερματισμού έναν συγκεκριμένο κόμβο-σταθμό (depot).Ο στόχος του προβλήματος είναι η ελαχιστοποίηση του κόστους μεταφοράς καθώς και την ελαχιστοποίηση του χρόνου ταξιδίου.

Το κόστος μεταφοράς αποτελεί σημαντικό μέρος του συνολικού κόστους ενός τελικού προϊόντος, της τάξης του 10%. Κατά συνέπεια, η ελαχιστοποίηση του θα απέφερε ένα σημαντικό οικονομικό όφελος σε επιχειρήσεις και οργανισμούς. Επιπλέον, το κόστος μεταφοράς είναι άρρηκτα συνδεδεμένο με την κατανάλωση ενέργειας, και καθώς το συντριπτικά μεγαλύτερο μέρος των στόλων μεταφοράς σήμερα αποτελείται από οχήματα που καταναλώνουν ορυκτά καύσιμα, η βελτιστοποίηση των διαδρομών των

οχημάτων αυτών, θα μπορούσε να αποφέρει σημαντικά οφέλη ως προς την προστασία του περιβάλλοντος.

Από την εμφάνιση του αρχικού προβλήματος, έχουν αναπτυχθεί πολλές γενικεύσεις του ώστε να μπορούν να μοντελοποιηθούν διαδικασίες του πραγματικού κόσμου.

Οι γενικεύσεις μπορούν να χωριστούν σε κατηγορίες, ανάλογα με το στοιχείο του προβλήματος που γενικεύεται σε κάθε περίπτωση. Υπάρχουν πάρα πολλές τέτοιες γενικεύσεις στη βιβλιογραφία. Κάποιες απ᾽ αυτές παρουσιάζονται παρακάτω.

Όσον αφορά τις χωρητικότητες και το είδος του φορτίου που μπορεί να μεταφέρει το κάθε όχημα , η βασικότερη γενίκευση είναι το πρόβλημα δρομολόγησης οχημάτων περιορισμένης χωρητικότητας (Capacitated Vehicle Routing Problem or CVRP) , όπου τα οχήματα του στόλου έχουν πεπερασμένη χωρητικότητα. Αυτή η γενίκευση είναι αναγκαία, καθώς κάθε πραγματικό όχημα μπορεί να μεταφέρει πεπερασμένη ποσότητα προϊόντων. Μία ακόμα σημαντική περίπτωση είναι αυτή του ανομοιογενούς στόλου οχημάτων (heterogeneous fleet), όπου τα οχήματα που απαρτίζουν τον στόλο έχουν διαφορετικές χωρητικότητες ή μπορούν να μεταφέρουν, το καθένα, συγκεκριμένα είδη προϊόντων και όχι απαραίτητα οποιοδήποτε προϊόν.

Όσον αφορά τις ζητήσεις των κόμβων-πελατών, αυτές μπορεί να είναι ντετερμινιστικές (καθορισμένες) ή στοχαστικές (τυχαίες), με την έννοια του να μην είναι γνωστές κατά την περίοδο του προγραμματισμού (stochastic demand). Επιπλέον οι κόμβοι –πελάτες μπορεί να ζητούν να παραλάβουν αλλά και να παραδώσουν κάποια προϊόντα (pick-up and delivery), αντί να συμβαίνει μόνο το ένα απ᾽ τα δύο, όπως στην ειδική περίπτωση του V.R.P.

Οι κόμβοι-πελάτες που θα πρέπει να εξυπηρετηθούν, είναι δυνατόν επίσης να μην είναι γνωστοί κατά την περίοδο του προγραμματισμού, αλλά και να προκύπτουν στοχαστικά, κατά τη διάρκεια της περιόδου εξυπηρέτησης.

Επίσης υπάρχει και η πιθανότητα όπου το σύνολο των κόμβων-σταθμών (depots), μπορεί να είναι περισσότεροι από έναν δηλαδή να υπαγόμαστε στην ειδική κατηγορία των πολλαπλών σταθμών (multi depot), όπως συμβαίνει στην ειδικότερη περίπτωση.

Τέλος όσον αφορά τον χρόνο εξυπηρέτησης ή προτιμότερα την στιγμή όπου θα πραγματοποιηθεί η εξυπηρέτηση του εκάστοτε πελάτη από τον συγκεκριμένο τύπου οχήματος , μπορεί αυτή να απαιτείται να είναι σε συγκεκριμένες χρονικές ζώνες ή στιγμές κατά τη διάρκεια της ημέρας εξυπηρέτησης τα οποία ονομάζονται χρονικά παράθυρα (time windows or TW). Επιπροσθέτως, είναι δυνατόν, οι κόμβοι-πελάτες

να απαιτείται να εξυπηρετηθούν με κάποια συγκεκριμένη περιοδικότητα κατά τη διάρκεια κάποιας χρονικής περιόδου. Διαφορετικά, η περίοδος προγραμματισμού στη προκειμένη περίπτωση, είναι μια περίοδος ημερών και όχι μία ημέρα (periodic).

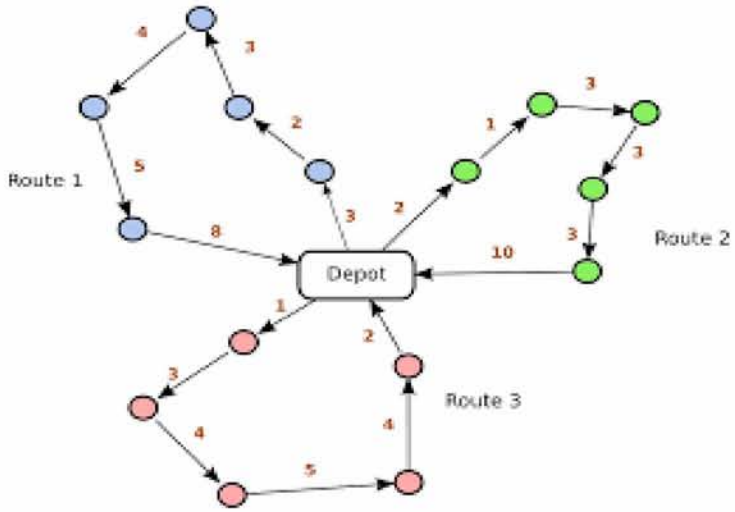


Figure 1 Γράφημα Δρομολόγησης Οχημάτων.

.

- Οι κορυφές των γραφημάτων όπου είναι οι τοποθεσίες των πελατών,
- Η Ζήτηση (demand) των πελατών στα προβλήματα διανομής και η ποσότητα συλλογής στα προβλήματα παραλαβής προϊόντων,
- Το χρονικό διάστημα της ημέρας στη διάρκεια της οποίας μπορούν να εξυπηρετηθούν οι πελάτες (T.W.),
- Ο χρόνος που απαιτείται για τη φόρτωση ή την εκφόρτωση προϊόντων από τις τοποθεσίες των πελατών (unloading - loading times),

- Το σύνολο των διαθέσιμων οχημάτων που μπορούν να εξυπηρετήσουν τους πελάτες,
- Οι διαδρομές των οχημάτων σχεδιάζονται έτσι ώστε να ελαχιστοποιείται η συνολική απόσταση που θα διανύσουν τα οχήματα.

1.3 ΠΑΡΑΛΛΑΓΕΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΥΣ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ

Λόγω των αρκετών υποθέσεων που μπορούμε να κάνουμε με το πρόβλημα δρομολόγησης οχημάτων όπως για παράδειγμα να προσθέσουμε περιορισμούς ή ακόμα και να αφαιρέσουμε έχουμε τα αποτέλεσμα των διαφόρων παραδοχών του VRP. Αυτό συμβαίνει επειδή το κλασικό πρόβλημα στερείται πρακτικής εφαρμογής στις επιχειρήσεις. Επομένως παρακάτω με βάση αυτά που προαναφέρθηκαν ορισμένες παραλλαγές του κλασσικού προβλήματος δρομολόγησης οχημάτων είναι οι εξής:

1. *Πρόβλημα του Περιπλανώμενου Πωλητή* (Travelling salesman problem),

   Το πρόβλημα του πλανόδιου πωλητή έχει να κάνει με την εύρεση της συντομότερης διαδρομής για ένα όχημα δεδομένου ότι αυτό έχει ξεκινήσει από μια και μόνο αφετηρία και μετά την εξυπηρέτηση των πελατών να επιστρέψει σ αυτήν.

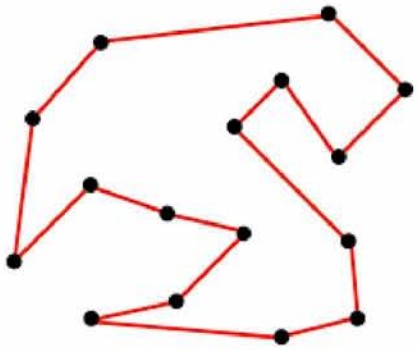


Figure 2 Επίλυση του προβλήματος του πλανόδιου πωλητή

2. *Το Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα* (Vehicle Routing Problem with Time Windows - VRPTW),

Το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα αποτελεί κλασσική παραδοχή του προβλήματος δρομολόγησης οχημάτων. Στην παραδοχή αυτή έχουμε την προσθήκη του περιορισμού ότι ο πελάτης πρέπει να εξυπηρετήσει από ένα όχημα μέσα σε ένα συγκεκριμένο χρονικό όριο {t1,t2}. Δηλαδή θα πρέπει όπως είναι λογικό η χρονική εξυπηρέτηση του πελάτη να ξεκινήσει αυστηρά την χρονική στιγμή t1 και όχι νωρίτερα και να διεκπεραιωθεί νωρίτερα ή ακριβώς την χρονική στιγμή t2. Στην αντίθετη άποψη όπου η εξυπηρέτηση του πελάτη είναι μετά το πέρας των χρονικών ορίων που έχει θεσπίσει ο εκάστοτε πελάτης τότε είναι αδύνατον να πραγματοποιηθεί η εξυπηρέτηση του και αυτή θα πραγματοποιηθεί την επομένη μέρα ή το νέο χρονικό όριο κατόπιν συνεννόηση του πελάτη με τον οδηγό του οχήματος.

3. *Δρομολόγηση Στόλου Οχημάτων με περιορισμό Χωρητικότητας* (Capacitated Vehicle Routing Problem - CVRP),

Το περιορισμένης Χωρητικότητας πρόβλημα της δρομολόγησης οχημάτων (CVRP) αποτελεί την συνέχεια του TSP καθώς και την βασική εκδοχή του VRP στις περιπτώσεις που ένα όχημα δεν είναι σε θέση να εξυπηρετήσει ένα σύνολο οχημάτων. Με αποτέλεσμα να πρέπει δημιουργηθούν περισσότερες κυκλικές διαδρομές σε σχέση με το TSP που έχει έναν κύκλο εξυπηρέτησης . Όλοι όμως με ξεκινάνε και τελειώνουν στο ίδιο σημείο-αφετηρία και όλοι οι πελάτες θα καλύπτονται από το σύνολο των κυκλικών διαδρομών.

Στο CVRP οι πελάτες οι οποίοι ανήκουν στους υπόκυκλους δέχονται την παράδοση των προϊόντων, η ζήτησή τους είναι ντετερμινιστική (καθορισμένη) και γνωστή εκ των προτέρων. Τα οχήματα είναι της ιδίας χωρητικότητας και πρέπει να εξυπηρετήσουν τις ήδη γνωστές απαιτήσεις στις ζητήσεις των πελατών από μια κοινή αποθήκη λαμβάνοντας υπόψη ότι το άθροισμα της ζήτησης των πελατών που επισκέπτονται δεν πρέπει να ξεπερνά την χωρητικότητα τους.

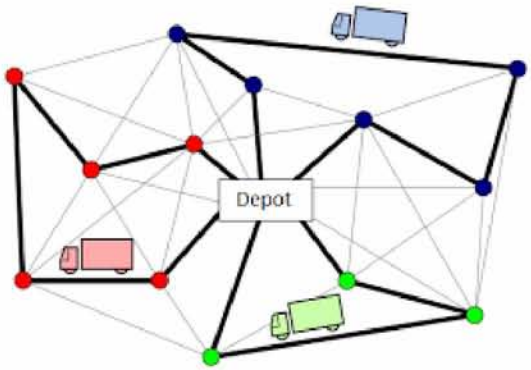


Figure 3 Capacitated Vehicle Routing Problem

4. *Ύπαρξη Πολλαπλών Αποθηκών* (Multidepot VRP),

Στην περίπτωση που υπάρχουν πολλαπλές αποθήκες αναδύονται 2 υποκατηγορίες οι οποίες είναι οι ακόλουθες:

1) Στην πρώτη υποκατηγορία έχουμε ότι κάθε μια αποθήκη περιέχει τον δικό της στόλο οχημάτων καθώς και τους δικούς της πελάτες όπου πρέπει να εξυπηρετηθούν από τα εκάστοτε οχήματα,

2) Η 2η υποκατηγορία έχει να κάνει με την περίπτωση ότι ένα όχημα μπορεί να ξεκινήσει από μια αποθήκη να εξυπηρετήσει τους πελάτες που είναι προγραμματισμένοι και στο τέλος να καταλήξει σε μια διαφορετική αποθήκη από αυτήν απ᾽ όπου ξεκίνησε όπως επίσης υπάρχει και η δυνατότητα να σταματήσει σε μια άλλη αποθήκη ώστε να φορτώσει τα προϊόντα και στην συνέχεια να τελειώνει την διαδρομή.

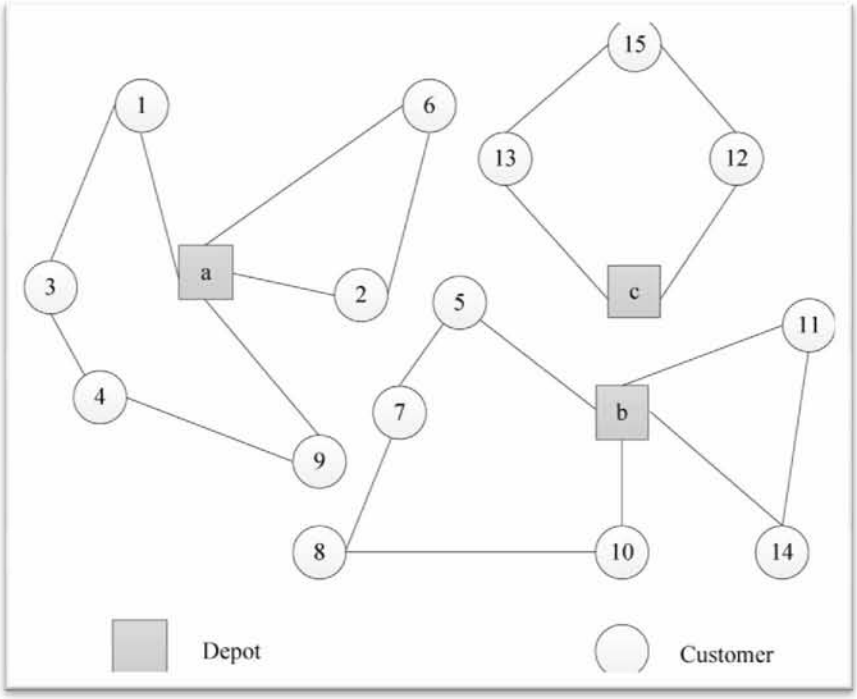


Figure 4 multi depot vehicle routing problem

5. *Πρόβλημα δρομολόγησης ετερογενών οχημάτων* (Heterogeneous fleet vehicle routing problem- HFVRP),

Στην περίπτωση του προβλήματος δρομολόγησης ετερογενών οχημάτων έχουμε να κάνουμε με έναν διαφορετικό στόλο οχημάτων όπου αυτός διαφέρει μεταξύ των φορτηγών στα εξής:

- Κάθε όχημα έχει διαφορετική χωρητικότητα,
- Το είδος κάθε φορτηγού μπορεί να είναι διαφορετικό
- Διαφορετικότητα στο είδος τα φορτίου που μεταφέρει όπως για παράδειγμα αν αυτό είναι ψυχρό-ξηρό φορτίο,
- Έχει καθορισμένο σταθερό αλλά και μεταβλητό κόστος.

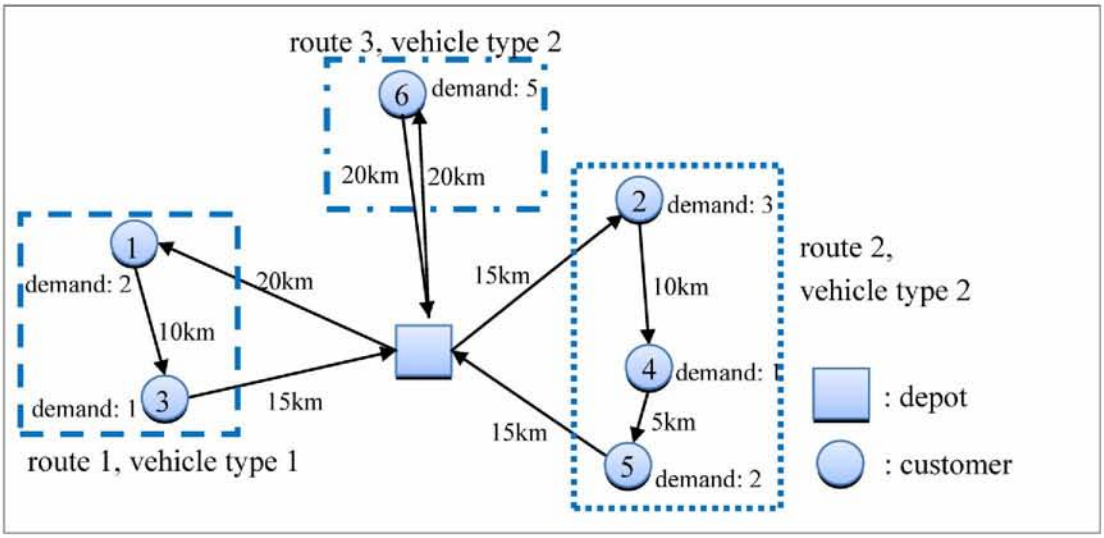


Figure 5 heterogeneous fleet vrp

6. *Πρόβλημα δρομολόγησης Οχημάτων χωρίς την επιστροφή στην αποθήκη* (Open VRP),

   Στις προηγούμενες παραδοχές παρατηρήσαμε ότι μετά το πέρας των εξυπηρετήσεων των πελατών από ένα όχημα, το τελευταίο επιστρέφει στην αποθήκη. Στην περίπτωση αυτή το φορτηγό δεν γυρίζει ποτέ στην αφετηρία, αλλά συνεχίζει να εξυπηρετεί τους πελάτες μέχρι να αδειάσει όλο το φορτίο του. Το κέρδος της περίπτωσης αυτής είναι ότι το όχημα δεν σπαταλάει τον χρόνο για να επιστρέψει στην αποθήκη αλλά τον αξιοποιεί με σκοπό να εξυπηρετήσει περισσότερους πελάτες. Με αποτέλεσμα να μειωθεί η συνολική απόσταση που θα διανυθεί με σκοπό να εξυπηρετήσουν όλοι οι πελάτες.

7. *Πρόβλημα Δρομολόγησης Οχημάτων με Διανομή και παραλαβή προϊόντων* (Pick-up and Delivery VRP),

   Η παραδοχή αυτή έχει να κάνει με διανομές (delivery) και παραλαβές προϊόντων (pick up's) από τους πελάτες κατά την διάρκεια μιας διαδρομής. Στην περίπτωση αυτή έχουμε να κάνουμε με την υπόθεση ότι ένας πελάτης θα εξυπηρετηθεί από το φορτηγό το οποίο θα ξεφορτώσει την παραγγελία καθώς και την περίπτωση όπου το φορτηγό θα παραλάβει από τον πελάτη προϊόντα ή κάποιο φορτίο .

8. *Στοχαστικό πρόβλημα δρομολόγησης οχημάτων* (Stochastic vehicle routing problem- SVRP).

Στην περίπτωση του στοχαστικού προβλήματος δρομολόγησης οχημάτων ένα ή περισσότερα στοιχεία του προβλήματος, όπως για παράδειγμα ο αριθμός των πελατών ή η ζήτησή τους, δεν θεωρούνται σταθερά, αλλά δυναμικά, δηλαδή δεν είναι γνωστά εκ των προτέρων αλλά μπορούν να αλλάξουν ανά πάσα στιγμή.

9. Πρόβλημα Δρομολόγησης Οχημάτων Περιόδου (Periodic Vehicle Routing Problem– PVRP)

Σε αντίθεση με συνηθισμένα προβλήματα VRP, όπου οι παραδόσεις ρυθμίζονται σε ημερήσιο επίπεδο, το πρόβλημα δρομολόγησης οχημάτων περιόδου είναι πιο περίπλοκο, καθώς επεκτείνει το διάστημα του προγραμματισμού σε Μ ημέρες. Στην ουσία πρόκειται για ένα Πολύ – Επίπεδο Συνδυαστικό Πρόβλημα Βελτιστοποίησης, τα επίπεδα του οποίου αναλύονται στη συνέχεια:

i. Πρώτο Επίπεδο: Στο επίπεδο αυτό καθορίζεται η περίοδος σύμφωνα με την οποία πραγματοποιούνται οι διανομές για κάθε πελάτη, ήτοι κάθε Χ ημέρες.
ii. Δεύτερο Επίπεδο: Στο εν λόγω επίπεδο πραγματοποιείται επιλογή από τις διάφορες εναλλακτικές για κάθε πελάτη, τηρώντας όμως τους περιορισμούς που έχουν τεθεί για τις διανομές. Συνεπώς, πραγματοποιείται ομαδοποίηση των πελατών που θα πρέπει να επισκεφτεί το όχημα την ίδια ημέρα της περιόδου
iii. Τρίτο Επίπεδο: Σε αυτό το επίπεδο το πρόβλημα επιλύεται ως απλό πρόβλημα VRP για κάθε μία ημέρα της υπό εξέταση περιόδου.

Συνοψίζοντας θα ήθελα να επισημάνω ότι είναι δυνατόν στην πραγματική ζωή πολλά από τα προβλήματα που προαναφέρθηκαν να αποτελούν συνδυασμό αυτών των παραλλαγών, όπως για παράδειγμα πρόβλημα δρομολόγησης οχημάτων περιορισμένης χωρητικότητας με ετερογενή στόλο με χρονικά παράθυρα, πρόβλημα δρομολόγησης οχημάτων με πολλαπλές αποθήκες με χρονικά παράθυρα κ.α. .

1.4 Καθημερινή χρήση VRP

Το VRP έχει ευρεία χρήση πάνω σε εφαρμογές της καθημερινότητας σε όλο τον κόσμο. Οι βασικοί κλάδοι που βρίσκουν εφαρμογή τα προβλήματα δρομολόγησης οχημάτων (VRP) της Καθημερινούς ζωής είναι οι εξής:

1. Διανομή τροφίμων ,

Η διανομή τροφίμων έχει τα δικά της χαρακτηριστικά, περιορισμούς και προκλήσεις, όπως η ποιότητα των προϊόντων, η υγεία και η ασφάλεια. Τα προϊόντα έχουν συχνά ένα περιορισμένη ζωή, έτσι ώστε οι εργασίες διανομής πρέπει να λαμβάνουν υπόψη τη θερμοκρασία, την υγρασία και το χρόνο – σε-μεταβατικές σκέψεις, καθώς και πολλούς άλλους περιορισμούς που σχετίζονται με τα προϊόντα. Χρήση τους π.χ. είναι διανομή γαλακτοκομικών προϊόντων σε μάρκετ, παντοπωλεία σπίτια κα. Η διανομή αναψυκτικών και αλκοολούχων πότων όπως και επίσης και οι μεταφορές βρωσίμων φαγητών κτλ.

2. Διαχείριση αποθεμάτων,

Στην περίπτωση της διαχείρισης αποθεμάτων έχουμε να κάνουμε ότι οι πελάτες κάνουν διαθέσιμες τις πληροφορίες του αποθέματος στους διανομείς οι οποίοι έπειτα αναλαμβάνουν την ευθύνη της απόφασης να εξυπηρετήσουν και ποιους πελάτες. Συνεπώς ο προμηθευτής έχει να επιλέξει πόσο συχνά πότε και σε ποιες ποσότητες θα εξυπηρετηθούν οι πελάτες. Αυτή η ολοκληρωμένη διαχείριση των αποθεμάτων και διανομής προσφέρει μεγαλύτερη ευελιξία και ευκολία στον σχεδιασμό αποτελεσματικότερων διαδρόμων για τα οχήματα, ενώ παράλληλα πραγματοποιείται βελτιστοποίηση στην αποθήκη κα όλη την διάρκεια της αλυσίδας ανεφοδιασμού.

3 Συλλογή και διαχείριση απορριμμάτων,

Ως απορρίμματα ή απόβλητα ορίζονται τα υπολείμματα και τα απορριφθέντα αντικείμενα όταν έχουν σταματήσει να εξυπηρετούν τον αρχικό σκοπό τους για τον οποίο έχουν κατασκευαστεί. Για τον λόγο αυτό η βελτιστοποίηση των δρομολογίων των απορριμματοφόρων αποκομιδής είναι ένας πολύ σημαντικός παράγοντας για την αποδοτικότερη και οικονομικότερη

αποκομιδή των αποβλήτων. Επομένως το πρόβλημα δρομολόγησης οχημάτων (VRP) πάνω στην περισυλλογή των απορριμμάτων/αποβλήτων έχει πολύ σημαντικό χαρακτήρα. Καθώς το μοντέλο δημιουργεί τις κατάλληλες διαδρομές οι οποίες έχοντας το λιγότερο δυνατό κόστος με τις λιγότερες δυνατές αποστάσεις να διανύονται λαμβάνοντας υπόψη την πυκνότητα του πληθυσμού, την χωρητικότητα των φορτηγών, το οδικό δίκτυο αλλά και τους κάδους απορριμμάτων.

4 Αποστολή αλληλογραφίας και δεμάτων.

Το ταχυδρομείο και η παράδοση δεμάτων είναι μια πολύ σημαντική βιομηχανία. Σε αυτή την ενότητα, τα σχόλια των εφαρμογών πραγματικής ζωής κυμαίνονται από την παράδοση αλληλογραφίας μέχρι την παράδοση των παραγγελιών του Internet, αγγίζοντας πολλές παραλλαγές του κλασικού VRP, όπως εκείνες που αφορούν τα χρονικά παράθυρα και τα pick-up and deliveries.

2^{0} ΚΕΦΑΛΑΙΟ

2.1 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Capacitated VRP with heterogeneous fleet (C.V.R.P.H.F.)

Το πρόβλημα που μελετήθηκε και παρουσιάζεται στη συγκεκριμένη διπλωματική εργασία αποτελεί μια πολύ γνωστή παραλλαγή του προβλήματος δρομολόγησης οχημάτων . Είναι το πρόβλημα δρομολόγησης οχημάτων που σχεδιάζει και δημιουργεί τις βέλτιστες διαδρομές που θα χρησιμοποιηθούν από έναν στόλο οχημάτων για να εξυπηρετήσουν ένα σύνολο πελατών με τον περιορισμό της χωρητικότητας του οχήματος. Ο στόλος των οχημάτων βασίζεται σε μια κεντρική αποθήκη και χαρακτηρίζεται από διαφορετικές χωρητικότητες, διαφορετικά κόστη καθώς και το φορτίο που δύναται να μεταφέρουν είναι διαφορετικό και τέλος αν μπορούν να εισέρθουν στον δακτύλιο μιας πόλης.

Στην συγκεκριμένη γενίκευση δεν υπάρχει ο περιορισμός της χρονικής περιόδου (time windows) , μέσα στην οποία πρέπει να επιτευχθεί η εξυπηρέτηση των πελατών. Ωστόσο έχουμε υποθέσει για να επιτύχουμε μεγαλύτερη ρεαλιστικότητα στην προσομοίωση μας ότι οι οδηγοί των φορτηγών θα δουλεύουν το καθημερινό και νόμιμο 8ωρο, ένας ετερογενής στόλος οχημάτων με πεπερασμένη χωρητικότητα, εκτελεί δρομολόγια εκκινώντας και καταλήγοντας σε έναν συγκεκριμένο κόμβο-σταθμό (depot).

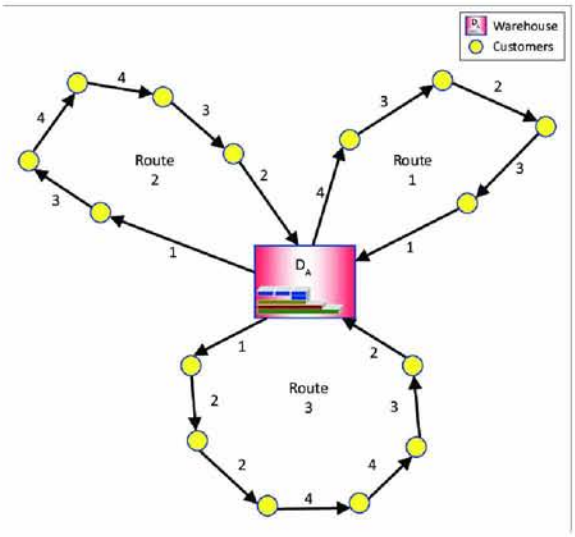


Figure 6 Capacitated VRP with heterogeneous fleet

Μία πιο λεπτομερής περιγραφή του προβλήματος είναι η εξής:

Δίνεται ένα σύνολο κόμβων-πελατών της τάξης των 171 και του κόμβου-σταθμού (depot), με τις μεταξύ τους αποστάσεις αλλά και με την αποθήκη να είναι γνωστές, όπως επίσης η ζήτηση κάθε κόμβου-πελάτη τόσο σε κιλά όσο και σε όγκο να είναι γνωστά. Ο χρόνος μεταξύ των πελατών αλλά και ο χρόνος από τον κόμβο προς τους πελάτες να είναι και αυτός γνωστός. Επιπλέον ο στόλος των οχημάτων είναι δεδομένος στα πλαίσια της διπλωματικής εργασίας και απαρτίζεται από ένα σύνολο στόλου της τάξης μεγέθους των 105 φορτηγών όπως επίσης και οι χωρητικότητες των φορτηγών ομοίως σε κιλά και σε όγκο να είναι γνωστές. Επιπροσθέτως μια ακόμα ιδιαιτερότητα που έχει το συγκεκριμένο πρόβλημα βρίσκεται στις συγκεκριμένες απαιτήσεις που έχουν οι πελάτες ως προς ποια φορτηγά θα τους εξυπηρετήσουν. Δηλαδή δεν μπορούν όλα τα φορτηγά να ικανοποιήσουν τις απαιτήσεις των πελατών καθώς όπως θα δούμε λεπτομερώς στην συνεχεία αυτοί χωρίζονται σε κατηγορίες έτσι ώστε να καθιστούν αδύνατο την εξυπηρέτηση τους από όλους τους τύπους φορτηγών που απαρτίζουν τον στόλο μας. Έτσι τα φορτηγά χωρίζονται με την σειρά τους σε κατηγορίες ώστε να μπορέσουν και αυτά με την σειρά τους να εξυπηρετήσουν τους σωστούς πελάτες και να καλυφτεί η ζήτησης τους τόσο σε κιλά αλλά όσο και σε όγκο.

2.2 ΚΑΤΗΓΟΡΙΕΣ ΠΕΛΑΤΩΝ

Μελετώνται 2 χαρακτηριστικά τα οποία έχουν οι πελάτες και καθένα από αυτά παίρνει δυαδικές τιμές (Boolean) του τύπου η πρώτη είναι η αληθής (TRUE) και η δεύτερη είναι η ψευδής (FALSE). Έτσι δημιουργούνται 4 συνδυασμοί, για παράδειγμα το 1° το χαρακτηριστικό : ένας πελάτης θέλει να του παραδοθεί ψυχρό φορτίο και ένας άλλος πελάτης θέλει να του παραδοθεί ξηρό φορτίο. 2° χαρακτηριστικό : ένας πελάτης βρίσκεται εντός του δακτυλίου μιας πόλης και ένας άλλος πελάτης βρίσκεται εκτός του δακτυλίου. Έτσι έχουμε τους παρακάτω συνδυασμούς:

1. TRUE-TRUE (T-T) : ο πελάτης θέλει ψυχρό φορτίο και βρίσκεται εντός δακτυλίου πόλης ,
2. TRUE-FALSE (T-F) : ο πελάτης θέλει ψυχρό φορτίο αλλά βρίσκεται εκτός δακτυλίου,
3. FALSE- TRUE (F-T) : ο πελάτης θέλει ξηρό φορτίο αλλά βρίσκεται εντός δακτυλίου πόλης ,
4. FALSE-FALSE (F-F) : ο πελάτης θέλει ξηρό φορτίο και βρίσκεται εκτός δακτυλίου πόλης.

Με βάση αυτούς τους τέσσερεις συνδυασμούς κατατάσσουμε όλους τους πελάτες καθώς και τον στόλο των οχημάτων μας. Αντίστοιχα με τις κατηγορίες πελατών και ο στόλος οχημάτων χωρίζεται στις παρακάτω κατηγορίες:

1. TRUE-TRUE : το φορτηγό μπορεί να φέρει ψυχρό φορτίο και να μπει στον δακτύλιο,
2. TRUE-FALSE : το φορτηγό μπορεί να μεταφέρει ψυχρό φορτίο αλλά δεν μπορεί να μπει στον δακτύλιο,
3. FALSE-TRUE : το φορτηγό μπορεί να μεταφέρει ξηρό φορτίο και μπορεί να μπει στον δακτύλιο,
4. FALSE-FALSE : το φορτηγό μπορεί να μεταφέρει ξηρό φορτίο αλλά δεν μπορεί να μπεί στον δακτύλιο.

Αναλυτικότερα έχουμε:

Όταν ο εκάστοτε πελάτης ανήκει στην κατηγορία TRUE-TRUE (T-T) δηλαδή θέλει ψυχρό φορτίο και βρίσκεται εντός δακτυλίου τότε το μόνο φορτηγό το οποίο μπορεί να τον εξυπηρετήσει είναι το όχημα T-T το οποίο μπορεί να μεταφέρει ψυχρό φορτίο και να μπει και

στον δακτύλιο. Στην περίπτωση που πελάτης είναι TRUE-FALSE (T-F) δηλαδή απαιτεί ψυχρό φορτίο αλλά βρίσκεται εκτός δακτυλίου πόλης τότε τα κατάλληλα φορτηγά που είναι σε θέση να τον εξυπηρετήσουν είναι τα TRUE-TRUE (T-T) καθώς μπορούν να μεταφέρουν ψυχρό φορτίο και εφόσον μπορούν να μπουν στον δακτύλιο μπορούν να εξυπηρετήσουν και τους πελάτες εκτός του δακτυλίου ή TRUE-FALSE (T-F) όπου τα συγκεκριμένα οχήματα φέρουν ψυχρό φορτίο και εξυπηρετούν πελάτες εκτός δακτυλίου πόλης. Εν συνεχεία στην αμέσως επόμενη κατηγορία όπου ο πελάτης ανήκει στην FALSE- TRUE (F-T) δηλαδή η απαίτηση του είναι ξηρό φορτίο και βρίσκεται εντός δακτυλίου τότε τα οχήματα θα πρέπει να ανήκουν στον συνδυασμό TRUE-TRUE (T-T) όπου τα συγκεκριμένου τύπου φορτηγά εκτός από το ψυχρό φορτίο μπορούν να μεταφέρουν και ξηρό φορτίο (κλείνοντας το ψυγείο της καρότσας) και μπορούν να εξυπηρετήσουν πελάτες εντος δακτυλίου και επίσης έχουμε τα FALSE- TRUE (F-T)τα οποία με την σειρά τους μεταφέρουν ξηρό φορτίο και εξυπηρετούν πελάτες εκτός δακτυλίου. Τέλος οι πελάτες που αντιστοιχούνται στον τελευταίο συνδυασμό FALSE-FALSE (F-F) δηλαδή θέλουν ξηρό φορτίο και βρίσκονται εκτός πόλης μπορούν να εξυπηρετηθούν από τα από όλα τα φορτηγά τα οποία απαρτίζουν τον στόλο μας.

Από την επίλυση του προβλήματος θα πρέπει να προκύψει η βέλτιστη ανάθεση των κόμβων-πελατών στο δρομολόγιο κάποιου φορτηγού, ούτως ώστε κάθε κόμβος να εξυπηρετείται και η βέλτιστη δρομολόγηση των φορτηγών να ελαχιστοποιεί το χρόνο εξυπηρέτησης των κόμβων-πελατών.

Επομένως προκύπτει ότι οι αποφάσεις που θα ληφθούν απ᾽ την επίλυση του C.V.R.P.H.F. είναι : Η ανάθεση των κόμβων-πελατών σε κάποιο φορτηγό και η δρομολόγηση καθενός από τα φορτηγά. Αυτό είναι λογικό καθώς αυτή η διπλωματική εργασία ασχολείται με μια παραλλαγή του προβλήματος δρομολόγησης .

3ο Κεφάλαιο

Παρουσίαση Μοντέλου

Μετά την αναγνώριση του είδους του προβλήματος που μου ανατέθηκε καθώς και της αποδόμησης του, σειρά έχει η συλλογή των στοιχείων και των υποθέσεων που θα αξιοποιηθούν. Εν συνεχεία θα αναλυθούν παρακάτω μέσω των περιορισμών του μαθηματικού μοντέλου. Ενδεικτικά θα αναφέρω κάποιες υποθέσεις οι οποίες συμπεριλήφθησαν στο πρόβλημα με στόχο την επίλυση του.

1. Όλοι οι πελάτες πρέπει να εξυπηρετηθούν
2. Κάθε πελάτη πρέπει να τον επισκεφθεί ένα φορτηγό ακριβώς μία φορά.
3. Όλα τα φορτηγά πρέπει να ξεκινάνε και να επιστρέφουν, στο τέλος του δρομολογίου στην αποθήκη (depot).
4. Δεν θα παραβιάζεται η χωρητικότητα των φορτηγών σε βάρος (kg) και σε όγκο (m^2).

Επίσης τα συγκεκριμένα <<δεδομένα>> μπορούν να χωριστούν σε δύο κατηγορίες:

Α) Των φορτηγών,

Β) Των πελατών

Τα στοιχεία των καταναλωτών/πελατών:

1. Η ζήτηση του προσφερόμενου προϊόντος του κάθε πελάτη τόσο σε βάρος όσο και σε όγκο,
2. Η συγκεκριμένες απαιτήσεις των πελατών σε συγκεκριμένου τύπου φορτηγά από τα οποία θα εξυπηρετηθούν, όπου οι πελάτες κατατάσσονται σε κατηγορίες (κατηγορία TRUE(T) και κατηγορία FALSE (F)),

Τα στοιχεία των φορτηγών:

1. Χωρητικότητα του κάθε φορτηγού σε βάρος (Kg),
2. Χωρητικότητα του κάθε φορτηγού σε όγκο (m3)
3. Οι Συγκεκριμένες δυνατότητες των φορτηγών όπου κατατάσσονται σε διαφορετικού τύπου φορτηγά ώστε να εξυπηρετήσουν τους αντιστοίχους πελότες (τύπου FALSE (F) και TRUE (T)).

Δείκτες

- i, j, z: κόμβοι-πελάτες ή κόμβος-σταθμός (i, j, z=0 : depot),

- k: αριθμός του κάθε φορτηγού.

Δεδομένα

- W_i: ζήτηση του κόμβου/πελάτη- i σε αγαθά μετρημένο σε Kg (i ∈ N , i ∈ (1 , 171)).
- V_i : ζήτηση του κόμβου/πελάτη- i μετρημένο σε όγκο (m^3)(i ∈ N , i ∈(1 , 171)).
- $T_{i,j}$: απαιτούμενος χρόνος για την μετάβαση από ένα σημείο διανομής i σε έναν σημείο διανομής j (i, j∈ N , i, j ∈ (0 , 171)) όπου για τιμή 0 συμβολίζεται το depot.
- Td_i: Χρόνος εξυπηρέτησης Κάθε πελάτη i για να παραλάβει την παραγγελία του αφότου φτάσει το φορτηγό σ αυτόν (i∈ N , i ∈ (1 , 171)).
- CW_k : Μέγιστη χωρητικότητα του φορτηγού k σε kg . (k ∈ N , k ∈ (1 , 105))
- CV_kk : Μέγιστη χωρητικότητα του φορτηγού k σε m3. (k ∈ N , k ∈ (1 , 105))
- $Reqtruck1_k$: Πρώτη δυνατότητα του k-φορτηγού η οποία όταν είναι ίση με μηδέν ($Reqtruck1_k$=0) τότε η απαίτηση είναι TRUE αντίθετα είναι FALSE.
- $Reqtruck2_k$: Δεύτερη δυνατότητα των k-φορτηγών όπου και αυτή με την σειρά της όταν είναι μηδέν ($Reqtruck2_k$=0) τότε η απαίτηση είναι TRUE ενώ στην αντίθετη περίπτωση είναι FALSE.

- $Reqcust1_j$: Πρώτη απαίτηση των i-πελατών όπου αν μηδέν ($Reqcust1_j$=0) τότε η απαίτηση είναι TRUE ενώ στην αντίθετη περίπτωση είναι FALSE.

- $Reqcust2_j$: Δεύτερη απαίτηση των i-πελατών όπου και αυτή με την σειρά της όταν είναι μηδέν ($Reqcust2_j$=0) τότε η απαίτηση είναι TRUE ενώ στην αντίθετη περίπτωση είναι FALSE.

Μεταβλητές απόφασης:

- X_{ijk}: Δυαδική Μεταβλητή απόφασης. Είναι η βασική μεταβλητή απόφασης του προβλήματος ,όπου οι δείκτες αντιστοιχούν σε δρομολόγιο (από το σημείο i στο σημείο j), ενώ ο δείκτης K αντιπροσωπεύει τα φορτηγά. Στην περίπτωση όπου ο δείκτης αυτός παίρνει την τιμή 1, αυτό σημαίνει πως το φορτηγό K εκτελεί δρομολόγιο από τον καταναλωτή i στον καταναλωτή j , και 0 εάν δεν εκτελείται αυτός ο συνδυασμός δρομολογίου-φορτηγού.

- U_i: Πρόκειται για μια γενική ακέραια μεταβλητή της οποίας η χρήση είναι στην επίλυση του γνωστού προβλήματος sub-touring που εμφανίζεται σε προβλήματα VRP και δείχνει τη σειρά εξυπηρέτησης του κάθε πελάτη από κάθε φορτηγό. Η χρήση της γίνεται στον αλγόριθμο των Miller-Tucker-Zemlin (1960) και παρουσιάζεται παρακάτω.

Αντικειμενική Συνάρτηση

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{K} X_{ijk} * T_{ij}$$

Περιορισμοί

1. $\sum_{\kappa=0}^{K}\sum_{j=0}^{N} X_{ijk} = 1, \forall i = 1, ..., N$

2. $\sum_{\kappa=0}^{K}\sum_{i=0}^{N} X_{ijk} = 1, \forall j = 1..., N$

3. $\sum_{j=1}^{N} X_{0,j,k} \le 1, \forall k = 0, ..., \mathrm{K}$

4. $\sum_{i=1}^{N} X_{i0k} \le 1, \forall k = 0, ..., K$

5. $U_i - U_j + N\sum_{k=0}^{K} X_{ijk} \le N - 1, \forall i = 1..., N, j = 1..., N$

6. $\mathrm{Re}qTruck1_k * \sum_{i=0}^{N} X_{ijk} \le \mathrm{Re}qCust1_j, \forall k, ..., K, \forall j = 1...., N$

7. $\mathrm{Re}qTruck2_k * \sum_{i=0}^{N} X_{ijk} \le \mathrm{Re}qCust2_j, \forall k...., K, \forall j = 1...., N$

8. $\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ijk} * (T_{ij} + Tdi) \le \max T(\min), \forall k.....K$

9. $\sum_{i=0}^{N}\sum_{j=0}^{N} (W_i * X_{ijk}) \le CW_k, \forall k....., K$

10. $\sum_{i=0}^{N}\sum_{j=0}^{N} (\mathrm{V}_i * X_{ijk}) \le CV_k, \forall k....., K$

11. $X_{ijk} = 0 - 1, \forall \mathrm{i}, \mathrm{j}, \mathrm{k}$

12. $U_i \ge 0, U_i \in Z^+$

Ανάλυση Περιορισμών

(1) Από κάθε κόμβο θα φύγει ακριβώς ένα φορτηγό ακριβώς μια φορά,

(2) Σε κάθε κόμβο θα πάει ακριβώς ένα φορτηγό ακριβώς μια φορά πρέπει να επισκεφθεί από κάποιο φορτηγό ακριβώς μία φορά,

(3) Ο περιορισμός αυτός αναγκάζει τα οχήματα τα οποία εν τέλει θα δρομολογηθούν να φύγουν από την βάση τους (αποθήκη) το πολύ μια φορά,

(4) Εξαναγκάζει τα φορτηγά μετά την εξυπηρέτηση των i-πελατών οι οποίοι τους έχουν ανατεθεί να επιστρέψουν στην αποθήκη,

(5) Ο περιορισμός αυτός χρησιμοποιήθηκε για να απαγορεύσει τα φορτηγά να δημιουργήσουν υποδιαδρομές και αποτελεί ένα βασικό πυλώνα-περιορισμό των VRP (sub tour elimination constraint).Για την επίλυση του χρησιμοποιήθηκε ο αλγόριθμος των Miller-Tucker-Zemlin (1960),

(6) Ο περιορισμός αυτός αφορά το $1^{ο}$ χαρακτηριστικό των πελατών και φορτηγών (TRUE ή FALSE),

(7) Ο περιορισμός αυτός αφορά το $2^{ο}$ χαρακτηριστικό των πελατών και φορτηγών (TRUE ή FALSE),

(8) Από τα δεδομένα του προβλήματος έχουμε ότι υπάρχει ανάγκη εργασίας για 8 ώρες απασχόλησης των οδηγών επομένως ο περιορισμός αυτός μας βεβαιώνει ότι θα τηρηθεί και δεν θα παραβιαστεί το 8ωρο εργασίας.

(9) Για κάθε φορτηγό η συνολική ζήτηση μετρημένη σε κιλά (kg) των πελατών που θα εξυπηρετήσει δεν θα πρέπει να ξεπερνάει την χωρητικότητα του.

(10) Για κάθε φορτηγό η συνολική ζήτηση μετρημένη σε όγκο (m^3) των πελατών που θα εξυπηρετήσει δεν θα πρέπει να ξεπερνάει την χωρητικότητα του.

(11) Εξασφαλίζει ότι η μεταβλητή X_{ijk} θα είναι δυαδική.

(12) Η U_i. είναι γενική ακέραια μεταβλητή .

Τέλος η αντικειμενική συνάρτηση αποτελεί την ελαχιστοποίηση του συνολικού χρόνου εξυπηρέτησης όλων των πελατών από τα φορτηγά.

4ο ΚΕΦΑΛΑΙΟ

ΟΜΑΔΟΠΟΙΗΣΗ(CLUSTERING)

4.1 ΕΙΣΑΓΩΓΗ

Ομαδοποίηση ή συσταδοποίηση δεδομένων (data clustering) είναι μια τεχνική στατιστικής ανάλυσης δεδομένων και χρησιμοποιείται σε πολλούς τομείς όπως: η μηχανική, η εξόρυξη δεδομένων, η αναγνώριση προτύπων, η ανάλυση εικόνων, βιοπληροφορική προκειμένου να βρεθούν ομάδες σε μεγάλες ποσότητες δεδομένων. Δηλαδή σκοπός της ομαδοποίησης είναι κατά κύριο λόγο η συσταδοποίηση μιας συγκέντρωσης από δεδομένα όπως πχ αναφέρθηκαν παραπάνω και εν συνεχεία ο αλγόριθμος της ομαδοποίησης τα κατατάσσει σε ομάδες (clusters) με βάση κάποιο μέτρο ομοιότητας. Επίσης υπάρχει η ανάγκη ομαδοποίησης και δεν λύνουμε απευθείας με όλους τους πελάτες, επειδή τα προβλήματα του πραγματικού κόσμου είναι πολύ μεγάλα και περιλαμβάνουν πάρα πολλούς πελάτες αυξάνοντας εκθετικά το μέγεθος του μαθηματικού μοντέλου του VRP. Για τον λόγο αυτό απαιτείται η κατηγοριοποίηση των πελατών σε ομάδες και η επίλυση του προβλήματος δρομολόγησης οχημάτων για κάθε ομάδα ξεχωριστά.

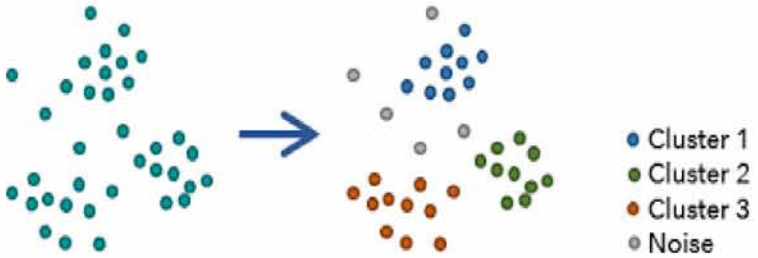


Figure 7 Clustering

Κάποιες εφαρμογές που είναι απαραίτητη η χρήση της ομαδοποίησης είναι :

- Η ταξινόμηση βιβλίων στις βιβλιοθήκες,
- Η ταξινόμηση των φυτών και των ζωών σε κατηγορίες σύμφωνα με τα χαρακτηριστικά τους,
- Ταξινόμηση σπιτιών σύμφωνα με τον τύπο τους, την αγοραστική αξία τους και τη γεωγραφική τους θέση.

4.2 ΦΑΣΕΙΣ ΟΜΑΔΟΠΟΙΗΣΗ

Όπως προαναφέρθηκε προηγουμένως η ομαδοποίηση περιγράφεται ως ο διαχωρισμός των δεδομένων σε ομάδες παρόμοιων αντικειμένων. Όπως μπορούμε να κατανοήσουμε η ομαδοποίηση δεν μπορεί να είναι μια διαδικασία η οποία μας δίνει αποτελέσματα μέσα από ένα βήμα αλλά από μια αλληλουχία σταδίων τα οποία είναι τα εξής:

1. Συλλογή δεδομένων : Σαν πρώτο βήμα παίρνουμε/επιλέγουμε προσεκτικά τα δεδομένα που θα χρησιμοποιήσουμε από τις πηγές που έχουμε,
2. Αρχικός έλεγχος : αναφέρεται στην διαχείριση των δεδομένων μετά την εξαγωγή τους από την πηγή. Αυτό το στάδιο είναι στενά συνδεμένο με το data cleaning, μια διαδικασία που είναι ευρέως γνωστή ως αποθήκη δεδομένων,
3. Αναπαράσταση δεδομένων : τα δεδομένα επεξεργάζονται με σκοπό να γίνουν κατάλληλα για τον αλγόριθμο του clustering. Σ αυτό το στάδιο έχουμε την εξέταση των χαρακτηριστικών και των διαστάσεων των δεδομένων και επιλέγεται το μέτρο σύγκρισης,
4. Ομαδοποιητική τάση : ελέγχει αν τα δεδομένα που θα χρησιμοποιηθούν μπορούν να μπουν σε συστάδες ή όχι,
5. Στρατηγικές ομαδοποίησης : προσεκτική επιλογή του αλγόριθμου ομαδοποίησης και των αρχικών παραμέτρων,
6. Ερμηνεία των αποτελεσμάτων : Σ αυτήν την τελική φάση τα αποτελέσματα της ομαδοποίησης , τα όποια μας τα παρείχε ο αλγόριθμος τα συγκρίνουμε με άλλες μελέτες.

4.3 ΑΛΓΟΡΙΘΜΟΙ ΟΜΑΔΟΠΟΙΗΣΗΣ

Οι περισσότεροι αλγόριθμοι ομαδοποίησης στηρίχτηκαν κυρίως στους 2 πιο διαδομένους αλγορίθμους οι οποίοι είναι οι :

1. Ιεραρχικοί (hierarchical),
2. Μη- ιεραρχικοί (non-hierarchical/partitional).

4.3.1 HIERARCHICAL ALGORITHMS

Οι αλγόριθμοι σ αυτήν την κατηγορία δημιουργούν ένα δενδροδιάγραμμα (cluster tree-dendrogram) με την χρήση ευρετικής διαίρεσης ή συγχώνευσης τεχνικών. Το δενδροδιάγραμμα ορίζεται ως ένα ‘‘δένδρο’’ όπου τα μεμονωμένα στοιχεία είναι από τη μια μεριά και η ομάδα με το κάθε στοιχείο από την άλλη.

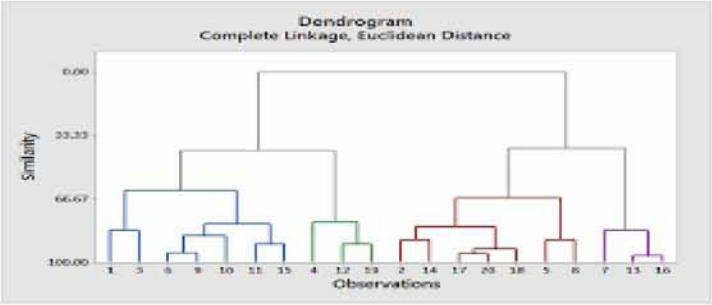


Figure 8 dendrogramm

Οι αλγόριθμοι οι οποίοι χρησιμοποιούν τον διαχωρισμό για να δημιουργήσουν το δενδροδιάγραμμα ονομάζονται διαχωριστικοί. Από την άλλη πλευρά οι πιο δημοφιλείς αλγόριθμοι οι οποίοι και αυτοί με την σειρά τους χρησιμοποιούν την συγχώνευση για την δημιουργία cluster tree ονομάζονται συσσωρευτικοί. Με άλλα λόγια αυτές οι δυο υποκατηγορίες των ιεραρχικών αλγορίθμων έχουν τις εξής φιλοσοφίες :

- Διαχωριστικοί αλγόριθμοι (Divisive algorithms): Ξεκινούν με μια ενιαία ομάδα όλα τα αντικείμενα και διαιρούν διαδοχικά τις ομάδες σε μικρότερες, μέχρι κάθε αντικείμενο ενταθεί σε ένα σύμπλεγμα. Ο διαχωριστικός αλγόριθμος προσεγγίζει τα αντικείμενα δεδομένων σε μη μεικτές ομάδες σε κάθε βήμα και ακολουθεί το ίδιο μοτίβο μέχρι να ενταχθούν όλα τα αντικείμενα σε ένα ξεχωριστό cluster,

- Συσσωρευτικός αλγόριθμος (Agglomerative algorithm): οι συσσωρευτικοί αλγόριθμοι αρχίζουν με κάθε αντικείμενο να είναι ένα ξεχωριστό cluster, και διαδοχικά τα αντικείμενα να συγχωνεύονται σε ομάδες σύμφωνα με τα μέτρα αποστάσεων. Η ομαδοποίηση μπορεί να σταματήσει εάν όλα τα αντικείμενα βρίσκονται σε μία ομάδα ή σε οποιοδήποτε άλλο σημείο ο χρήστης θέλει. Αυτές οι μέθοδοι γενικά ακολουθούν μια ‘‘άπληστη’’ συγχώνευση από κάτω προς τα πάνω.

4.3.2 NON HIERARHICAL/ PARTITIONAL ALGORITHMS

Οι αλγόριθμοι σ αυτήν την κατηγορία διαιρούν το σύνολο δεδομένων σε έναν καθορισμένο αριθμό cluster. Αυτοί οι αλγόριθμοι προσπαθούν να ελαχιστοποιήσουν ορισμένα κριτήρια και επομένως μπορούν να αντιμετωπιστούν ως βέλτιστα προβλήματα. Εξαιτίας των πλεονεκτημάτων των non hierarchical/ partitional algorithms είναι πιο διαδεδομένοι από αυτούς των hierarhical στην αναγνώριση μοτίβων. Στην κατηγορία των partitional algorithms ανήκουν οι εξής αλγόριθμοι :

- The K-means Algorithm,
- The Fuzzy C-means Algorithm,
- The Gaussian Expectation-Maximization Algorithm,
- The K-harmonic Means Algorithm,
- Affinity Propagation.

4.4 Ο ΑΛΓΟΡΙΘΜΟΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΕ

Ο αλγόριθμος ο οποίος δημιουργήθηκε και υλοποιήθηκε ώστε να χρησιμοποιηθεί κατάλληλος, με αποτέλεσμα να μπορέσει το πλήθος των πελατών να ομαδοποιηθεί σε συστάδες είναι ο Affinity Propagation (AP). Αρχικά ο παραπάνω αλγόριθμος αποτελεί υποκατηγορία των μη ιεραρχικών αλγορίθμων (partitional algorithms). Εν συνεχεία ο AP δημιουργεί συστάδες στέλνοντας μηνύματα μεταξύ των σημείων δεδομένων μέχρι τη σύγκλιση. Ο συγκεκριμένος αλγόριθμος έχει κάποια πλεονεκτήματα σε σχέση με άλλους αλγορίθμους ομαδοποίησης όπως:

- Επιτυγχάνει πολύ μικρότερα σφάλματα ομαδοποίησης σε σχέση με ήδη προϋπάρχοντες αλγορίθμους,
- Μπορεί να υποστηρίξει ομοιότητες που δεν είναι συμμετρικές ή δεν ικανοποιούν την τριγωνική ανισότητα,

Τα στοιχεία εισόδου του Affinity Propagation είναι οι ομοιότητες μεταξύ των σημείων δεδομένων T[i,j] όπου τα i,j= 1,2,…..,N με N να είναι ο μέγιστος αριθμός πελατών όπου στην παρούσα διπλωματική το N=171. Κάθε είδος ομοιότητας είναι αποδεκτή όπως για παράδειγμα η αρνητική ευκλείδεια απόσταση. Δεδομένης της ομοιότητας του πίνακα Ti,j[i,j] ο AP επιχειρεί να βρει τα δείγματα που μεγιστοποιούν την καθαρή ομοιότητα δηλαδή το συνολικό άθροισμα των ομοιώσεων μεταξύ των exemplars και τα στοιχεία που ανήκουν στα δεδομένα.

Η διαδικασία του AP μπορεί να προβληθεί ως μια διαδικασία διέλευσης μηνυμάτων με 4 είδη μηνυμάτων που ανταλλάσσονται μεταξύ των σημείων δεδομένων:

- Similarity [Sij [imax][jmax]].
- Responsibility [Rij[imax][jmax]],
- Availability [Aij[imax][jmax]],
- Criterion [Cij [imax][jmax]].

Αρχικά ο πινάκας ο οποίος αφορά τον Similarity είναι ίσος με την αρνητική τιμή της διαφοράς τετραγώνων του T(i,j) .Όσων αφορά τον πίνακα responsibility Rij αποτελεί ένα μήνυμα από το σημείο δεδομένων i έως j που αντανακλά τα συσσωρευμένα στοιχεία για τα πόσο καλά προσαρμοσμένο στοιχείο j είναι να χρησιμεύει ως exemplar για το σημείο δεδομένων i. Επιπροσθέτως ο Aavailability Aij είναι

ένα μήνυμα από τα στοιχεία j στο i που αντικατοπτρίζει τα συσσωρευμένα στοιχεία για το πόσο κατάλληλο θα ήταν για τα σημείο i να επιλέξει το σημείο δεδομένων j ως exemplar. Ο Sij, ο Rij και ο Aij έχουν ρυθμιστεί στο μηδέν και οι τιμές τους ενημερώνονται για τον υπολογισμό των τιμών σύγκλισης από τις σχέσεις :

1. S(i , j)= -||xi-xj||²
2. $R(i,j) \leftarrow S(i,j) - \max_{j', j' \neq j} \{A(i,j') + S(i,j')\}$,
3. $a) A(j,j) \leftarrow \sum_{i', i' \neq j} \max\{0, R(i',j)\}$.

   b) $A(i,j) \leftarrow min\big(0, R(j,j) + \sum_{i' \not\in} \max\big(0, R(i', \mathrm{j})\big)\big), \forall i \neq j$
4. $C(i,j) \leftarrow R(i,j) + A(i,j)$

Η 1η εξίσωση S(i,j) αποτελεί τον πίνακα του Similarity για τια στοιχεία i,j όπου η δημιουργία του οφείλεται όταν υπολογίζουμε το αρνητικό άθροισμα των τετραγώνων της διαφοράς μεταξύ των στοιχείων i,j

Η 2η εξίσωση στην ουσία αναφέρεται στο άθροισμα των τιμών άνω του 0 κατά μήκος της στήλης εκτός από την σειρά της οποίας η τιμή είναι ίση με την εν λόγο στήλη. Δηλαδή αφαιρει από την γραμμή το μικρότερο στοιχείο από τα υπόλοιπα αυτής και όταν φτάσει στην αφαίρεση του συγκεκριμένου στοιχείου αφαιρει το αμέσως μεγαλύτερο.

Η 3η εξίσωση που αφορά τον Availability προσθέτει όλα τα θετικά στοιχεία στην διαγώνιο στην περίπτωση όπου δεν έχει θετικά στοιχεία η διαγώνιος η τιμή που θα πάρει είναι μηδέν. Στην συνέχεια για τα στοιχεία που δεν ανήκουν στην διαγώνιο κάνει το εξής:

- Πάει στα στοιχεία με i=j και προσθέτει την τιμή αυτή στα θετικά στοιχεία της στήλης και έπειτα αντικαθιστά όλα με αυτήν την τιμή,

- Στην περίπτωση όπου μετά το i=j και την πρόσθεση των θετικών στοιχείων το αποτέλεσμα είναι θετικό τότε όλα τα στοιχεία όπου είναι μικρότερα του μηδενός είναι όλα μηδέν ενώ τα θετικά στοιχεία τα αφαιρούμε από το i=j.

Στην συνέχεια δημιουργούμε έναν πίνακα κριτηρίου (Criyterion matrix Cij[imax][jmax]) του οποίου κάθε κελί είναι τα άθροισμα του Rij και του Aij μαζί.

Η υψηλότερη τιμή του κριτηρίου κάθε σειρά ορίζεται ως exemplar. Οι σειρές που μοιράζονται το ίδιο exemplar βρίσκονται στην ίδια συστάδα. Συνοψίζοντας από τον πίνακα κριτηρίου όπου αποτελεί και το τελικό κομμάτι του αλγορίθμου φτιάχνονται και τα ζεύγη τιμών του AP καθώς οι γραμμές του πίνακα Cij οι οποίες έχουν ή μοιράζονται τα ίδια στοιχεία τότε αυτά ανήκουν στην ίδια ομάδα ή μπορούμε να πούμε ότι ανήκουν στο ίδιο υπόδειγμα (exemplar).

4.4.1 ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Μετά την διεκπεραίωση του αλγορίθμου του Affinity Propagation οι πελάτες ομαδοποιηθήκαν στις εξής συστάδες :

1. Η ομάδα 1η έπειτα από την ομαδοποίηση περιέχει τους εξής πελάτες:

Table 1 1η λίστα πελατών

Ομάδα 1η	Πελάτες
	1-8-11-12-13-14-16-24-25-26-39-40-41-43-44-54-58-59-60- 62- 66- 69 -70 -72- 74- 76- 81- 85 - 94- 95- 99- 108- 114- 125- 130- 131- 139- 142- 144- 151

.

2. Η ομάδα 2η έπειτα από την ομαδοποίηση περιέχει τους εξής πελάτες:

Table 2 2η λίστα πελατών

Ομάδα 2η	Πελάτες

2 -15- 28- 32- 33- 34- 36- 37- 38- 47- 53- 61- 68- 71- 75- 87- 88- 90- 106- 109- 115- 120- 122- 129- 132- 136- 137- 138- 140- 145- 146- 147- 148- 150- 152- 153- 156- 157

3. Η ομάδα 3η έπειτα από την ομαδοποίηση περιέχει τους εξής πελάτες:

Table 3 3η λίστα πελατών

Ομάδα 3η	Πελάτες
	3-5-6 -7-20-21-22 27-31-35-45- 46-52-55-56-57-64-65- 67-77-78-82- 84-89-93-97- 98- 103- 119 -121- 123 127 -128 -141-155- 158

4. Η αμάδα 4η έπειτα από την ομαδοποίηση περιέχει τους εξής πελάτες:

Table 4 4η λίστα πελατών

Ομάδα 4η	Πελάτες
	4-9-10-17-18-19-23-29-30 -42-48 -49-50-51 63-73-79-80-83 -86- 92-100-110-113-116-118-124-35-143-159

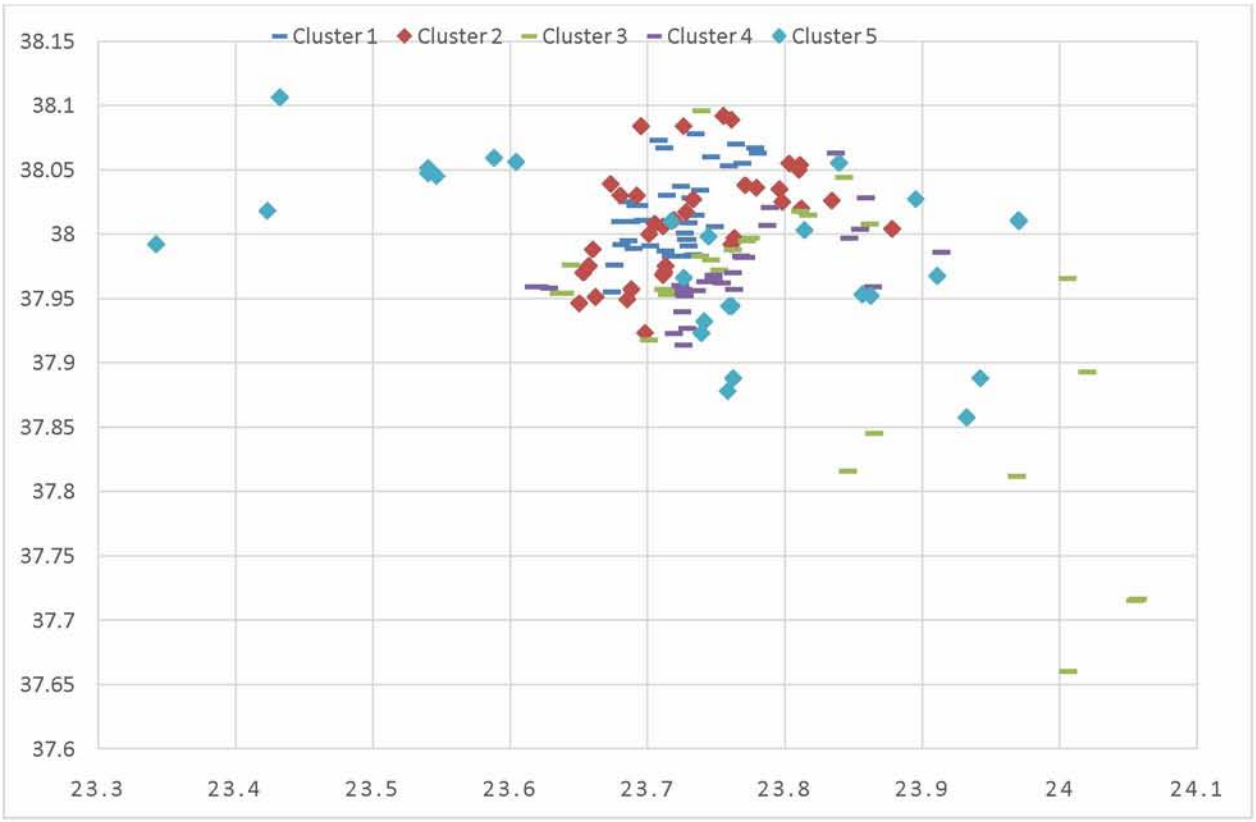
.

5. Η ομάδα 5η έπειτα από την ομαδοποίηση περιέχει τους εξής πελάτες :

Table 5 5η λίστα πελατών

Ομάδα 5η	Πελάτες
	27-64-78-91-96-97-98-101-102-103-104-105-107 111-112-117-123-126-133-134-149-154-155 160 161-162-163-164-165-166 167-168-169-170-171

Και τέλος οι πελάτες με βάση τις συντεταγμένες που απέχουν μεταξύ τους καθώς και την ομαδοποίηση που πραγματοποιήθηκε αποτυπώνονται στο παρακάτω γράφημα :



Γράφημα 1: Γραφική αναπαράσταση πελατών.

5ο ΚΕΦΑΛΑΙΟ

ΕΠΙΛΥΣΗ

5.1 Εισαγωγή

Η επίλυση των προβλημάτων πραγματοποιήθηκε με την βοήθεια των βιβλιοθηκών βελτιστοποίησης της CPLEXILOGIBM σε Microsoft Visual Studio 2010 στο περιβάλλον C++. Τα χαρακτηριστικά του υπολογιστή που χρησιμοποιήθηκε ώστε να υλοποιηθεί η παρούσα διπλωματική ήταν Windows 10 i5-3665 με εγκατεστημένη μνήμη της τάξης των 8 gb ram.

Αρχικά λόγο του μεγάλου όγκου των πελατών στο πρόβλημα έπρεπε να χρησιμοποιηθεί η μέθοδος της ομαδοποίησης (clustering method) , με την βοήθεια του Affinity Propagation ώστε να χωριστεί το πλήθος των πελατών ο οποίος ισούται με N= 171 σε 5 υποομάδες ή αλλιώς συστάδες με τον αριθμό των εκάστοτε πελατών να διαφέρει από ομάδα σε ομάδα. Επίσης ο στόλος των φορτηγών που είναι κατανεμημένος σε κάθε ομάδα είναι διαφορετικός μεταξύ τους.

Οι συνθήκες τερματισμού των επιλύσεων για κάθε μια ξεχωριστά από τις 5 φορές που χρειάστηκε να τρέξει ο κώδικας όσες και οι ομάδες των πελατών διαφέρουν μεταξύ τους τόσο στον χρόνο όσο και στο gap.

5.2 Αποτελέσματα

5.2.1 1η Ομάδα

Η λίστα των πελατών και φορτηγών που ανήκουν στην πρώτη ομάδα είναι η εξής:

Table 6 Ομαδοποιημένη λίστα πελατών και φορτηγών.

Ομάδα πελατών	1-8-11-12-13-14-16-24-25-26-39-40-41-43-44-54-

	58-59-60-62-66-69-70-72-74-76-81-85-94-95-99-108-114-125-130-131-139-142 144-151
Ομάδα φορτηγών	12-28-16-86

Table 7 Πελάτες και ζητήσεις τους για 1η ομαδα.

Πελάτης	Ζήτηση σε κιλά	Ζήτηση σε όγκο	Είδος πελάτη
1	2098	102	TF
8	1055	60	FT
11	159	13	FT
12	104	8	TT
13	3121	145	FF
14	1017	70	TT
16	418	21	FT
24	1067	48	FF
25	3986	263	FF
26	103	4	FF
39	1519	110	FF
40	2669	195	FF
41	646	42	TT
43	1186	76	FF
44	93	8	TT
54	686	50	FF
58	322	30	FF
59	266	28	FT
60	191	17	FT
62	133	11	FT
66	171	14	FT
69	273	25	FF
70	94	9	FT
72	163	14	FT
74	105	10	FT
76	88	8	FT
81	138	11	FT
85	864	63	FF
94	189	15	FT
95	92	10	FT
99	149	14	FF
108	155	13	FT
114	128	15	FT
125	168	15	TT

130	169	15	FT
131	579	28	FF
139	174	25	TT
142	208	32	TT
144	502	40	TF
151	149	18	FT

.

Table 8 Φορτηγά και οι χωρητικότητες τους.

Φορτηγά	Χωρητικότητα σε κιλά	Χωρητικότητα σε όγκο	Είδος φορτηγού
12	5330	350	TT
28	4100	350	FT
16	9550	700	TF
86	25000	1300	FF

Μετά το τρέξιμο του κώδικα τα αποτελέσματα του είναι τα παρακάτω :

Table 9 Αποτελέσματα 1ου κώδικα.

Διαδρομή φορτηγού 1ου	0-44-41-130-142-81-11-95-14-60-74-62-76-125-66 - 139-70-12-151-0
Διαδρομή φορτηγού 2ου	0-69-18-8-43-59-114-108-94-720
Διαδρομή φορτηγού 3ου	0-131-58-99-1-13-54-144-0
Διαδρομή φορτηγού 4ου	0-24-26-25-85-39-40- 0

Η συνολική ζήτηση των πελατών για το 1^{ο} γκρουπ είναι 3900 κιλά και 336 m^3 (όγκος), ενώ το 1^{ο} φορτηγό να έχει συνολική χωρητικότητα 5330kg και 350 m^3 και να είναι τύπος TT. Με αποτέλεσμα οι πελάτες να εξυπηρετούνται άνετα στις ζητήσεις τους καθώς και στα ποια φορτηγά θα τους εξυπηρετήσουν. Εν συνεχεία το 2^{ο} γκρουπ έχει συνολική ζήτηση της τάξης των 2723kg και 267 m^3 ενώ η μέγιστη χωρητικότητα τα 2^{ου} φορτηγού είναι 4100κιλα και 350 m^3 και ο τύπος του είναι FT. Όσον αφορά το 3^{ο} δρομολόγιο έχει μέγιστες χωρητικότητας· 7457kg και 409 m^3 ενώ το αντίστοιχο φορτηγό έχει 9550kg και 700 m^3 με τύπο TF. Τέλος η τελευταία διαδρομή καλύπτεται πλήρως καθώς τα φορτηγό έχει χωρητικότητες 25000kg και 1300 m^3 με τύπο FF.

Οι συνθήκες τερματισμού της επίλυσης για την 1^{η} ομάδα πραγματοποιήθηκε στο χρονικό περιθώριο της 1 ώρας με gap μικρότερου του 30%.

Η τιμή της αντικειμενικής συνάρτησης για την 1^{η} ομάδα είναι ίση με 406 λεπτά, τιμές που αναλογούν στην συνολική εξυπηρέτηση των πελατών από τον στόλο των φορτηγών που επιλέχτηκαν. Με αποτέλεσμα καλυφτούν όλες οι απαιτήσεις των πελατών και ως εκ τούτου με την προϋπόθεση ότι οι πελάτες θα εξυπηρετηθούν από τα αρμόδια οχήματα.

Το φορτηγό 1 θα εξυπηρετήσει τους πελάτες του σε χρόνο ίσο σε 122 λεπτά, το φορτηγό 2 θα χρειαστεί 97 λεπτά, το φορτηγό 3 θα εξυπηρετήσει όλους τους πελάτες του σε 125 λεπτά ενώ το τελευταίο φορτηγό θα κάνει 62 λεπτά.

5.2.2 Ομάδα 2η

Η λίστα των πελατών και φορτηγών που ανήκουν στην δεύτερη ομάδα είναι η εξής:

Table 10 Λίστα πελατών και φορτηγών 2ης ομάδας.

Ομάδα πελατών	2-15-28-32-33-34-36-37-38-47-53-61-68-71-75-87-88-90-106-109-115-120-122-129-132-136-137-

	138-140-145-146-147-148-150-152-153-156-157
Ομάδα φορτηγών	13-29-75-87

Table 11 Πελάτες και οι ζητήσεις τους για 2η ομάδα.

Πελάτης	Ζήτηση σε κιλά	Ζήτηση σε όγκο	Είδος πελάτη
2	518	20	TT
15	1265	80	TF
28	247	121	FF
32	280	24	FF
33	1245	67	TF
34	1859	69	FF
36	886	48	TF
37	273	58	FF
38	586	30	FF
47	247	15	TF
53	286	20	FF
61	174	13	FT
68	162	12	FF
71	200	16	FT
75	193	17	FT
87	357	18	FT
88	391	38	FT
90	210	20	FT
106	400	40	FT
109	147	14	FT
115	187	16	FT
120	123	20	FT
122	111	8	TT
129	130	10	FT
132	532	30	TT
136	146	23	FT
137	212	36	TT
138	251	37	TT
140	2620	600	FF
145	164	17	TT
146	271	33	TT
147	239	22	TT
148	225	20	TT
150	170	29	FT

152	128	6	FF
153	1958	109	TF
156	564	30	FF
157	212	10	FF

Table 12 Φορτηγά και οι χωρητικίτητες για 2η ομάδα.

Φορτηγά	Χωρητικότητα σε κιλά	Χωρητικότητα σε όγκο	Είδος φορτηγού
13	4424	30	TT
29	4100	350	FT
75	9550	300	TF
87	25000	1300	FF

Μετά το τρέξιμο του κώδικα τα αποτελέσματα του είναι τα παρακάτω :

Table 13 Αποτελέσματα 2ου κώδικα.

Διαδρομή φορτηγού 1ου	0-90-152-47-132-137-147-138-115-146-122-148-157- 145-150-136-2-0
Διαδρομή φορτηγού 2ου	0-61-109-120-53-106-75-87-88-71-129 -0
Διαδρομή φορτηγού 3ου	0-33-32-34-38-37-36-153-15-28-68-0
Διαδρομή φορτηγού 4ου	0-156-140-0

Η συνολική ζήτηση των πελατών για το 1ο γκρουπ είναι 3823 κιλά και 342 m^3 (όγκος), ενώ το 1ο φορτηγό να έχει συνολική χωρητικότητα 4424kg και 350 m^3 και να είναι τύπος TT. Με αποτέλεσμα οι πελάτες να εξυπηρετούνται και στις ζητήσεις τους καθώς και στα ποια φορτηγά θα τους εξυπηρετήσουν. Εν συνεχεία το 2ο γκρουπ έχει συνολική ζήτηση της τάξης των 2401kg και 206 m^3 ενώ η μέγιστη χωρητικότητα τα 2ου φορτηγού είναι 4100κιλα και 350 m^3 και ο τύπος του είναι FT. Όσον αφορά το 3ο δρομολόγιο έχει μέγιστες χωρητικότητας· 9461kg και 618 m^3 ενώ το αντίστοιχο φορτηγό έχει 9550kg

και 700 m^3 με τύπο TF. Τέλος η τελευταία διαδρομή καλύπτεται πλήρως καθώς τα φορτηγό έχει χωρητικότητες 25000kg και 1300 m^3 με τύπο FF.

Οι συνθήκες τερματισμού της επίλυσης για την 2η ομάδα πραγματοποιήθηκε στο χρονικό περιθώριο της 1,5 ώρας με gap μικρότερου του 20%.

Η τιμή της αντικειμενικής συνάρτησης για την 2η ομάδα είναι ίση με 591 λεπτά, που αναλογούν στην συνολική εξυπηρέτηση των πελατών από τον στόλο των φορτηγών που επιλέχτηκαν. Με αποτέλεσμα καλυφτούν όλες οι απαιτήσεις των πελατών και ως εκ τούτου με την προϋπόθεση ότι οι πελάτες θα εξυπηρετηθούν από τα αρμόδια οχήματα.

Το φορτηγό 1 θα εξυπηρετήσει τους πελάτες του σε χρόνο ίσο σε 215 λεπτά, το φορτηγό 2 θα χρειαστεί 174 λεπτά, το φορτηγό 3 θα εξυπηρετήσει όλους τους πελάτες του σε 122 λεπτά ενώ το τελευταίο φορτηγό θα κάνει 80 λεπτά.

5.2.3 Ομάδα 3η

Η λίστα των πελατών και φορτηγών που ανήκουν στην τρίτη ομάδα είναι η εξής:

Table 14 Λίστα πελατών και φορτηγών για 3η ομάδα.

Ομάδα πελατών	3-5-6-7-20-21-22-31-35-45-46-52-55-56-57-65-67-77- 82-84-89-93-119-121-127-128-141-158
Ομάδα φορτηγών	71-30-50-88

Table 15 Πελάτες και οι ζητήσεις για 3η ομάδα

Πελάτης	Ζήτηση σε κιλά	Ζήτηση σε όγκο	Είδος πελάτη
3	450	47	FT

5	129	12	FT
6	118	10	FT
7	297	25	FT
20	1288	70	TT
21	722	35	TT
22	2129	144	FF
31	395	35	FT
35	62	1	TT
45	84	10	TT
46	152	30	TT
52	1310	98	FT
55	263	25	FT
56	113	10	FF
57	378	30	FF
65	279	20	FF
67	235	18	FF
77	189	15	FT
82	121	3	TF
84	399	64	TF
89	52	10	FT
93	139	11	TT
119	413	43	TT
121	221	20	TF
127	186	32	FT
128	181	18	FT
141	299	41	TT
158	428	25	TF

Table 16 Φορτηγά και οι χωρητικότητες τους για 3η ομάδα.

Φορτηγά	Χωρητικότητα σε κιλά	Χωρητικότητα σε όγκο	Είδος φορτηγού
71	5300	350	TT
30	4100	350	FT
50	9550	700	TF
88	25000	1300	FF

Table 17 Αποτελέσματα 3ου γκρουπ κώδικα.

Διαδρομή φορτηγού 1ου	0-35-20-21-121-84-46-93-82-45-119-141-77-0
Διαδρομή φορτηγού 2ου	0-3-6-52-57-31-128-65-67-127-89 55→0
Διαδρομή φορτηγού 3ου	0-22-15-8-57-56-0

Η συνολική ζήτηση των πελατών για το 1ο γκρουπ είναι 4089 κιλά και 343 m^3 (όγκος), ενώ το 1ο φορτηγό να έχει συνολική χωρητικότητα 5330kg και 350 m^3 και να είναι τύπος TT. Με αποτέλεσμα οι πελάτες να εξυπηρετούνται και στις ζητήσεις τους καθώς και στα ποια φορτηγά θα τους εξυπηρετήσουν. Εν συνεχεία το 2ο δρομολόγιο έχει συνολική ζήτηση της τάξης των 3895kg και 350 m^3 ενώ η μέγιστη χωρητικότητα του 2ου φορτηγού είναι 4100κιλα και 350 m^3 και ο τύπος του είναι FT. Όσον αφορά το 3ο δρομολόγιο έχει μέγιστες χωρητικότητας· 3048kg και 209 m^3 ενώ το αντίστοιχο φορτηγό έχει 9550kg και 700 m^3 με τύπο TF.

Οι συνθήκες τερματισμού της επίλυσης για την 3η ομάδα πραγματοποιήθηκε στο χρονικό περιθώριο των 2 ωρών με gap μικρότερου του 10%.

Η τιμή της αντικειμενικής συνάρτησης για την 3η ομάδα είναι ίση με 693 λεπτά , που αναλογούν στην συνολική εξυπηρέτηση των πελατών από τον στόλο των φορτηγών που επιλέχτηκαν. Με αποτέλεσμα καλυφτούν όλες οι απαιτήσεις των πελατών και ως εκ τούτου με την προϋπόθεση ότι οι πελάτες θα εξυπηρετηθούν από τα αρμόδια οχήματα.

Το φορτηγό 1 θα εξυπηρετήσει τους πελάτες του σε χρόνο ίσο σε 320 λεπτά, το φορτηγό 2 θα χρειαστεί 186 λεπτά, ενώ το τελευταίο φορτηγό 3 θα εξυπηρετήσει όλους τους πελάτες του σε 187 λεπτά

5.2.4 Ομάδα 4η

Η λίστα των πελατών και φορτηγών που ανήκουν στην τέταρτη ομάδα είναι η εξής:

Table 18 Λίστα πελατών και φορτηγών για 4η ομάδα.

Ομάδα πελατών	4-9-10-17-18-19-23-29-30-42-48-49—50-51-63-73-79-80-83-86-92-100-110-113-116-118-124-135-143-159
Ομάδα φορτηγών	38-3-60-73-89

Table 19 Πελάτες και οι ζητήσεις τους για 4η ομάδα.

Πελάτης	Ζήτηση σε κιλά	Ζήτηση σε όγκο	Είδος πελάτη
4	141	13	FT
9	1228	67	TF
10	533	28	FF
17	160	300	FF
18	1009	113	TF
19	758	42	TT
23	274	15	FT
29	170	25	FT
30	185	19	FT
42	87	9	FF
48	393	24	FT
49	307	17	FT
50	1193	62	FT
51	86	8	FT
63	239	23	FT
73	252	20	FT
79	377	43	TF
80	174	10	TF
83	348	18	TT
86	353	17	FT
92	126	10	FT
100	101	8	FT
110	215	15	FT
113	145	12	FT
116	87	13	FT
118	191	24	FT
124	247	27	TT
135	251	29	FF
143	768	111	FF
159	283	14	

Table 20Φορτηγα και οι χωρητικότητες τους για 4η ομάδα.

Φορτηγά	Χωρητικότητα σε κιλά	Χωρητικότητα σε όγκο	Είδος φορτηγού
38	3880	300	TT
3	3290	300	FT
60	3655	250	FT
73	8340	600	TF
89	2500	1300	FF

Table 21 Αποτελέσματα 4ης ομάδας.

Διαδρομή φορτηγού $1^{ου}$	0-42-86-110-23-73-29-63-92-19-118- 113-135-79 -0
Διαδρομή φορτηγού $2^{ου}$	0-124-116-4-30-50-49-100-51-143-0
Διαδρομή φορτηγού $3^{ου}$	0-83-48-159-18-17-10-9-0

Η συνολική ζήτηση των πελατών για το $1^{ο}$ γκρουπ είναι 3438κιλά και 290 m^3 (όγκος), ενώ το $1^{ο}$ φορτηγό να έχει συνολική χωρητικότητα 3880kg και 300 m^3 και να είναι τύπος TT. Με αποτέλεσμα οι πελάτες να εξυπηρετούνται και στις ζητήσεις τους καθώς και στα ποια φορτηγά θα τους εξυπηρετήσουν. Εν συνεχεία το $2^{ο}$ δρομολόγιο έχει συνολική ζήτηση της τάξης των 3115kg και 278 m^3 ενώ η μέγιστη χωρητικότητα του $2^{ου}$ φορτηγού είναι 3290κιλα και 300 m^3 και ο τύπος του είναι FT. Όσον αφορά το $3^{ο}$ δρομολόγιο έχει μέγιστες χωρητικότητας· 4128kg και 580 m^3 ενώ το αντίστοιχο φορτηγό έχει 8340kg και 600 m^3 με τύπο TF.

Οι συνθήκες τερματισμού της επίλυσης για την $4^{η}$ ομάδα πραγματοποιήθηκε στο χρονικό περιθώριο των 2.5 ωρών με gap μικρότερου του 25%.

Η τιμή της αντικειμενικής συνάρτησης για την $4^{η}$ ομάδα είναι ίση με 423, που αναλογούν στην συνολική εξυπηρέτηση των πελατών από τον στόλο των φορτηγών που επιλέχτηκαν. Με αποτέλεσμα καλυφτούν όλες οι απαιτήσεις των πελατών και ως εκ τούτου με την προϋπόθεση ότι οι πελάτες θα εξυπηρετηθούν από τα αρμόδια οχήματα.

Το φορτηγό 1 θα εξυπηρετήσει τους πελάτες του σε χρόνο ίσο σε 204 λεπτά, το φορτηγό 2 θα χρειαστεί 106 λεπτά ενώ, το φορτηγό 3 θα εξυπηρετήσει όλους τους πελάτες του σε 113 λεπτά.

5.2.5 Ομάδα 5η

Η λίστα των πελατών και φορτηγών που ανήκουν στην πέμπτη ομάδα είναι η εξής:

Table 22 Λίστα πελατών και φορτηγών για 5η ομάδα.

Ομάδα πελατών	27-64-78-91-96-97-98-101-102-103-104-105-107-111-112-117-123-126-133-134-149-154-155-160-161→162-163-164-165-166-167-168-169-170-171
Ομάδα φορτηγών	46-8-20-90-37

Table 23 Πελάτες και οι ζητήσεις τους για 5η ομάδα.

Πελάτης	Ζήτηση σε κιλά	Ζήτηση σε όγκο	Είδος πελάτη
27	176	17	FF
64	140	12	FT
78	155	10	FF
91	626	21	TF
96	143	13	FF
97	143	12	FT
98	325	30	FT
101	128	12	FT
102	88	8	FF
103	95	8	FT
104	105	10	TF
105	256	21	FT
107	185	16	TT
111	107	8	FF
112	116	9	FT

117	221	30	FT
123	312	25	FT
126	143	26	FT
133	378	30	FT
134	126	13	TT
149	227	20	TT
154	4102	336	TF
155	3158	278	TF
160	577	46	FF
161	969	60	FF
162	36	84	FF
163	2	500	FF
164	14	42	FF
165	8	24	FF
166	32	60	FF
167	20	33	FF
168	52	596	FF
169	31	54	FF
170	29	1060	FF
171	12	60	

Table 24 Φορτηγά και οι χωρητικότητες τους για 5η ομάδα.

Φορτηγά	Χωρητικότητα σε κιλά	Χωρητικότητα σε όγκο	Είδος φορτηγού
46	5330	350	TT
8	3360	300	FT
20	15997	800	TF
52	24000	1300	TF
90	2500	1300	FF

.

Table 25 αποτελέσματα 5ου κώδικα.

Διαδρομή φορτηγού 1ου	0-97-103-160-112-117-107-101-98-27-123-126-105-134-

	149-44-133-0
Διαδρομή φορτηγού $2^{ου}$	0-161-102-96-111-91-104-154-155-78-0
Διαδρομή φορτηγού $3^{ου}$	0-167-164-162-165-170-169-0
Διαδρομή φορτηγού $4^{ου}$	0-166-163-171-168-0

Η συνολική ζήτηση των πελατών για την $1^{η}$ διαδρομή είναι 3548κιλά και 227 m^3 (όγκος), ενώ το $1^{ο}$ φορτηγό να έχει συνολική χωρητικότητα 5330kg και 350 m^3 και να είναι τύπος TT. Με αποτέλεσμα οι πελάτες να εξυπηρετούνται και στις ζητήσεις τους καθώς και στα ποια φορτηγά θα τους εξυπηρετήσουν. Εν συνεχεία το $2^{ο}$ δρομολόγιο έχει συνολική ζήτηση της τάξης των 9363kg και 744 m^3 ενώ η μέγιστη χωρητικότητα του $2^{ου}$ φορτηγού είναι 15997κιλα και 800 m^3 και ο τύπος του είναι TF. Όσον αφορά το $3^{ο}$ δρομολόγιο έχει μέγιστες χωρητικότητας· 1297kg και 138 m^3 ενώ το αντίστοιχο φορτηγό έχει 24000kg και 1300 m^3 με τύπο TF. Ενώ τέλος στο $4^{ο}$ δρομολόγιο οι πελάτες έχουν συνολικές ζητήσεις ίσες με 98 κιλά και 1216 m^3 με το $4^{ο}$ φορτηγό να έχει συνολική χωρητικότητα 25000 κιλά και 1300 m^3

Οι συνθήκες τερματισμού της επίλυσης για την $4^{η}$ ομάδα πραγματοποιήθηκε στο χρονικό περιθώριο της 1 ώρας με gap μικρότερου του 25%.

Η τιμή της αντικειμενικής συνάρτησης για την $5^{η}$ ομάδα είναι ίση με 663 λεπτά , που αναλογούν στην συνολική εξυπηρέτηση των πελατών από τον στόλο των φορτηγών που επιλέχτηκαν. Με αποτέλεσμα καλυφτούν όλες οι απαιτήσεις των πελατών και ως εκ τούτου με την προϋπόθεση ότι οι πελάτες θα εξυπηρετηθούν από τα αρμόδια οχήματα.

Το φορτηγό 1 θα εξυπηρετήσει τους πελάτες του σε χρόνο ίσο σε 247 λεπτά, το φορτηγό 2 θα χρειαστεί 155 λεπτά, το φορτηγό 3 θα εξυπηρετήσει όλους τους πελάτες του σε 166 λεπτά ενώ το τελευταίο φορτηγό θα κάνει 95 λεπτά.

ΠΑΡΑΡΤΗΜΑ Α (Κώδικας VRP)

```
#include <ilcplex/ilocplex.h>

ILOSTLBEGIN

int i, j, h;

int k;


const int      imax = 51;

const int      jmax = 51;

const int      maxt = 480;

const int      hmax = 51;

const int              kmax = 5;

const int      M = 1000;

float                  Wi[imax];

float                  Vi[imax];

float                  CWk[kmax];

float                  CVk[kmax];

float          Tij[imax][jmax];

float          TDi[imax];

double         ReqTruck1k[kmax];

double         ReqTruck2k[kmax];

double         ReqCust1j[jmax];

double         ReqCust2j[jmax];


// ypotheseis
```

```
int main(int argc, char **argv) {
        for (k = 0; k < kmax; k++) {
                CWk[k] = 0;
                CVk[k] = 0;
        }
        for (i = 0; i < imax-1; i++) {
                Wi[i] = 0;
                Vi[i] = 0;
        }

        // Import of Data

        ifstream ifs_1;             //Time from i to j
        ifstream ifs_2;             //Demand in kg(visits)
        ifstream ifs_3;             //Demand in m^3(visits)
        ifstream ifs_4;             //Demand in requirment1(visits)
        ifstream ifs_5;             //Demand in requirment2(visits)
        ifstream ifs_6;             //max capacity of vehicle in kg
        ifstream ifs_7;             //max capacity of vehicle in m^3
        ifstream ifs_8;             //requirment 1 of vehicle
        ifstream ifs_9;             //requirment 2 of vehicle

        ifs_1.open("timeliga.txt");
        if (ifs_1.fail())
        {
```

```
        cout << "input file could not be opened" << endl;
        system("pause");
        exit(1);
}
for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {

                ifs_1 >> Tij[i][j];
                Tij[i][j] = Tij[i][j] / 60;
                //cout << Tij[i][j] << "    ";
        }
        cout << endl;
}
ifs_1.close();

ifs_2.open("visits1.txt");
for (i = 1; i < imax; i++) {
        ifs_2 >> Wi[i];
        //cout<<"Wi["<<i<<"]="<<Wi[i]<<endl;
}

ifs_2.close();

ifs_3.open("visits2.txt");
for (i = 1; i < imax; i++) {
```

```
        ifs_3 >> Vi[i];
        //cout << "Vi[" << i << "]=" << Vi[i] << endl;
}

ifs_3.close();

ifs_4.open("visits3.txt");
for (i = 1; i < imax; i++) {
        ifs_4 >> ReqCust1j[i];
        //cout << "ReqCust1j[" << i << "]=" << ReqCust1j[i] << endl;
}

ifs_4.close();

ifs_5.open("visits4.txt");
for (i = 1; i < imax; i++) {
        ifs_5 >> ReqCust2j[i];
        //cout << "ReqCust2j[" << i << "]=" << ReqCust2j[i] << endl;
}

ifs_5.close();

ifs_6.open("vehic1.txt");
for (k = 0; k < kmax; k++) {
        ifs_6 >> CWk[k];
```

```
        //cout << "CWk[" << k << "]=" << CWk[k] << endl;
}

ifs_6.close();

ifs_7.open("vehic2.txt");
for (k = 0; k < kmax; k++) {
        ifs_7 >> CVk[k];
        //cout << "CVk[" << k << "]=" << CVk[k] << endl;
}

ifs_7.close();

ifs_8.open("vehic3.txt");
for (k = 0; k < kmax; k++) {
        ifs_8 >> ReqTruck1k[k];
        //cout << "ReqTruck1k[" << k << "]=" << ReqTruck1k[k] << endl;
}

ifs_8.close();


ifs_9.open("vehic4.txt");
for (k = 0; k < kmax; k++) {
        ifs_9 >> ReqTruck2k[k];
```

```
        //cout << "ReqTruck2k[" << k << "]=" << ReqTruck2k[k] << endl;
}

ifs_9.close();

// Building Model

IloEnv env;

try {

        IloModel model(env);

        typedef IloArray<IloNumArray> IloNumMatrix2x2;
        typedef IloArray<IloNumMatrix2x2> IloNumMatrix3x3;
        typedef IloArray<IloNumMatrix3x3> IloNumMatrix4x4;

        typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
        typedef IloArray<IloNumVarMatrix2x2> IloNumVarMatrix3x3;
        typedef IloArray<IloNumVarMatrix3x3> IloNumVarMatrix4x4;

        typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
        typedef IloArray<IloRangeMatrix2x2> IloRangeMatrix3x3;
        typedef IloArray<IloRangeMatrix3x3> IloRangeMatrix4x4;
```

```
IloCplex cplex(env);

// Decision Variables

// Xijk
IloNumVarMatrix3x3 Xijk(env, 0);
for (i = 0; i < imax; i++) {
        IloNumVarMatrix2x2 Xjk(env, 0);
        for (j = 0; j < jmax; j++) {
                IloNumVarArray Xk(env, 0);
                for (k = 0; k < kmax; k++) {
                        char Path[1000];
                        sprintf(Path, "Xijk(i%d,j%d,k%d)", i, j, k);
                        IloNumVar X(env, 0, 1, ILOINT, Path);
                        Xk.add(X);
                }
                Xjk.add(Xk);
        }
        Xijk.add(Xjk);
}

//Oi
IloNumVarArray Oi(env, 0);
```

```
for (i = 0; i < imax; i++) {
        char Order[100];
        sprintf(Order, "Oi(i%d)", i);
        IloNumVar O(env, 0, IloInfinity, ILOFLOAT, Order);
        Oi.add(O);
}
// Constraints
// Starting from i node we have to travel to another j node

IloRangeArray Sum1_Xi(env, 0);
for (i = 1; i < imax; i++) {
        IloExpr expr(env, 0);
        for (j = 0; j < jmax; j++) {
                if (i != j) {
                        for (k = 0; k < kmax; k++) {
                                expr += Xijk[i][j][k];
                        }
                }
        }
        char con1[60];
        sprintf(con1, "Sum1_Xi(i%d)", i);
        int LB = 1, UB = 1;
        IloRange Sum1_X(env, LB, expr, UB, con1);
        expr.end();
        model.add(Sum1_X);
```

```
        Sum1_Xi.add(Sum1_X);
}



// Constraint (2)
// Each customer must be visited exactly once

IloRangeArray Sum2_Xj(env, 0);
for (j = 1; j < jmax; j++) {
        IloExpr expr(env, 0);
        for (i = 0; i < imax; i++) {
                if (i != j) {
                        for (k = 0; k < kmax; k++) {
                                expr += Xijk[i][j][k];
                        }
                }
        }
        char con2[60];
        sprintf(con2, "Sum2_Xj(j%d)", j);
        int LB = 1, UB = 1;
        IloRange Sum2_X(env, LB, expr, UB, con2);
        expr.end();
        model.add(Sum2_X);
        Sum2_Xj.add(Sum2_X);
}
```

```
// Constraint (3)
//Each truck leave from Depot

//IloRangeArray depotijk(env, 0);
//for (k = 0; k < kmax; k++) {
//        IloExpr expr(env, 0);
//        for (j = 1; j < jmax; j++) {
//                expr += Xijk[0][j][k];
//        }
//        char con3[60];
//        sprintf(con3, "depotij(k%d)", k);
//        float LB = -IloInfinity, UB = 1;
//        IloRange depotij(env, LB, expr, UB, con3);
//        model.add(depotij);
//        depotijk.add(depotij);
//        expr.end();
//}

//// Constraint (4)
//// Each vehicle retutn to depot
//IloRangeArray Sum4_Xk(env, 0);
//for (k = 0; k < kmax; k++) {
//        IloExpr expr(env, 0);
//        for (i = 1; i < imax; i++) {
```

```
//                expr += Xijk[i][0][k];
//        }
//        char con4[60];
//        sprintf(con4, "  Sum4_Xk(k%d)", k);
//        int LB = -IloInfinity, UB = 1;
//        IloRange Sum4_X(env, LB, expr, UB, con4);
//        expr.end();
//        model.add(Sum4_X);
//        Sum4_Xk.add(Sum4_X);
//}

// Constraint (5)
//subtour elimination
IloRangeMatrix2x2 Subtourij(env, 0);
for (i = 1; i < imax; i++) {
        IloRangeArray Subtourj(env, 0);
        for (j = 1; j < jmax; j++) {
                IloExpr expr(env, 0);
                if (i != j) {
                        for (k = 0; k < kmax; k++) {
                                expr += Xijk[i][j][k];
                        }
                }
                expr += Oi[i] - Oi[j] + (imax-1) * expr - (imax-1) + 1;
                char con5[100];
```

```
                sprintf(con5, "Subtourij(i%d,j%d)", i, j);
                float LB = -IloInfinity, UB = 0;
                IloRange Subtour(env, LB, expr, UB, con5);
                expr.end();
                model.add(Subtour);
                Subtourj.add(Subtour);
        }
        Subtourij.add(Subtourj);
}

// Constraint (6a)
//requirments(T-F)
IloRangeMatrix2x2 Sum6ajk(env, 0);
for (k = 0; k < kmax; k++) {
        IloRangeArray Sum6aj(env, 0);
        for (j = 1; j < jmax; j++) {
                IloExpr expr(env, 0);
                for (i = 0; i < imax; i++) {
                        expr += Xijk[i][j][k];
                }
                expr = expr * ReqTruck1k[k];
                expr -= ReqCust1j[j];
                char con6a[100];
                sprintf(con6a, "Sum6ajk(j%d,k%d)", j, k);
```

```
                float LB = -IloInfinity, UB = 0;
                IloRange Sum6a(env, LB, expr, UB, con6a);
                expr.end();
                model.add(Sum6a);
                Sum6aj.add(Sum6a);
        }
        Sum6ajk.add(Sum6aj);
}

// Constraint (6b)
//requirments(T-F)
IloRangeMatrix2x2 Sum6bjk(env, 0);
for (k = 0; k < kmax; k++) {
        IloRangeArray Sum6aj(env, 0);
        for (j = 1; j < jmax; j++) {
                IloExpr expr(env, 0);
                for (i = 0; i < imax; i++) {
                        expr += Xijk[i][j][k];
                }
                expr = expr * ReqTruck2k[k];
                expr -= ReqCust2j[j];
                char con6b[100];
                sprintf(con6b, "Sum6bjk(j%d,k%d)", j, k);
                float LB = -IloInfinity, UB = 0;
                IloRange Sum6b(env, LB, expr, UB, con6b);
```

```
                expr.end();
                model.add(Sum6b);
                Sum6aj.add(Sum6b);
        }
        Sum6bjk.add(Sum6aj);
}

// constraint (7)
//if goes to i, it leaves i

IloRangeMatrix2x2 Leavehk(env, 0);
for (k = 0; k < kmax; k++) {
        IloRangeArray Leaveh(env, 0);
        for (h = 1; h < hmax; h++) {
                IloExpr expr(env, 0);

                for (i = 0; i < imax; i++) {
                        expr += Xijk[i][h][k];
                }
                for (j = 0; j < jmax; j++) {
                        expr -= Xijk[h][j][k];
                }

                char con7[60];
                sprintf(con7, "Leavehk(h%d,k%d)", h, k);
```

```
                float LB = 0, UB = 0;
                IloRange Leave(env, LB, expr, UB, con7);
                model.add(Leave);
                Leaveh.add(Leave);
                expr.end();
        }
        Leavehk.add(Leaveh);
}

//constraint 8
//8 hour Limit

IloRangeArray Shiftk(env, 0);
for (k = 0; k < kmax; k++) {
        IloExpr expr(env, 0);
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        expr += Xijk[i][j][k] * (Tij[i][j] + TDi[i]);
                }
        }
        char con8[60];
        sprintf(con8, "Shiftk(k%d)", k);
        float LB = 0, UB = maxt;
        IloRange Shift(env, LB, expr, UB, con8);
        expr.end();
```

```
        model.add(Shift);
        Shiftk.add(Shift);
}



//constraint 9
//Weight Capacity


IloRangeArray CapacityWeight_k(env, 0);
for (k = 0; k < kmax; k++) {
        IloExpr expr(env, 0);
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        expr += Wi[i] * Xijk[i][j][k];
                }
        }
        expr -= CWk[k];
        char WeightCapacity[100];
        sprintf(WeightCapacity, "CapacityWeight_k(k%d)", k);
        double LB = -IloInfinity, UB = 0;
        IloRange CapacityWeight(env, LB, expr, UB, WeightCapacity);
        expr.end();
```

```
        model.add(CapacityWeight);
        CapacityWeight_k.add(CapacityWeight);
}


//constraint 10
//volume Capacity(m3)


IloRangeArray CapacityVolume_k(env, 0);
for (k = 0; k < kmax; k++) {
        IloExpr expr(env, 0);
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        expr += Vi[i] * Xijk[i][j][k];
                }
        }
        expr -= CVk[k];
        char VolumeCapacity[100];
        sprintf(VolumeCapacity, "CapacityVolume_k(k%d)", k);
        double LB = -IloInfinity, UB = 0;
        IloRange CapacityVolume(env, LB, expr, UB, VolumeCapacity);
        expr.end();
        model.add(CapacityVolume);
        CapacityVolume_k.add(CapacityVolume);
```

```
}



IloExpr exprobj(env);


for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                for (k = 0; k < kmax; k++) {
                        exprobj += Xijk[i][j][k] * Tij[i][j];
                        //exprobj += Xijk[i][j][k] * TDi[i];
                }
        }
}


model.add(IloMinimize(env, exprobj));
exprobj.end();


//solve
cplex.extract(model);
cplex.exportModel("onoma.lp");
cplex.setParam(IloCplex::EpGap, 0.2687);
cplex.solve();
// results
if (!(cplex.solve())) {
```

```
            env.error() << "Faild to optimize LP." << endl;
            throw(-1);
        }

        env.out() << "Solution status = " << cplex.getStatus() << endl;
        env.out() << "Solution value = " << cplex.getObjValue() << endl;



        for (k = 0; k < kmax; k++) {
            for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                    if (i != j) {
                        float g = cplex.getValue(Xijk[i][j][k]);
                        if (g != 0) cout << "Xijk" << "(" << i << "," << j
<< "," << k << ")" << "=" << g << endl;
                    }
                }
            }
        }
}
catch (IloException& e) {
        cerr << "concert exception caught:" << e << endl;
}
catch (...) {
```

```
            cerr << "Unknown exception caught" << endl;
        }

        env.end();

        system("pause");
        return 0;

}
```

ΠΑΡΑΡΤΗΜΑ Β (Αλγόριθμος Ομαδοποίησης)

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN



int i, j, v, k;
const int      imax = 171;
const int      jmax = 171;


const int      BigM = 100000;
float          Tij[imax][jmax];
float          Sij[imax][jmax];//Similarity between i and j
float          Rij[imax][jmax];//Responsibility between i and j
float          Aij[imax][jmax];//Availability between i and j
float          Cij[imax][jmax];//Criterion between i and j
float          MaxA_S[imax][jmax];//Parameter equal to Max (A+S)
float          MaxO_R[imax][jmax];//Parameter equal to Max (0,R)
float          MinO_R[imax][jmax];//Parameter equal to Max (0,R+MaxO_R)
float          MaxExemplar[imax];//Parameter equal to Max (0,R+MaxO_R)
bool                   IsNodeSetToCluster[imax];//True if Node i is set to a cluster
bool                   FirstNode[imax];//First node of a cluster
int                          NodeCluster[imax][jmax];//Node i belongs to cluster j
float                  MinS = BigM;//Minimum of Sij
float                  MedianS = BigM;//Median of Sij
```

```
int                         MaxNoOfExemplars = 5;
float                       damping = 0.9;
float                       Preference = -500000;


int main(int argc, char **argv) {
        // Import of Data

        ifstream ifs_1;                                         //Time from i to j
        ifs_1.open("timeola.txt");
        if (ifs_1.fail())
        {
                cout << "input file could not be opened" << endl;
                system("pause");
                exit(1);
        }
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        ifs_1 >> Tij[i][j];
                        Tij[i][j] = Tij[i][j] / 60;
                        //cout << Tij[i][j] << "    ";
                }
                cout << endl;
        }
        ifs_1.close();
```

```
for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                Aij[i][j] = 0;
                Rij[i][j] = 0;
        }
}
for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                Sij[i][j] = -Tij[i][j];
                if (MinS > Sij[i][j]) {
                        MinS = Sij[i][j];
                }
        }
}
MedianS = 0;
for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                MedianS += Sij[i][j];
        }
}
MedianS = MedianS / (i*j);
```

```
//Preference
for (i = 0; i < imax; i++) {
        Sij[i][i] = Preference;
}
k = BigM;
while (k > MaxNoOfExemplars) {
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        MaxA_S[i][j] = -BigM;
                        for (k = 0; k < imax; k++) {
                                if (k != j && k != i && MaxA_S[i][j] < Aij[i][k] +
Sij[i][k]) {
                                        MaxA_S[i][j] = Aij[i][k] + Sij[i][k];
                                }
                        }
                        Rij[i][j] = Rij[i][j] * damping + (1 - damping)*(Sij[i][j] -
MaxA_S[i][j]);
                }
        }
        //cout << "Rij" << endl;
        /*for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
        cout << Rij[i][j] << "\t";
        }
        cout << endl;
        }
```

```
cout << endl;*/



for (k = 0; k < imax; k++) {
        MaxO_R[k][k] = 0;
        for (i = 0; i < imax; i++) {
                if (i != k && Rij[i][k] > 0) {
                        MaxO_R[k][k] += Rij[i][k];
                }
        }
        Aij[k][k] = Aij[k][k] * damping + (1 - damping)* MaxO_R[k][k];
}


for (i = 0; i < imax; i++) {
        for (k = 0; k < imax; k++) {
                if (i != k) {
                        MaxO_R[i][k] = 0;
                        for (j = 0; j < jmax; j++) {
                                if (j != k && j != i) {
                                        if (Rij[j][k] > 0) {
                                                MaxO_R[i][k] += Rij[j][k];
                                        }
                                }
                        }
```

```
                                        Aij[i][k] = Aij[i][k] * damping + (1 -
damping)*(Rij[k][k] + MaxO_R[i][k]);
                                        if (Aij[i][k] > 0) {
                                                Aij[i][k] = 0;
                                        }
                                }
                        }
                }
                //cout << "Aij" << endl;
                /*for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                cout << Aij[i][j] << "\t";
                }
                cout << endl;
                }
                cout << endl;*/
                for (i = 0; i < imax; i++) {
                        for (j = 0; j < jmax; j++) {
                                Cij[i][j] = Rij[i][j] + Aij[i][j];
                        }
                }
                //cout << "Cij" << endl;
                /*for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                cout << Cij[i][j] << "\t";
```

```
}
cout << endl;
}*/
for (i = 0; i < imax; i++) {
        IsNodeSetToCluster[i] = false;
        MaxExemplar[i] = -BigM;
        for (j = 0; j < jmax; j++) {
                if (MaxExemplar[i] < Cij[i][j]) {
                        MaxExemplar[i] = Cij[i][j];
                }
        }
}
/*for (i = 0; i < imax; i++) {
cout << "MaxExemplar[" << i << "]=" << MaxExemplar[i] << endl;
}*/
for (i = 0; i < imax; i++) {
        IsNodeSetToCluster[i] = false;
        FirstNode[i] = false;
        for (j = 0; j < imax; j++) {
                NodeCluster[i][j] = 0;
        }
}
k = 0;
for (i = 0; i < imax; i++) {
        if (IsNodeSetToCluster[i] == false) {
```

```
                        NodeCluster[i][k] = 1;
                        IsNodeSetToCluster[i] = true;
                        FirstNode[i] == true;
                        for (j = i + 1; j < jmax; j++) {
                                if (IsNodeSetToCluster[j] == false && MaxExemplar[j]
== MaxExemplar[i]) {
                                        NodeCluster[j][k] = 1;
                                        IsNodeSetToCluster[j] = true;
                                }
                        }
                        k++;
                }
        }
        /*for (j = 0; j < jmax; j++) {
        for (i = 0; i < imax; i++) {
        if (NodeCluster[i][j] == 1) {
        cout << "NodeCluster[" << i << "][" << j << "]=" << NodeCluster[i][j] <<
endl;
        }
        }
        }*/
        cout << "MaxNoOfExemplars=" << k << endl;
    }//end while
    std::ostringstream osClusters;
    osClusters << "C:\\Results_TSP\\Clusters(AFFINITY PROPAGATION).txt";
    std::string FileNameClusters = osClusters.str();
```

```
	std::ofstream fsClusters;
	fsClusters.open(FileNameClusters.c_str(), std::ios::out);
	fsClusters << "Cluster" << "\t" << "Node" << endl;
	for (k = 0; k < imax; k++) {
		for (i = 0; i < imax; i++) {
			if (NodeCluster[i][k] == 1) {
				fsClusters << k + 1 << "\t" << i + 1 << endl;
			}
		}
	}
	fsClusters.close();
	system("pause");
	return 0;
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Improvements and extensions to the Miller-Tucker-Zemlin elimination constraints,
2. Zambito, Leonardo.*The Traveling Salesman Problem: A Comprehensive Survey.* s.l. : Submitted as a project for CSE 4080, Fall 2006.
3. Wen, Min.*Rich Vehicle Routing Problems and Applications.* s.l. : PhD Thesis, DTU Management Engineering, July 2010.
4. Gilbert Laporte, Paolo Toth, Daniele Vigo.*Vehicle routing: historical perspective and recent contributions.* s.l. : Springer-Verlag Berlin Heidelberg and EURO - The Association of European Operational Research Societies, March 2013
5. Laporte, G.,Ropke, S., Vidal, T. (2014). Heuristics for the vehicle routing problem. Vehicle routing: Problems, Methods and Applications, Second Edition. Philadelphia: SIAM, 18–87
6. Clarke, G., Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research
7. Kara, I. (2010). Tightening Bounding Constraints of the Miller-Tucker-Zemlin Based Formulation of the Capacitated Vehicle Routing Problems and Some Extensions. In Proceeding of the 2nd International Conference on Manufacturing Engineering, Quality and Production Systems, edited by C. Panait et al. WSEAS Press, Constantza, Romania
8. Solving vehicle routing problem with simultaneous pickup and delivery using genetic algorithm and prins splitting procedure
9. Delbert Dueck and Brendan J. Frey. Non-metric affinity propagation for unsupervised image categorization. In IEEE Int. Conf. Computer Vision (ICCV)
10. Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. Science
11. A. Plangprasopchok, K. Lerman, and L. Getoor, "Integrating structured metadata with relational affinity propagation." in Statistical Relational Artificial Intelligence, 2010