ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Μηχανολόγων Μηχανικών

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΧΡΟΝΙΚΑ ΠΑΡΑΘΥΡΑ ΚΑΙ ΑΝΟΜΟΙΟΓΕΝΗ ΣΤΟΛΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του φοιτητή

Ανδρέα-Βασιλείου Κάλλου

Α.Μ. 1425

Εξεταστική Επιτροπή: **Γεώργιος Σαχαρίδης (Κύριος Επιβλέπων)**

Δημήτριος Παντελής

Γεώργιος Λυμπερόπουλος

Βόλος 2019

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε το διάστημα μεταξύ Ιουνίου 2018 και Απριλίου 2019 στα πλαίσια του προπτυχιακού προγράμματος του Τμήματος Μηχανολόγων Μηχανικών του Πανεπιστημίου Θεσσαλίας. Ως την ελάχιστη δυνατή μνεία, με την παρούσα παράγραφο οφείλω να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνησή της και ιδιαίτερα:

Τον επιβλέποντα καθηγητή μου, Επίκουρο Καθηγητή κύριο Γεώργιο Σαχαρίδη για τη πολύτιμη υποστήριξή του, τις παραγωγικές υποδείξεις του και το πολύ καλό κλίμα συνεργασίας που διαμόρφωσε συμβάλλοντας τα μέγιστα για την κατάρτιση της διπλωματικής μου εργασίας, καθώς και τους διδακτορικούς του φοιτητές που συμμετείχαν στο πρόγραμμα. Σε αυτόν οφείλω επίσης και ένα μεγάλο ευχαριστώ γιατί με δέχτηκε στην ομάδα του προγράμματος «Green your Move" που αποτέλεσε για μένα το ξεκίνημα σε αυτό το αντικείμενο.

Τους γονείς μου, Γιώργο και Στέλλα που με στήριξαν όλα αυτά τα χρόνια καθώς και την αδερφή μου Εύα για την καθοδήγηση που μου προσέφερε.

Τέλος θα ήθελα να ευχαριστήσω όλους τους συναδέλφους μου που συμμετείχαμε μαζί στο μεταπτυχιακό μάθημα 'Εφαρμογές Επιχειρησιακής Έρευνας' για τις ανταλλαγές απόψεων, το ειλικρινές ενδιαφέρον τους και για τη σημαντική βοήθειά τους σε όλα τα στάδια της εργασίας.

ΠΕΡΙΛΗΨΗ

Το εμπόριο αγαθών και η ανταλλαγή προϊόντων έχουν γνωρίσει δραματική πρόοδο τον τελευταίο αιώνα. Πλέον το θέμα της ασφάλειας των ταξιδιών έχει αντιμετωπιστεί και καταβάλλεται συνεχώς προσπάθεια για βελτίωση στον απαιτούμενο χρόνο μεταφοράς. Από τις προσπάθειες αυτές άνθησε ο κλάδος της επιχειρησιακής έρευνας και της εφοδιαστικής αλυσίδας. Σε αυτόν τον κλάδο ανήκουν όλα εκείνα τα προβλήματα στα οποία ο σκοπός τους είναι να βρεθεί μία λύση που θα επιφέρει ικανοποιητικά αποτελέσματα σε αποδεκτό χρόνο. Κατά την εκπόνηση της εργασίας αυτής, αντιμετωπίζεται το πρόβλημα της δρομολόγησης οχημάτων για κάλυψη ζητήσεων γνωστό και ως Vehicle Routing Problem (VRP).

Όλα τα προβλήματα αυτής της κατηγορίας διακρίνονται από τα χαρακτηριστικά τους, όπως συγκεκριμένοι χρόνοι εξυπηρέτησης, περιορισμένος αριθμός οχημάτων, υποχρεωτικές αναθέσεις σε συγκεκριμένα οχήματα κλπ. Το πρόβλημα που αναλύεται παρακάτω χαρακτηρίζεται από ανομοιογένεια του στόλου των διαθέσιμων οχημάτων, περιορισμένη χωρητικότητα σε κάθε όχημα και ύπαρξη χρονικών παραθύρων εξυπηρέτησης για τους πελάτες.

Το πρώτο βήμα της επίλυσης είναι η δημιουργία ενός μοντέλου περιορισμών με τεχνικές μεικτού ακέραιου προγραμματισμού με βάση το οποίο θα γίνονται οι δοκιμές. Μετά την μοντελοποίηση, ακολούθησαν τρεις διαφορετικές δοκιμές και σύγκριση των αποτελεσμάτων τους. Η πρώτη επίλυση προήρθε από την αναλυτική επίλυση του μοντέλου μέσω βιβλιοθηκών βελτιστοποίησης, χωρίς να υπάρχει εφαρμογή κάποιου δοκιμασμένου αλγορίθμου. Στο δεύτερο μέρος πραγματοποιήθηκε ομαδοποίηση των δεδομένων μας με τον αλγόριθμο K-Means και με τις ίδιες τεχνικές υπήρξε αισθητή βελτίωση στον χρόνο εξυπηρέτησης που απαιτείτε. Λόγω της φύσης του προβλήματος όμως, ζητούνταν ένας αρκετά μεγαλύτερος από τον επιθυμητό αριθμό οχημάτων για την γρήγορη κάλυψη των ζητήσεων. Δοκιμάστηκε έτσι σαν τρίτο στάδιο η επεξεργασία ενός άπληστου αλγορίθμου με σκοπό την μείωση του αριθμού των οχημάτων που θα εξυπηρετηθούν σε συνδυασμό με την ελαχιστοποίηση του χρόνου εξυπηρέτησης. Τα αποτελέσματα της περίπτωσης αυτής ήταν αρκετά ικανοποιητικά, καθώς επιτυγχάνεται δραστική μείωση του αριθμού των δρομολογηθέντων οχημάτων, με αποδεκτή αύξηση του χρόνου που θα χρειαστεί για την εξυπηρέτηση, στην περίπτωση ειδικά που η δρομολόγηση πολλαπλών οχημάτων θα έχει κόστος. Τέλος δοκιμάστηκε να πραγματοποιηθεί μία διαδικασία ομαδοποίησης των δεδομένων με διαφορετικά κριτήρια από αυτά της τοποθεσίας τους. Στόχος είναι η ομαδοποίηση των πελατών με βάση την ομοιότητα των χρονικών παραθύρων εξυπηρέτησής τους (Time-Windows). Τα αποτελέσματα στην περίπτωση αυτή χαρακτηρίζονται ικανοποιητικά παρόλο που απέχουν από την

καλύτερη λύση ως προς τις εργατοώρες. Αυτό λόγω του ότι η απαίτηση επιπλέων εργατοωρών συνδυάζεται με μείωση των φορτηγών που απαιτούνται γεγονός που σε άλλες περιπτώσεις είναι κρίσιμο.

APPLICATION OF A CAPACITATED VEHICLE ROUTING PROBLEM WITH TIME-WINDOWS AND HETEROGENEOUS FLEET

Andreas-Vasilios Kallos

EXTENDED ABSTRACT

The transportation of goods on a quick, economic and safe way was always an important issue. The need for reliable trade routes has led to the creation of the first commercial roads. These were either land or sea routes. One of the most famous trade routes is the "silk road" that connects Asia with Europe. Another typical case is the establishment of ship and ground transportation system of the ancient Greek cities with their colonies.

With the industrial revolution and the technological advancement of the last century the amount of produced products has increased dramatically while the exchange of goods between states became more intense. With the progress of time and the rapid development of technology the trade has entirely changed. In the past, the main objective was to travel safely while nowadays it is the speed and cost of transport. Due to the continuous economic growth, there is a need for development of new scientific tools to obtain the best possible decisions on economic and technical issues concerning the transport and distribution of products.

The break-through in the new scientifically-supported transportation field started in the United Kingdom during World War II in order to achieve fast routes for supplying the necessary food and war equipment. Upon the end of the war it was established as a new scientific field with its first applications in the industry and production management in the early 1950s. At the same time, competition was increased. In order to remain competitive in the market the businesses are now called to optimize their efficiency and therefore the profitability. This can be achieved, at least in part, by reducing the managing and transporting cost for produced products. The transfer of manufactured products is a key issue for any business, since optimal planning saves manpower while better and state of the art tools like transport vehicles help saving significant amounts of money. According to the results of various studies, a cost reduction of up to 5% can be achieved if operational research methods are applied properly. This percentage cannot be considered as small if we take in the consideration that a 15% share of the total

production cost corresponds to the cost of its transportation from the production line to the warehouses and from there to the consumption centers. Taking these in the consideration, it has been decide to deal with the Vehicle Routing Problem (VRP) at the framework of this Thesis work.

The main objective of this work is to deal with the particular problem of the assignment of a number of vehicles of a fleet in order to satisfy the requests from 171 customers in the Greater Athens Area. In order to deliver the products, there is a fleet of 105 trucks from which to choose. Every customer must be served at the same day while no driver is allowed to work over 8 hours. Each customer can accept the delivery during a specific time-window within the day.

The Capacitated Vehicle Routing Problem (CVRP) is considered as one of the most complicated problems of the optimization. Particular attention is needed to find a feasible solution that can be as close as possible to the optimal one. The specific problem has a high degree of difficulty because:

- Each vehicle of our fleet has a specific capacity described both in weight and space needed.
- The dimension of time must be considered since each customer has specific times for which he is available of receiving his order, known as Time-Windows.

In addition, we have to deal with the fact that the criteria of optimality for this problem is the total working hours needed in order to satisfy all the requests implies the consideration of more parameters for our solution. Sometimes it will be more profitable for the company to order a vehicle to wait idle for a Time-Window to open rather than order a new vehicle to start from the depot. In this Thesis, four different solutions have been tested with each one providing quite interesting results.

For all four solutions a model of constraint programming is needed. With our mathematical model we set all the constraints and the parameters of the problem described in numbers. Modeling is the first step for every optimization problem since it serves as a common denominator for us and the computer. Inside this model, we express all the parameters and the constraints that need to be respected from a solution in order to describe it as feasible. For this problem, the value of each solution is expressed as the sum of work-hours needed for the satisfaction of all the demands. This value comes from the objective function of our model.

The first solution was approached with an analytical solving-way of the problem while making use of the CPLEX libraries on Visual Studio 2015 environment. Due to the vast number of possible solutions and computational space needed, customer clustering was used. The data clustering for this case was performed in a myopic way, without the usage of any tested algorithm. Solving the smaller routing

problems this offers a starting solution. With this first feasible solution obtained, the next step is to find ways of optimizing its value. These approaches are explained during the next solutions.

The second solution can be described as an improvement of the first one since the clustering of our customers will now be performed in a more scientific and better tested way. From all the possible algorithms available the K-Means algorithm for clustering was preferred for this part. The model adopted for this solution is almost the same with the first one while the solving tool used in this case is ILOG CPLEX Optimization Studio. After performing tests in order to find the optimal number of clusters, the model was set and ready to run. The results obtained were easy to compare and proved the value of the algorithm usage since the time needed for this case was 28% less than the one in the first solution. The solution is feasible and acceptable but there was one factor that pushed us towards the decision to seek another one. The new factor for the new solution was the usage of 39 trucks from our 105-truck fleet in order to satisfy all the demands. This makes perfect sense for the specific problem since there is no cost–penalty for the usage of a truck and the only factor for comparing solutions, which is the total time spent.

Since the main objective of the optimization is the travel time, the optimal solutions will include the usage of more vehicles than they needed. Taking in the consideration the effectiveness of greedy algorithms for finding efficient solutions both in terms of time and quality, a third testing was performed on our problem.

For the third solution the Nearest Neighbor Algorithm (NAA) was modified and used. During the preparation of the code for the solution, most of the constraints of the VRP problem were described with simple comparisons and «if» cases and the usage of a new truck had a minimum empty load. As expected, the results were not better concerning the time value but the number of trucks used is halved. This proves the importance of having more than one «objectives» when facing an optimization problem. In order to achieve this reduction, we performed several tests with a fixed number of vehicles and keep the one with the best results. Finally, another notable fact about the application of the third solution is the reduction of computing time required that for many cases can be critical when deciding for the proper solving tool. Many times, the minimization of the time needed to find a solution is the main objective, especially when time is at the issues in order to take a decision.

Finally, following the Affinity Propagation algorithm methodology, the clients were clustered with different criteria than the one of their location. Many similarities were found in the clients Time-Window times so the new clusters were formed by their similarity functions. The results on this occasion can be described as decent concerning the time needed to serve all the demands, but not optimal. What is

interesting about these results is the fact that all the wait times of the trucks were eliminated since there main criteria of our clustering are the common times of Time-Windows.

Summarizing the findings from the described exercises we can say the following:

- Analytical solutions will offer better numerical values but there is a need of a decision-making mechanism that will take in the consideration every factor of the problem.
- The majority of operational research problems require the usage of tested algorithms on many parts of their solving process.
- The human factor must always be taken in the consideration.
- For many cases simple feasible solutions provide better results in terms of risk or total cost.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

Από την αρχαιότητα μέχρι σήμερα η μεταφορά αγαθών με γρήγορους και ασφαλείς τρόπους έχει προβληματίσει και εξακολουθεί να προβληματίζει τον άνθρωπο. Η ανάγκη αυτή για εμπορία προϊόντων έφερε την δημιουργία των πρώτων εμπορικών δρόμων. Οι δρόμοι αυτοί ήταν είτε χερσαίοι είτε θαλάσσιοι.

Χαρακτηριστικά παραδείγματα τέτοιων δρόμων είναι η σύνδεση των αρχαίων Ελληνικών πόλεων με τις αποικίες τους (π.χ. η Αθήνα με τις πόλεις της Κάτω Ιταλίας). Είναι γνωστοί οι δρόμοι που είχαν οργανωθεί στη Ρωμαϊκή αυτοκρατορία και οι δρόμοι που ένωναν τις πόλεις μεταξύ τους.

Ένας από του πλέον γνωστούς εμπορικούς δρόμους είναι αυτός «του μεταξιού» που ένωνε δύο διαφορετικές ηπείρους την Ασία και την Ευρώπη. Αντίστοιχα παραδείγματα θα μπορούν να αναφερθούν για το εμπόριο με την Ινδία και άλλες χώρες της Ασίας για πρώτες ύλες και σπάνια αγαθά (π.χ. μπαχαρικά).

Με την βιομηχανική επανάσταση πλέον παραγόταν πληθώρα προϊόντων , οι ανταλλαγές αγαθών μεταξύ κρατών έγιναν πιο έντονες, ενώ πλέον με το πέρασμα του χρόνου και την ραγδαία ανάπτυξη της τεχνολογίας το εμπόριο άλλαξε ολοκληρωτικά. Από την εποχή που το βάρος δινόταν στην ασφάλεια του ταξιδιού πλέον το κύριο κριτήριο είναι η ταχύτητα της μεταφοράς και το κόστος της.

Έτσι, με την δημιουργία μεγάλων σταθμών παραγωγής για την οικονομικότερη και ταχύτερη παραγωγή, καθώς και λόγω της συνεχόμενης προόδου της οικονομίας, εμφανίστηκε η ανάγκη για μελέτη νέων επιστημών, με σκοπό την λήψη των βέλτιστων δυνατών αποφάσεων σε οικονομικά και τεχνικά ζητήματα.

Δημιουργήθηκε έτσι ο κλάδος της επιχειρησιακής έρευνας με σκοπό την μελέτη της παραγωγής και της διανομής των αγαθών. Η οργάνωση σε επιστήμη έγινε στην Αγγλία κατά την περίοδο του Β Παγκοσμίου Πολέμου με κύριο στόχο τον ανεφοδιασμό της αγοράς με τα απαραίτητα τρόφιμα και υλικά αλλά και τη χρήση νέων τεχνολογικών εργαλείων όπως το δίκτυο των ραντάρ στην οργάνωση της άμυνας και των περιπολιών. Η επιχειρησιακή έρευνα απετέλεσε χρήσιμο εργαλείο στο σχεδιασμό νέων οπλικών συστημάτων. Με τη λήξη του πολέμου καθιερώθηκε ως νέο επιστημονικό πεδίο με τις πρώτες εφαρμογές της στην βιομηχανία και την διοίκηση παραγωγής τη δεκαετία του 1950.

Με τη λήξη του πολέμου και την έναρξη των διαδικασιών ανασυγκρότησης η παραγωγή αγαθών αυξανόταν κατακόρυφα. Παράλληλα όμως αυξανόταν και ο ανταγωνισμός. Οι σύγχρονες επιχειρήσεις, για να παραμείνουν ανταγωνιστικές στην αγορά, καλούνται πλέον να βελτιστοποιούν την δυνατότητα

κέρδους τους. Αυτό μπορεί να γίνει, εν μέρει τουλάχιστον, με την ελάττωση του κόστους διαχείρισης και μεταφορών των παραγόμενων προϊόντων. Η μεταφορά των παραγόμενων προϊόντων είναι ένα καίριο ζήτημα στις επιχειρήσεις, καθώς με τον βέλτιστο προγραμματισμό εξοικονομούνται εργατοώρες και γίνεται καλύτερη διαχείριση των εργαλείων (π.χ. οχημάτων μεταφοράς) με αποτέλεσμα να εξοικονομούνται σημαντικά χρηματικά ποσά. Σύμφωνα με τα αποτελέσματα διαφόρων ερευνών με σωστή μελέτη των τρόπων μεταφοράς και διανομής των προϊόντων μπορεί να προκύψει μείωση κόστους έως και 5%. Το ποσοστό αυτό δε μπορεί να θεωρηθεί και τόσο μικρό αν αναλογιστεί κανείς πως ένα ποσοστό της τάξη του 15% του συνολικού κόστους ενός προϊόντος αντιστοιχεί στο κόστος μεταφοράς του από το σταθμό παραγωγής στις αποθήκες και από εκεί στα κέντρα κατανάλωσης.

Τα τελευταία χρόνια, με την αύξηση του ηλεκτρονικού εμπορίου και της διεθνοποίησης των συναλλαγών, το θέμα της οργάνωσης των συστημάτων αποθήκευσης και διανομής έγινε ακόμη πιο σημαντικό για την ελαχιστοποίηση του χρόνου και του κόστους παράδοσης και φυσικά της ελαχιστοποίησης του κόστους μεταφοράς και διαχείρισης εν γένει.

Η παρούσα εργασία αφορά ένα κλασικό πρόβλημα διανομής προϊόντων από έναν αποθηκευτικό χώρο στην Αθήνα σε έναν αριθμό πελατών. Για την παράδοση των παραγγελιών του κάθε πελάτη η επιχείρηση έχει στην διάθεση της έναν στόλο φορτηγών που μπορεί να αξιοποιήσει. Το κάθε φορτηγό χαρακτηρίζεται από δύο στοιχεία του:

1. Τη μέγιστη χωρητικότητά του σε όγκο προϊόντος σε κυβικά μέτρα (m^3)
2. Τη μέγιστη χωρητικότητά του σε βάρος προϊόντος σε κιλά (kg)

Κάθε σημείο εξυπηρέτησης έχει μια ζήτηση που πρέπει να ικανοποιηθεί που χαρακτηρίζετε αντίστοιχα από βάρος και όγκο προϊόντος. Γνωρίζοντας τα χρονικά παράθυρα που μπορεί να εξυπηρετηθεί ο κάθε πελάτης καθώς και τις αποστάσεις σε χρόνο και απόσταση μεταξύ όλων των πελατών αλλά και της βάσης, θα στηθεί ένα δίκτυο διανομής το οποίο θα δίνει τη βέλτιστη πιθανή λύση στο πρόβλημα αυτό. Ζητείτε δηλαδή ένα σύστημα που θα βελτιστοποιεί το συνολικό χρόνο ταξιδιών που απαιτούνται για να ικανοποιηθούν όλες οι ζητήσεις λαμβάνοντας υπόψη τους ακόλουθους περιορισμούς:

- Κάθε οδηγός- φορτηγό δεν πρέπει να υπερβαίνει τις 8 ώρες εργασίας;
- Το κάθε φορτηγό έχει συγκεκριμένη χωρητικότητα που διαφέρει από τα υπόλοιπα και δεν πρέπει να παραβιάζεται;
- Τα φορτηγά του στόλου επιτρέπεται να φύγουν και να γυρίσουν στην βάση μία το πολύ φορά;
- Η εξυπηρέτηση των πελατών πρέπει να ξεκινήσει μέσα στα επιτρεπόμενα χρονικά παράθυρα του κάθε πελάτη;
- Στην διάρκεια μίας ημέρας (24 ώρες) θα πρέπει να έχουν ικανοποιηθεί όλες οι ζητήσεις;

Η εργασία αυτή χωρίζεται σε πέντε κεφάλαια ήτοι:

Το πρώτο κεφάλαιο δίνει μία εισαγωγή στην επιστήμη της επιχειρησιακής έρευνας και των προβλημάτων βελτιστοποίησης.

Στο δεύτερο κεφάλαιο παρέχονται γενικές πληροφορίες για τα προβλήματα δρομολόγησης(VRP) και τις κατηγορίες τους.

Στο τρίτο κεφάλαιο περιλαμβάνεται η περιγραφή της μεθοδολογίας που ακολουθήθηκε για την εύρεση μια αρχικής λύσης του προβλήματος μέσω μορφοποίησης και επίλυσης προβλήματος προγραμματισμού βελτιστοποίησης.

Στο τέταρτο κεφάλαιο πραγματοποιείτε χρήση αλγορίθμων ομαδοποίησης δεδομένων για την επίτευξη καλύτερων αποτελεσμάτων.

Στο πέμπτο κεφάλαιο συνοψίζονται τα αποτελέσματα και συμπεράσματα

ΚΕΦΑΛΑΙΟ 2: ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ

2.1. Εφοδιαστική αλυσίδα

Ο όρος της εφοδιαστικής αλυσίδας, γνωστό και ως τομέας «Logistics» παγκοσμίως, είναι μία επιστήμη που από τις αρχές του 1990 παρουσιάζεται όλο και πιο έντονα στις σύγχρονες επιχειρήσεις, έχοντας τεράστιο αντίκτυπο. Η διαχείριση εφοδιαστικής αλυσίδας είναι ο συστηματικός, στρατηγικός συντονισμός των παραδοσιακών επιχειρηματικών λειτουργιών μέσα στην επιχείρηση και μεταξύ των επιχειρήσεων μέσα στην εφοδιαστική αλυσίδα, για τους σκοπούς βελτίωσης της μακροπρόθεσμης απόδοσης των μεμονωμένων επιχειρήσεων και της εφοδιαστικής αλυσίδας ως σύνολο (Mentzer et al. 2001).

Δηλαδή, με την σωστή χρήση των logistics, επιτυγχάνεται ο βέλτιστος προγραμματισμός για την παραγωγή, την αξιοποίηση των απαιτούμενων πρώτων υλών προς αξιοποίηση και η διανομή των αγαθών στην κατανάλωση. Παρόλο που η χρήση του τομέα αυτού εμφανίζεται σε κάθε εταιρία-οργανισμό, η εφαρμογή του επικεντρώνεται κυρίως σε εταιρίες που δραστηριοποιούνται στον χώρο της διανομής προϊόντων (μεταφορικές εταιρίες, εταιρίες αποθήκευσης, αεροδρόμια κ.λ.π).

Συνεπώς, logistics και εφοδιαστικές αλυσίδες συχνά χαρακτηρίζονται ως το ίδιο πράγμα. Η μελέτη των εφοδιαστικών αλυσίδων αποτελεί την κύρια χρήση της επιστήμης αυτής. Με την σωστή μελέτη τους, γίνεται ο σχεδιασμός των παραμέτρων μιας εφοδιαστικής αλυσίδας, όπως είναι η συχνότητα των παραδόσεων, το μέγεθος των παρτίδων προς παράδοση και τα δρομολόγια που θα ακολουθήσουμε για την ελαχιστοποίηση των εξόδων που απαιτούνται για τον σκοπό αυτό. Πλέον προβλήματα που απασχολούσαν συνεχώς τα κέντρα εφοδιασμού επιλύνονται μέσα από τυποποιημένες διαδικασίες. Το πιο «κλασικό» πρόβλημα διανομής, αυτό της δρομολόγησης των οχημάτων ανεφοδιασμού (Vehicle Routing Problem - VRP) μελετάται σε νέα στάδια, λαμβάνοντας υπόψη νέες παραμέτρους και αξιοποιώντας την ασταμάτητη τεχνολογική πρόοδο.

2.2. Γενική εικόνα του προβλήματος δρομολόγησης οχημάτων VRP

Το πρόβλημα δρομολόγησης οχημάτων, γνωστό και ως VRP, αποτελεί σημείο αναφοράς της επιστήμης της επιχειρησιακής έρευνας. Μελετήθηκε πρώτη φορά από τους Dantzig and Ramser (1959), όταν κλήθηκαν να βρουν τον οικονομικότερο σχεδιασμό ανατροφοδότησης βενζίνης από έναν αποθηκευτικό χώρο στους απαιτούμενους σταθμούς εξυπηρέτησης. Αποτελεί μία πιο περίπλοκη μορφή του προβλήματος του πλανόδιου πωλητή. Στο πρόβλημα του πλανόδιου πωλητή, γνωστό και ως Traveling Salesman΄s Problem (TSP), υπάρχει ένας αριθμός πελατών που πρέπει να εξυπηρετηθούν από ένα μόνο όχημα - πωλητή. Σκοπός του είναι να σχεδιαστεί η διαδρομή που θα ακολουθηθεί για την παράδοση των αγαθών, με το μικρότερο δυνατό κόστος. Με την έννοια κόστος, μπορεί να χαρακτηριστεί η χρονική διάρκεια του ταξιδιού, η απόσταση που θα διανυθεί ή το κόστος μεταφοράς των αγαθών.

Στην περίπτωση του VRP, αντί για ένα όχημα μεταφοράς ο σχεδιασμός της ανατροφοδότησης απαιτεί την επιλογή συγκεκριμένων οχημάτων από έναν στόλο που το καθένα εκτελεί ένα ξεχωριστό δρομολόγιο. Σκοπός του, όπως και στο πρόβλημα του Πλανόδιου Πωλητή, είναι η εξυπηρέτηση όλων των πελατών, επιτυγχάνοντας ταυτόχρονα ελαχιστοποίηση για όλα τα κόστη που απαιτούνται.

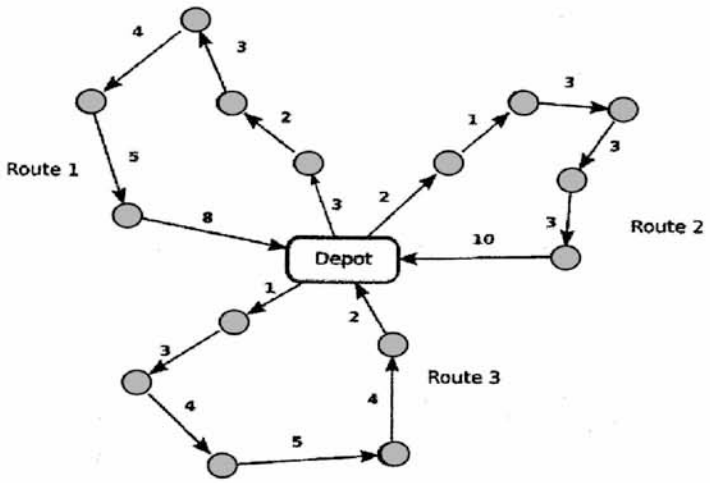
Από το 1959 που έγιναν τα πρώτα βήματα πάνω σε αυτό μέχρι σήμερα έχει πραγματοποιηθεί σημαντικός αριθμός μελετών. Για το πρόβλημα αυτό υπάρχει ένα σύνολο διαφορετικών περιπτώσεων που εξαρτώνται από περιορισμούς που προκύπτουν όταν καλείται κάποιος να σχεδιάσει το δίκτυο διανομής. Επειδή το σύνολο των υποπεριπτώσεων του προβλήματος αυτού είναι μεγάλο, θα ξεκινήσουμε πρώτα με την προσέγγιση του βασικού προβλήματος και στην πορεία θα παρουσιαστούν και θα αναλυθούν τα υπό-προβλήματά του.

2.3. Ορισμός και σκοπός του βασικού προβλήματος δρομολόγηση οχημάτων

Το VRP αποτελεί ένα από τα πιο συχνά εμφανιζόμενα προβλήματα του ακέραιου προγραμματισμού και ανήκει στην κατηγορία των μη πολυωνυμικών δύσκολων προβλημάτων, δηλαδή η δυσκολία επίλυσης του αυξάνεται εκθετικά με την αύξηση του μεγέθους του προβλήματος προς αντιμετώπιση. Στην βασική και απλουστευμένη του μορφή, έχει ως σκοπό την αξιοποίηση ενός στόλου οχημάτων για την ικανοποίηση της ζήτησης ενός αριθμού πελατών. Η ανάθεση των οχημάτων που θα χρησιμοποιηθούν στα προβλήματα δρομολόγησης γίνονται με τέτοιο τρόπο ώστε η λύση που θα βρεθεί να ελαχιστοποιεί τα έξοδα που απαιτούνται για την πλήρη κάλυψη όλων των ζητήσεων. Η ελαχιστοποίηση αυτή προκύπτει μέσα από την επίτευξη κάποιων επιμέρους στόχων του προβλήματος. Οι στόχοι αυτοί συνήθως είναι:

- Η ελαχιστοποίηση του αριθμού οχημάτων που θα δρομολογηθούν, μειώνοντας έτσι τα έξοδα για τους μισθούς οδηγών και τις ενοικιάσεις οχημάτων εάν απαιτούνται;
- Η ελαχιστοποίηση της συνολικής απόστασης που θα διανυθεί από τον στόλο των οχημάτων μας για την πλήρη εξυπηρέτηση των πελατών;
- Η ελαχιστοποίηση του χρόνου που θα χρειαστεί για την πλήρη εξυπηρέτηση των πελατών;
- Η ελαχιστοποίηση «ποινών» που μπορεί να προκύψουν από μη επαρκή κάλυψη αναγκών των πελατών.

Στην απλή μορφή του προβλήματος δρομολόγησης έχουμε ένα κέντρο επιχειρήσεων - αποθηκευτικό χώρο - από το οποίο τα οχήματα που θα δρομολογηθούν ξεκινάνε την αποστολή τους. Στο τέλος του δρομολογίου που έχει να διανύσει κάθε όχημα θα πρέπει να επιστρέφει πίσω στην βάση αυτή. Επίσης, κάθε πελάτης παραλαμβάνει ολόκληρη την παραγγελία του από ένα μόνο όχημα. Τέλος, στην κλασσική αντιμετώπιση του προβλήματος αυτού, ένα όχημα φεύγει και επιστρέφει στην βάση μόνο μία φορά. Μια σχηματική αναπαράσταση για τη δομή λειτουργίας της απλής μορφής του VRP δίνεται στο Σχήμα 1.



Σχήμα 1. Σχηματική αναπαράσταση της δομής λειτουργίας ενός απλού προβλήματος VRP (Tunga H. et al., 2017).

Στο Σχήμα 1 κάθε κύκλος χαρακτηρίζει ένα πελάτη προς εξυπηρέτηση. Με τον όρο «depot» περιγράφεται η βάση των φορτηγών ενώ τα βελάκια δείχνουν την πορεία των οχημάτων που ανέλαβαν τον εφοδιασμό. Στο συγκεκριμένο παράδειγμα, τρία φορτηγά πραγματοποίησαν τρία ξεχωριστά δρομολόγια για να καλύψουν το σύνολο των ζητήσεων (routes 1, 2, 3).

Στην πολύ απλή μορφή του, το μοντέλο επίλυσης ενός προβλήματος VRP χαρακτηρίζεται από μία αντικειμενική συνάρτηση ελαχιστοποίησης κόστους και συγκεκριμένους περιορισμούς όπως παρουσιάζεται παρακάτω:

Η αντικειμενική συνάρτηση προς ελαχιστοποίηση z ορίζεται σαν:

$$z = \min \sum_{i=0}^{imax} \sum_{j=0}^{jmax} \sum_{k=1}^{kmax} Xi,j,k * Ci,j,k \qquad (2.3.1)$$

με τους βασικούς περιορισμούς:

$$\sum_{i=0}^{imax} \sum_{k=1}^{kmax} Xi,j,k = 1 \qquad \forall\ j \in N\ (1, jmax) \qquad (2.3.2)$$

$$\sum_{j=1}^{jmax} Xo,j,k \leq 1 \qquad \forall\ k \in N\ (1, kmax) \qquad (2.3.3)$$

$$\sum_{i=0}^{imax} Xihk - \sum_{j=0}^{jmax} Xhjk = 0 \qquad \forall\ k,h\ ,\ h,k \in N\ (1, max) \qquad (2.3.4)$$

Οι δείκτες i, j παίρνουν τιμές από το μηδέν έως το σύνολο των πελατών προς εξυπηρέτηση, ενώ ο k δείκτης χαρακτηρίζει το σύνολο των οχημάτων του στόλου μας. Στους δείκτες πελατών i και j η θέση 0 είναι η βάση που ξεκινάνε τα δρομολογημένα οχήματα. Με την δυική μεταβλητή απόφασης X εκφράζεται η πραγματοποίηση ή μη του δρομολογίου από το σημείο i στο σημείο j μέσω του οχήματος k. Στην περίπτωση που το δρομολόγιο εκτελείται η μεταβλητή X παίρνει την τιμή 1 αλλιώς τη τιμή 0. Η μεταβλητή C στην γενική της μορφή εκφράζει τα κόστη πραγματοποίησης της διαδρομής από το σημείο i στο σημείο j μέσω του οχήματος k και είναι δεδομένη.

Με τους βασικούς περιορισμούς (2.3.2), (2.3.3) και (2.3.4) εξασφαλίζεται η εξυπηρέτηση όλων των πελατών από μία φορά. Περισσότερη ανάλυση των περιορισμών πραγματοποιείται στο επόμενο κεφάλαιο της εργασίας αυτής που αφορά την επίλυση του προβλήματος. Προφανώς ανάλογα με τις συνθήκες που καλούμαστε να αντιμετωπίσουμε σε ένα πρόβλημα δρομολόγησης προστίθενται αρκετοί περιορισμοί, όπως θα δούμε και στο επόμενο υποκεφάλαιο.

2.4. Τα βασικά είδη του VRP

Η μελέτη και η επίλυση του κάθε προβλήματος VRP αποτελεί ξεχωριστή περίπτωση ανάλογα με το είδος του αντικειμένου που μελετάμε και τις συνθήκες που μας δίνονται σε κάθε περίπτωση. Τα κύρια είδη που έχουν δομηθεί παρουσιάζονται στην πορεία του υποκεφαλαίου αυτού μαζί με τα χαρακτηριστικά και τις ιδιαιτερότητες της κάθε περίπτωσης. Τις περισσότερες φορές εμφανίζεται συνδυασμός των παρακάτω χαρακτηριστικών των προβλημάτων.

2.4.1. Πρόβλημα δρομολόγησης περιορισμένης χωρητικότητας οχημάτων - Capacitated VRP

Χαρακτηρίζουν την πλειοψηφία των προβλημάτων δρομολόγησης σήμερα. Στην περίπτωση του προβλήματος δρομολόγησης οχημάτων με περιορισμένη χωρητικότητα (Capacitated Vehicle Routing Problem - CVRP) τα οχήματα που έχουμε στην διάθεση μας για την πραγματοποίηση της διανομής μπορούν να μεταφέρουν μία συγκεκριμένη ποσότητα προϊόντων. Δηλαδή στην επίλυσή τους είναι σίγουρο ότι για την ολοκληρωτική κάλυψη της ζήτησης θα χρειαστεί η επιστράτευση ενός αριθμού οχημάτων των οποίων η μέγιστη χωρητικότητα δεν μπορεί να παραβιαστεί. Επίσης, επειδή ο κάθε πελάτης θα μπορεί να εξυπηρετηθεί από ένα μόνο όχημα, η ζήτηση του δεν θα πρέπει να ξεπερνάει την μέγιστη χωρητικότητα.

Έχουμε δηλαδή ,πέρα από τους σταθερούς περιορισμούς που περιέχει ένα πρόβλημα VRP, επιπρόσθετους περιορισμούς χωρητικότητας, οι οποίοι έχουν την μορφή:

$$\sum_{i=1}^{imax}\{\sum_{j=0}^{jmax} \text{Qi} * Xi,j,k\} \leq \text{CAP} \qquad \forall\ \text{k} \qquad (2.4.1)$$

Οι δείκτες i και j χαρακτηρίζουν τα σημεία εξυπηρέτησης, παίρνοντας τιμές από μηδέν έως τον αριθμό των πελατών. Με την τιμή 0 χαρακτηρίζουμε την βάση των οχημάτων, το σημείο δηλαδή που πρέπει να ξεκινήσουν και να καταλήξουν την διαδρομή τους. Ο δείκτης k αντίστοιχα παίρνει τιμές από 1 έως το συνολικό αριθμό των οχημάτων, και εκφράζει τα οχήματα. Στον μονοδιάστατο πίνακα Qi τοποθετούνται οι ζητήσεις των πελατών.

Στην περίπτωση του CVRP πρέπει να ληφθεί υπόψη ότι ο στόλος των οχημάτων προς αξιοποίηση μπορεί να χαρακτηρίζεται από ομοιογένεια ή όχι ως προς την χωρητικότητα του δηλαδή τα οχήματα μας να έχουν ίδια χωρητικότητα ή όχι.

Στην πρώτη περίπτωση, αυτή της ομοιογένειας του στόλου, το CAP στην σχέση (2.4.1) είναι η χωρητικότητα των οχημάτων. Η περίπτωση αυτή είναι αρκετά απλή στην επίλυση της καθώς εάν τα

κόστη δρομολόγηση του κάθε οχήματος δεν διαφέρει, η επιλογή των οχημάτων που θα πραγματοποιήσουν δρομολόγιο δεν επηρεάζει την λύση.

Στην περίπτωση όμως που η χωρητικότητα του στόλου μας χαρακτηρίζεται από ετερογένεια (Heterogeneous Fleet) παρουσιάζονται κάποιες αλλαγές. Πλέον η επιλογή των δρομολογίων που θα πραγματοποιηθούν επηρεάζεται από την διαφορετικότητα των οχημάτων. Πρέπει δηλαδή στην επίλυση να εξεταστούν όλες οι δυνατές επιτρεπτές περιπτώσεις με κάθε όχημα για να βρεθεί η βέλτιστη επιλογή.

Η σχέση (2.4.2) παρουσιάζει μία αλλαγή στη δομή καθώς το στοιχείο CAP πλέον μετατρέπεται σε έναν μονοδιάστατο πίνακα CAPk και περιέχει πλέον τις χωρητικότητες του κάθε οχήματος k. Δηλαδή λαμβάνει τη μορφή:

$$\sum_{i=1}^{imax}\{\sum_{j=0}^{jmax} \text{Qi} * Xi,j,k\} \leq \text{CAPk} \qquad \forall\ \text{k} \qquad (2.4.2)$$

2.4.2. Πρόβλημα δρομολόγησης με χρονικά παράθυρα (VRP with Time-Windows)

Μία εξίσου σημαντική περίπτωση των προβλημάτων δρομολόγησης είναι αυτή στην οποία ο ανεφοδιασμός των σημείων πρέπει να πραγματοποιηθεί μέσα σε συγκεκριμένα χρονικά πλαίσια (Vehicle Routing Problem with Time Windows - VRPTW). Στη συγκεκριμένη κατηγορία προβλημάτων, πέρα από το μέγεθος της ζήτησης του κάθε πελάτη, δίνονται στα στοιχεία του προβλήματος και τα χρονικά περιθώρια στα οποία θα μπορεί να παραλάβει την παραγγελία του. Εισάγοντας την έννοια του χρόνου στο πρόβλημα αυτό η δυσκολία επίλυσης του αυξάνεται δραματικά ενώ απαιτείται ιδιαίτερη προσοχή στην εισαγωγή της χρονικής αξίας στο προγραμματιστικό κομμάτι.

Στην γενική του μορφή παρουσιάζει δύο επιπρόσθετες κατηγορίες περιορισμών οι οποίοι εισάγουν την έννοια του χρόνου στο πρόβλημα αλλά και περιορίζουν την πραγματοποίηση των δρομολογίων εντός των δεδομένων χρονικών περιθωρίων.

$$\text{Wj,k} - \text{Wi,k} - \text{Tdi} - \text{Ti,j} \leq \text{M}\ (1 - \text{Xi,j,k}) \qquad \forall\ \text{i, j, k} \in \text{N}$$ ενώ το M ένας πολύ μεγάλος αριθμός (2.4.3)

Για την είσοδο του χρόνου σαν έννοια στο πρόβλημα, η γενική δομή που ακολουθείται είναι αυτή της σχέσης (2.4.3) που ακολουθεί παρακάτω.

Με Wj,k προσδιορίζουμε μία νέα μεταβλητή απόφασης η οποία θα εκφράζει την χρονική στιγμή όπου ξεκινά η εξυπηρέτηση του πελάτη j από το όχημα k ενώ θα ισούται με μηδέν στην περίπτωση μη εξυπηρέτησης του j από το συγκεκριμένο όχημα. Τα στοιχεία Tdi και Ti,j ορίζουν τον χρόνο

εξυπηρέτησης για να ολοκληρωθεί η παράδοση αφού φτάσει το όχημα στον πελάτη i και την χρονική απόσταση για την διαδρομή από το σημείο i στο σημείο j αντίστοιχα. Και τα δύο αυτά στοιχεία του προβλήματος αποτελούν δεδομένα του και πρέπει να υπάρχουν για την επίλυση.

Όσον αφορά την εξασφάλιση ότι ο κάθε πελάτης θα εξυπηρετηθεί εντός των δεδομένων χρονικών περιθωρίων, χρησιμοποιούνται οι περιορισμοί (2.4.4) και (2.4.5) σε γενική μορφή όπως δίνονται παρακάτω:

$$aj * \sum_{i=1}^{max} Xi,j,k \leq Wj,\ k \qquad \forall\ j,k \in N\ (1, max) \qquad (2.4.4)$$

$$Wj,\ k \leq bj * \sum_{i=1}^{max} Xi,j,k \qquad \forall\ j,k \in N\ (1, max) \qquad (2.4.5)$$

Κάθε σημείο παράδοσης j χαρακτηρίζεται από μία τιμή aj που σηματοδοτεί την αρχή του χρονικού περιθωρίου εξυπηρέτησης του καθώς και μία τιμή bj για το κλείσιμο του.

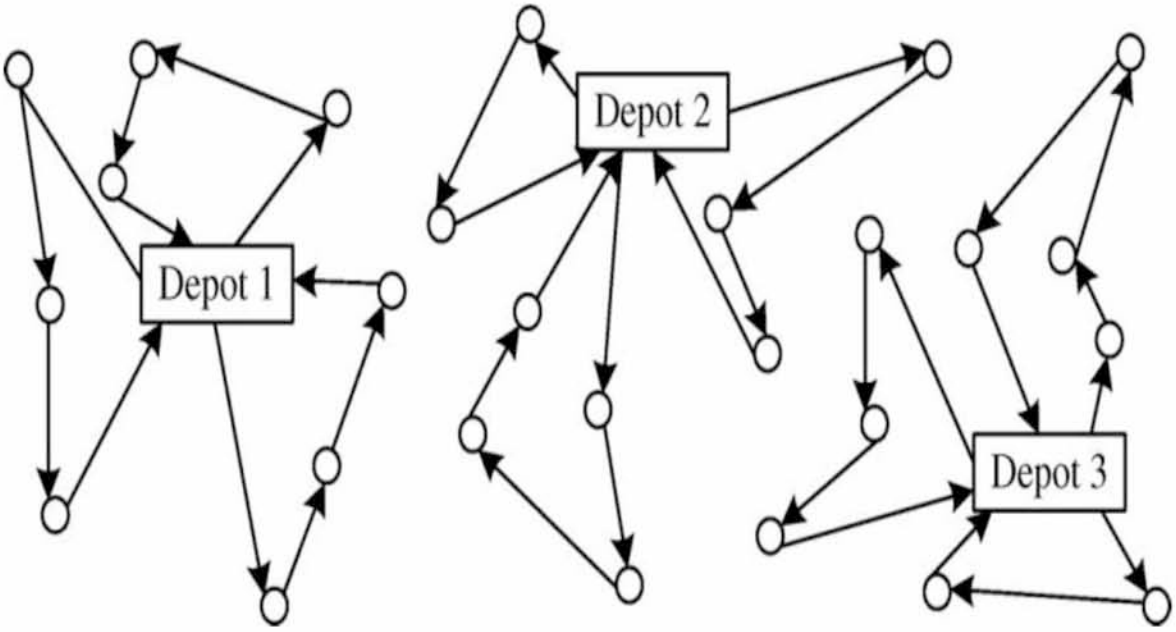
2.4.3. Πρόβλημα δρομολόγηση οχημάτων με πολλές αποθήκες (Vehicle Routing Problem with Multiple Depot – MDVRP)

Η περίπτωση του VRP με περισσότερες από μία αποθήκες αποτελεί ένα πρόβλημα που εμφανίζεται πολύ συχνά σε μεγάλες εταιρίες διανομής. Κατά την επίλυση του, ο αριθμός και η τοποθεσία των αποθηκών είναι γνωστά από την αρχή. Κάθε σταθμός αποθήκευσης πρέπει να είναι αρκετά μεγάλος ώστε να μπορεί να χωρέσει το σύνολο των παραγγελιών από όλους τους πελάτες. Εάν ένα όχημα ξεκινήσει την διαδρομή του από μία αποθήκη πρέπει υποχρεωτικά να τελειώσει το ταξίδι του σε αυτήν. Επειδή στην συγκεκριμένη περίπτωση προβλήματος υπάρχουν περισσότεροι αποθηκευτικοί χώροι για τα προϊόντα, πρέπει πριν την επίλυση του προβλήματος να ληφθεί απόφαση για την αποθήκη που θα εξυπηρετήσει τον κάθε πελάτη. Πραγματοποιείται λοιπόν μία ομαδοποίηση των πελατών αναλόγως με την απόστασή τους από κάθε βάση αλλά και τις μεταξύ τους αποστάσεις.

Αφού ολοκληρωθεί το πρώτο στάδιο της ομαδοποίησης των πελατών, αναλόγως με την αποθήκη εξυπηρέτησης, το πρόβλημα υποδιαιρείται σε μικρότερα VRP προβλήματα. Αναλόγως με το ζητούμενο σε κάθε περίπτωση μεταβάλλεται και η μορφοποίηση που ακολουθείται. Συνήθως στόχος των προβλημάτων αυτών είναι ένας συνδυασμός ελαχιστοποίησης χρόνου παράδοσης προϊόντων και ελαχιστοποίησης συνολικού κόστους της διαδικασίας.

Στα Σχήματα 2 και 3 παρουσιάζεται η διαδικασία επίλυσης ενός προβλήματος MDVRP και ένα παράδειγμα ομαδοποίησης πελατών αντίστοιχα.

Σχήμα 2. Διαδικασία επίλυσης MDVRP.



Σχήμα 3. Παράδειγμα ομαδοποίησης πελατών σε πρόβλημα MDVRP (Surekha et al., 2011).

Τέλος η ομαδοποίηση των πελατών ακολουθεί συνήθως τα εξής βήματα:

- Εάν ισχύει $D(ci, A) < D(ci, B)$, τότε ο πελάτης εξυπηρετείτε από την αποθήκη A
- Εάν ισχύει $D(ci, A) > D(ci, B)$, ο πελάτης θα εξυπηρετηθεί από την αποθήκη B

- Εάν ισχύει D(ci ,A) = D(ci ,B), τότε ο πελάτης ανατίθεται σε μία από τις αποθήκες A ή B αυθαιρέτως.

Με ci σημειώνεται ο κάθε πελάτης του προβλήματος ενώ το i παίρνει τιμές από το 1 έως το συνολικό αριθμό πελατών. Με A και B συμβολίζουμε τις δύο διαφορετικές αποθήκες που συγκρίνουμε σε κάθε περίπτωση. Τέλος, στην παραπάνω διαδικασία όπου έχουμε D(ci ,A) εννοούμε την ευθεία απόσταση του κάθε πελάτη ci από την αποθήκη A.

2.4.4. Πρόβλημα δρομολόγηση οχημάτων pick-up and delivery (VRPPD)

Στην περίπτωση του VRP με Pick Up And Delivery, ένας στόλος οχημάτων καλείτε να ικανοποιήσει τις υπάρχουσες ζητήσεις γνωρίζοντας ότι υπάρχει η πιθανότητα να γίνει επιστροφή αγαθών από τους πελάτες. Το πρόβλημα αυτό περιέχει αρκετές διασκευές ανάλογα τις συνθήκες μελέτης του. Σε αρκετές περιπτώσεις υπάρχουν δύο ή περισσότερες αποθήκες για τα αγαθά που θα επιστραφούν. Ο βαθμός δυσκολίας της επίλυσης του προβλήματος αυτού αυξάνεται αισθητά, αν αναλογιστούμε πως πλέον η μη παραβίαση της χωρητικότητας των οχημάτων απειλείται από τα επιστρεφόμενα προϊόντα. Σύμφωνα με τους Y. Dumas, J. Desrosiers, and F. Soumis.(The pickup and delivery problem with time windows. European Journal of Operational Research, 54:7-22, 1991) για να βρεθεί λύση στο πρόβλημα αυτό ακολουθούνται ένα σύνολο περιορισμών. Αρχικά, όλοι οι πελάτες θα εξυπηρετηθούν από μία φορά και από ένα μόνο όχημα. Επίσης στην πλειοψηφία τέτοιου είδους προβλημάτων δεν πραγματοποιείτε ανταλλαγή αγαθών μεταξύ των πελατών. Απαιτείτε επίσης προσοχή ώστε τα οχήματα που θα δρομολογηθούν να ξεκινήσουν και να τελειώσουν την διαδρομή τους στην επιθυμητή αποθήκη. Τέλος, μία λύση χαρακτηρίζεται εφικτή στην περίπτωση που δεν παραβιάζεται το όριο χωρητικότητας ενός οχήματος οποιαδήποτε στιγμή.

Η μορφοποίηση στην περίπτωση του VRPPD είναι όμοια με το κλασσικό πρόβλημα δρομολόγησης, με κύρια προσθήκη περιορισμού αυτήν της μη παραβίασης της του συνολικού φορτίου του κάθε K φορτηγού.

Αρχικά χαρακτηρίζουμε την παραγγελία του κάθε i πελάτη σε δύο σύνολα P και D. Στο πρώτο σύνολο περιέχονται τα στοιχεία των πελατών που αφορούν την ποσότητα αγαθού που θα επιστρέψουν και αντίστοιχα στο D σύνολο τις παραγγελίες παραλαβής . Στο σύνολο P=(1,2….n) ενώ στο δεύτερο έχω D=(n+1,n+2,…..,2n) όπου n ο αριθμός των πελατών. Το υποσύνολο A_K περιλαμβάνει όλα τα εφικτά τόξα.

Προστίθεται στο πρόβλημα μια νέα μεταβλητή απόφασης, η L_{ik} που περιέχει το συνολικό φορτίο του φορτηγού k μετά το πέρασμα του από το σημείο εξυπηρέτησης i.Με b_j περιγράφουμε την αλλαγή που θα πραγματοποιηθεί στο φορτίο του οχήματος μετά το πέρασμα από τον j πελάτη. Η τιμή αυτή μπορεί να είναι είτε θετική είτε αρνητική, καθώς σε κάποιες κατηγορίες προβλημάτων υπάρχει πιθανότητα ένας πελάτης να επιστρέφει περισσότερα αγαθά (εκφραζόμενα ως προς τον όγκο τους) από όσα είναι προγραμματισμένος να παραλάβει. Τέλος με C_k περιγράφουμε την μέγιστη χωρητικότητα του κάθε k οχήματος.

Οι επιπρόσθετοι περιορισμοί που προκύπτουν είναι οι εξής ;

$$X_{ijk}(L_{ik}+b_j - L_{jk}) = 0 \qquad \forall\ k \in K,\ i,j \in A_\kappa \qquad (2.4.6)$$

$$b_i \leq L_{ik} \leq C_k \qquad \forall\ k \in K \quad i \in P_\kappa \qquad (2.4.7)$$

$$0 \leq L_{n+i,k} \leq C_k - b_i \qquad \forall\ k \in K \quad n+i \in D_\kappa \qquad (2.4.8)$$

Λόγω του αυξημένου όγκου δεδομένων που πρέπει να ληφθεί υπόψη για την εύρεση μιας βέλτιστης λύσης, ειδικά στην περίπτωση που το πρόβλημα περιέχει και άλλες μεταβλητές (μη- ομοιογένεια του στόλου οχημάτων, χρονικά περιθώρια κ.α), γίνονται συχνά υποχωρήσεις ως προς την αυστηρότητα των περιορισμών για την απλοποίηση του. Στην περίπτωση ειδικά που υπάρχουν πολλαπλά διαθέσιμα οχήματα για εξυπηρέτηση, σύμφωνα με τους U. Derigs and A. Metz. A matching-based approach for solving a delivery /pick-up VRP with time constraints. OR-Spektrum, 14:91-106, 1992 πραγματοποιείτε συνήθως λύση με μονόπλευρα Time-Windows, ενώ συχνά συμφέρει να πραγματοποιηθούν όλες οι διαδικασίες διανομής πριν τα οχήματα ξεκινήσουν τις διαδικασίες παραλαβής. Η πλειοψηφία των εφαρμογών του VRPPD εμφανίζεται σε περιπτώσεις μελέτης μεταφοράς αγαθών μέσω πλοίων και αεροπλάνων, αλλά και μεταφοράς ανθρώπων με μέσα μαζικής μεταφοράς.

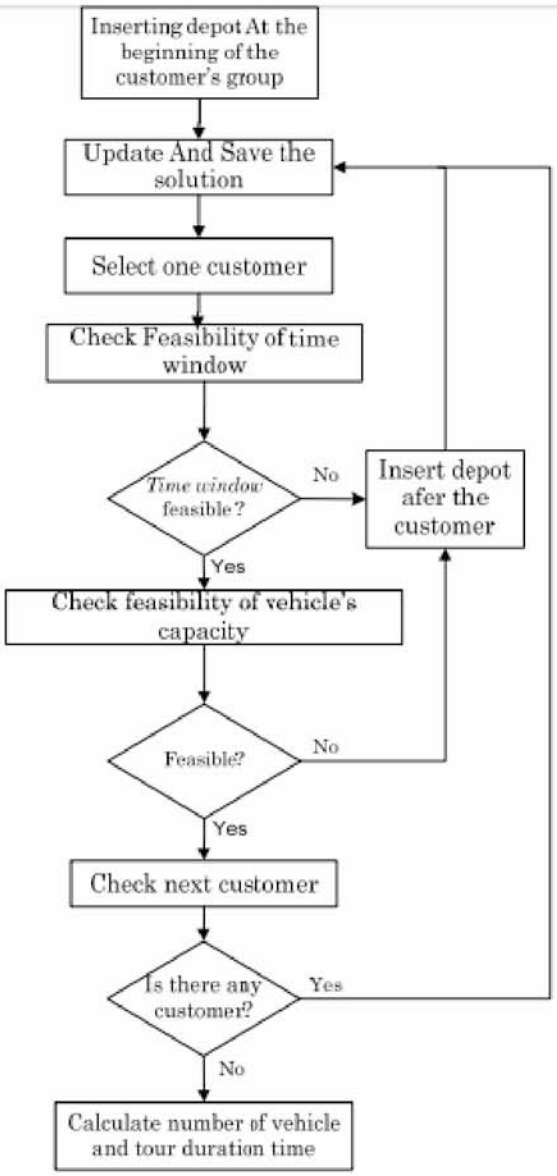
2.4.5 Το πρόβλημα δρομολόγησης οχημάτων με Backhauls (VRPB)

Η κατηγορία προβλημάτων VRPB αποτελεί μια προέκταση των κλασσικών προβλημάτων δρομολόγησης με Pick Up And Delivery που παρουσιάστηκαν στην προηγούμενη ενότητα. Όπως και στην περίπτωση του VRPPD, οι πελάτες πέρα από παραλαβή προϊόντων διαθέτουν και μία ποσότητα για επιστροφή. Η διαφορά στα δύο προβλήματα βρίσκετε στο γεγονός πως στην περίπτωση του VRPB η περισυλλογή των επιστρεφόμενων αγαθών ξεκινά αφότου ολοκληρωθούν οι διαδικασίες αποστολής. Πρόκειται δηλαδή για ένα πρόβλημα που εμφανίζεται συχνά σε περιπτώσεις που τα μεταφορικά μέσα είναι οριακά γεμάτα ή η αναδιάταξη των φορτίων κρίνεται ασύμφορη οικονομικά ή χρονικά.

Για την λύση στο πρόβλημα η εξυπηρέτηση των πελατών χαρακτηρίζεται από δύο μέρη, την ποσότητα προϊόντων που θέλει ο κάθε πελάτης να παραλάβει και την ποσότητα προϊόντων που θα επιστρέψει πίσω. Δημιουργούνται λοιπόν 2 υπό-σύνολα, αυτό των πελατών εμπρόσθιας εξυπηρέτησης (linehaul customers) και αυτό των πελατών οπίσθιας εξυπηρέτησης (backhaul customers). Το πρώτο υποσύνολο περιέχει τους πελάτες που έχουν μία ζήτηση που πρέπει να καλυφτεί από την αποθήκη, ενώ σαν «Backhaul» πελάτες χαρακτηρίζονται αυτοί που θα τους επισκεφτεί ένα όχημα για να παραλάβει μία ποσότητα αγαθών για επιστροφή.

Η ακολουθιακή εισχώρηση είναι η μέθοδος που χρησιμοποιείτε για την επίλυση τέτοιων προβλημάτων. Αυτό προκύπτει λόγω της ευκολίας στην χρήση της, την ταχύτητα που φέρνει την λύση και λόγω του ότι μεταβάλετε εύκολα για χειρισμό δύσκολων μεταβλητών. Ο αλγόριθμος της ακολουθιακής εισχώρησης χρησιμοποιείτε για την εύρεση μιας αρχικής εφικτή λύσης. Ο αλγόριθμος επιλέγει έναν linehaul πελάτη με βάση κανόνων προτεραιότητας. Αυτοί είναι το μικρότερο χρονικό παράθυρο, το μεγαλύτερο χρονικό διάστημα ταξιδιού και οι πρώτες προθεσμίες.

Ακολουθείται έπειτα η διαδικασία αποκωδικοποίησης που περιγράφεται και στο Σχήμα4.



Σχήμα 4. Vehicle Routing Problem with Backhaul, Multiple Trips and Time Window. (Johan Oscar Ong, Suprayogi). Jurnal Teknik Industri, Vol. 13, No. 1, Juni 2011, 1-10.

Μέσω της διαδικασίας αποκωδικοποίησης υπολογίζεται ο αριθμός των οχημάτων, αναλόγως των time-windows, της συνολικής χωρητικότητας των οχημάτων ή άλλων κανόνων προτεραιότητας που προέκυψαν από το πρόβλημα προς μελέτη. Από την αρχικοποίηση παίρνουμε μία ακολουθία των προς εξυπηρέτηση πελατών. Έτσι ελέγχουμε την ικανοποίηση των χρονικών και χωρικών απαιτήσεων για κάθε πελάτη. Όταν ολοκληρωθεί η εύρεση του αριθμού των οχημάτων που θα δρομολογηθούν υπολογίζεται ο χρόνος ταξιδιού και το εύρος του. Από τα αποτελέσματα που θα πάρουμε διαλέγουμε λύσεις ανάλογα τα προκαθορισμένα μας κριτήρια, όπως αριθμός οχημάτων, χρόνος διαδρομής ή το

κόστος δρομολόγησης. Σε περίπτωση που στο πρώτο κριτήριο οι διαφορές είναι μικρές, προχωράμε σε σύγκριση με το επόμενο στην σειρά σημαντικότητας.

2.4.6 Το πρόβλημα δρομολόγησης οχημάτων με περιοδικότητα(PVRP)

Το PVRP αποτελεί και αυτό μια σημαντική υποκατηγορία προβλημάτων δρομολόγησης. Εμφανίζεται συχνά σε εταιρίες που πραγματοποιούν διαδικασίες επισκευής και συντήρησης ή που συλλέγουν παραδίδουν εμπορεύματα ανά τακτά χρονικά διαστήματα. Η επίλυση του διαφέρει από το κλασσικό πρόβλημα VRP, καθώς το χρονικό διάστημα του προγραμματισμού επεκτείνετε σε έναν αριθμό ημερών. Στην περίπτωση όπου οι πελάτες του προβλήματος έχουν μία δεδομένη απαίτηση σε καθημερινή βάση, υπάρχει ταύτιση του PVRP με το απλό VRP, καθώς αρκεί η δρομολόγηση για μία ημέρα.

Το Periodic Vehicle Routing Problem αποτελεί ένα πρόβλημα σχεδιασμού ενός συνόλου ημερήσιων διαδρομών, προγραμματισμένες ώστε να ικανοποιούνται οι περιορισμού του προβλήματος και με επίτευξη ελαχιστοποίησης του συνολικού κόστους. Αποτελεί δηλαδή ένα πολύ-επίπεδο συνδυαστικό πρόβλημα βελτιστοποίησης.

Για την επίλυση του ακολουθείται η εξής διαδικασία :

- Στόχος του πρώτου βήματος είναι να δημιουργηθούν ομάδες των διαφορετικών-εφικτών συνδυασμών για κάθε πελάτη. Στην περίπτωση που η περίοδος διανομών έχει τρείς μέρες (Day1, Day2, Day3) οι συνδυασμοί εξυπηρέτησης είναι :

    a) (0,0,0)
    b) (0,0,1)
    c) (0,1,0)
    d) (0,1,1)
    e) (1,0,0)
    f) (1,0,1)
    g) (1,1,0)
    h) (1,1,1)

Όπου με 1 χαρακτηρίζεται η πραγματοποίηση διαδρομής και με 0 η μη πραγματοποίηση την αντίστοιχη μέρα. Ανάλογα τον αριθμό των επισκέψεων που πρέπει να πραγματοποιηθούνε μέσα στην περίοδο εξυπηρέτησης, προκύπτει ο ακόλουθος πίνακας:

Πελάτης	Αριθμός επισκέψεων	Πλήθος συνδυασμών	Πιθανοί συνδυασμοί
1	1	3	b, c, e
2	2	3	d, f, g
3	3	1	h

- Στο δεύτερο βήμα της διαδικασίας επίλυσης πρέπει να επιλεγεί μία από τις εναλλακτικές λύσεις που βρέθηκαν στο προηγούμενο βήμα, με σεβασμό στους περιορισμούς που έχουν τεθεί για τις διαδρομές. Συνεπώς θα πρέπει να γίνει η επιλογή των πελατών που θα δεχτούν επίσκεψη την κάθε μία μέρα της περιόδου.
- Στο τελευταίο βήμα πραγματοποιείτε επίλυση του απλού προβλήματος δρομολόγησης VRP για κάθε μέρα της περιόδου.

Με την επίλυση των προβλημάτων PVRP επιτυγχάνεται ο στόχος της ελαχιστοποίησης των δρομολογηθέντων οχημάτων σε συνδυασμό με ελάχιστο χρόνο ταξιδιού. Μία λύση θα θεωρηθεί εφικτή εάν καλύπτει το σύνολο των περιορισμών του κλασσικού προβλήματος VRP. Στην συγκεκριμένη όμως περίπτωση ένα όχημα έχει την δυνατότητα να μην γυρίσει πίσω στην αποθήκη την ημέρα που ξεκίνησε την διαδρομή του. Επίσης, κατά την διάρκεια μιας περιόδου πρέπει όλοι οι πελάτες να εξυπηρετηθούν τουλάχιστον μία φορά.

2.4.7 Το στοχαστικό πρόβλημα δρομολόγησης οχημάτων (SVRP)

Σαν στοχαστικό (πιθανολογικό) πρόβλημα δρομολόγησης, χαρακτηρίζεται ένα πρόβλημα VRP στο οποίο ένα ή περισσότερα στοιχεία του θεωρούνται μη σταθερά, καθώς γίνεται να μεταβληθούν ανά πάσα στιγμή. Με βάση το στοχαστικό τους χαρακτηριστικό διαχωρίζονται στις εξής κατηγορίες :

- Στοχαστικοί πελάτες.
- Στοχαστικές απαιτήσεις πελατών.
- Στοχαστικός χρόνος εξυπηρέτησης ταξιδιού.

Στην πρώτη περίπτωση ο κάθε i πελάτης έχει μία πιθανότητα να πραγματοποιήσει παραγγελία ίση με p_i,και αντίστοιχα πιθανότητα να μην πραγματοποιήσει παραγγελία ίση με 1-p_i .Στη δεύτερη περίπτωση η ζήτηση του κάθε πελάτη i έχει τυχαία μεταβλητή τιμή d_i και δεν είναι σταθερή. Τέλος στην Τρίτη περίπτωση ο χρόνο που χρειάζεται ένα όχημα για να πραγματοποιήσει ένα δρομολόγιο TR_{ij}ή για να εξυπηρετήσει κάποιον πελάτη S_i είναι μεταβλητός.

Για να βρεθεί μία εφικτή λύση που πλησιάζει τη βέλτιστη, σε προβλήματα SVRP, υπάρχουν δύο βήματα. Στο πρώτο βήμα επιλέγεται μία πρώτη λύση χωρίς να λαμβάνονται υπόψη οι τυχαίες μεταβλητές. Έπειτα, γνωρίζοντας τις αλλαγές στις μεταβλητές πλέον, ακολουθούν κάποια επιδιορθωτικά μέτρα πάνω στην αρχική λύση.

Η αντικειμενική συνάρτηση που εκφράζει το γενικό στοχαστικό πρόβλημα δρομολόγησης έχει τη μορφή $\sum_{i\leq j} c_{ij}X_{ij} + Q_x$ με σκοπό την ελαχιστοποίηση της. Με X_{ij} ορίζεται η ακέραιη μεταβλητή που περιέχει την χρονική στιγμή εμφάνισης των κόμβων. Εάν i,j ≥1 τότε η μεταβλητή X_{ij} μπορεί να πάρει μόνο τιμές 1 ή 0, ενώ για i=1 παίρνει την τιμή 2 εάν το όχημα κινείται στον κόμβο u_i από την αποθήκη.

Με το Q_x χαρακτηρίζουμε την είσοδο στο δεύτερο βήμα της επίλυσης. Το πρόβλημα πλέον είναι εξαρτώμενο στις συνθήκες των αλλαγών που πραγματοποιούνται. Παραδείγματα σε ένα SVRP με περιορισμένη χωρητικότητα, πιθανές ενέργειες προσφυγής είναι οι εξής :

- Η πραγματοποίηση επιστροφής ενός πλήρους οχήματος στον αποθηκευτικό χώρο για να επιστρέψει τις παραλαβές και στην συνέχεια να γυρίσει στην συλλογή των υπόλοιπων προϊόντων ακολουθώντας το πρόγραμμα.
- Επιστροφή ενός οχήματος στην αποθήκη όπως και πριν, πραγματοποιώντας όμως επανα-βελτιστοποίηση της υπόλοιπης διαδρομής.
- Ένα όχημα που δεν είναι ακόμα πλήρες να υποχρεωθεί να επιστρέψει στην αποθήκη μόνο εάν είναι γνωστό το γεγονός πως η εξυπηρέτηση επόμενου πελάτη θα το ανάγκαζε να ξεπεράσει την χωρητικότητα του.
- Η πραγματοποίηση κάποιων προληπτικών επιστροφών στην αποθήκη, παρόλο που το όχημα μπορεί να μην είναι πλήρες. Στη συγκεκριμένη περίπτωση η απόφαση θα μπορεί να παρθεί από την απόσταση που βρίσκεται το όχημα από την βάση ή το μέγεθος των παραλαβών που έχει πραγματοποιήσει.

Επειδή κάποια στοιχεία και τιμές του προβλήματος μεταβάλλονται τυχαία, δεν είναι δυνατόν να ισχύουν όλοι οι περιορισμοί του προβλήματος για όλο το σύνολο των πιθανών τιμών αυτών. Έτσι πρέπει να παρθεί απόφαση εάν θα γίνει ικανοποίηση κάποιων από τους περιορισμούς με δεδομένη πιθανότητα ή πραγματοποίηση διορθωτικών αλλαγών σε περίπτωση που κάποιος από τους περιορισμούς δεν τηρείται.

2.4.8 Άλλες ειδικευμένες περιπτώσεις προβλημάτων VRP

Κάποια άλλα προβλήματα δρομολόγησης που εμφανίζονται σε συγκεκριμένες περιπτώσεις και χαρακτηρίζονται από κάποιες ιδιομορφίες στην δομή τους ή στην δομή των περιορισμών που προκύπτουν είναι τα εξής :

- **Heterogeneous Fleet VRP.** Στην περίπτωση αυτήν ο στόλος των οχημάτων που μπορούν να αξιοποιηθούν για μεταφορές δεν χαρακτηρίζεται από ομοιογένεια. Αυτό σημαίνει πως μπορεί να υπάρχουν κατηγορίες ανάλογα το μέγεθος τους ή το είδος τους, πχ μεγάλου μεγέθους, μεσαίου και μικρού ή ακόμα και κάθε όχημα να έχει διαφορετική χωρητικότητα από τα άλλα.

- **Open VRP.** Το συγκεκριμένο πρόβλημα μοιάζει αρκετά με το κλασσικό πρόβλημα δρομολόγησης, με την διαφορά ότι τα οχήματα τελειώνουν το δρομολόγιο τους σε κάποιο από τα σημεία εξυπηρέτησης και όχι στην αρχική αποθήκη. Εφαρμόζεται κυρίως όταν οι προμηθευτές δεν διαθέτουν στόλο οχημάτων και προτιμούν να αναθέσουν σε εξωτερικούς συνεργάτες τα θέματα μεταφοράς και διανομής τους.

- **Site-Dependent Vehicle Routing Problem.** Στο πρόβλημα αυτό, ένας στόλος ετερογενών οχημάτων εξυπηρετεί ένα σύνολο πελατών. Συνήθως, υπάρχουν διάφοροι τύποι οχημάτων (για παράδειγμα, οχήματα μικρού, μέσου και μεγάλου δυναμικού). Επιπλέον, υπάρχει σχέση συμβατότητας μεταξύ των πελατών και των οχημάτων. Για παράδειγμα, ορισμένοι πελάτες που βρίσκονται σε αστικές περιοχές με συμφόρηση μπορούν να εξυπηρετηθούν μόνο από οχήματα μικρής χωρητικότητας, ενώ ορισμένοι πελάτες που βρίσκονται σε προαστιακές περιοχές μπορούν να εξυπηρετούνται από οποιοδήποτε τύπο οχήματος.

- **LIFO VRP.** Αρχικά, ο όρος LIFO σημαίνει Last-In-Frist-Out (ότι εισέρχεται τελευταίο δηλαδή εξέρχεται πρώτο). Το πρόβλημα αυτό μοιάζει αρκετά με το πρόβλημα δρομολόγησης με συλλογές κατά τις παραδόσεις (VRPPD). Σε κάθε σημείο διανομής τα προϊόντα που παραδίδονται πρώτα στον πελάτη είναι αυτά που παραλήφθηκαν πιο πρόσφατα από την αποθήκη. Έτσι, επιτυγχάνεται μείωση των χρόνων που απαιτούνται για την εκφόρτωση και την φόρτωση στα σημεία διανομής, καθώς υπάρχει άμεση πρόσβαση εύκολα στα προϊόντα που πρέπει να παραδοθούν.

- **VRP with Satellite Facilities.** Μια σημαντική πτυχή του προβλήματος δρομολόγησης οχημάτων (VRP) που έχει παραβλεφθεί σε μεγάλο βαθμό είναι η χρήση δορυφορικών εγκαταστάσεων για

την ανασύσταση των οχημάτων κατά τη διάρκεια μιας διαδρομής. Όταν είναι δυνατόν, επικοινωνία οχημάτων με αποθήκη μέσω δορυφόρου επιτρέπει στους οδηγούς να συνεχίσουν τις παραδόσεις μέχρι το κλείσιμο της διαδρομής τους χωρίς να υποχρεωθούν να επιστρέψουν στην κεντρική αποθήκη. Αυτή η περίπτωση εμφανίζεται συχνά σε περιπτώσεις διανομής καυσίμων και συγκεκριμένων ειδών λιανικής.

2.5 Μελέτη δοθέντος προβλήματος και μεθοδολογία

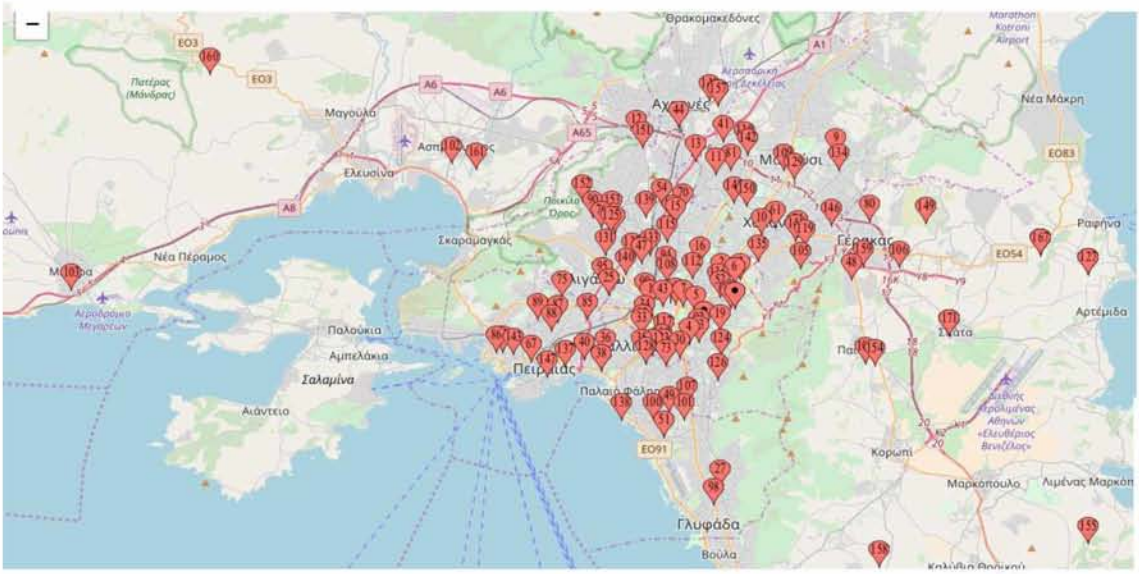
Όπως είδαμε στο κεφάλαιο αυτό, αναλόγως τους περιορισμούς που πρέπει να καλυφθούν και των χαρακτηριστικών των δεδομένων του, ένα πρόβλημα VRP μπορεί να μπει σε υπάρχουσες κατηγορίες και με γνωστές διαδικασίες έπειτα να επιλυθεί. Στα πλαίσια αυτής της διπλωματικής εργασίας, πραγματοποιήθηκε μελέτη ενός πραγματικού προβλήματος δρομολόγησης. Πιο συγκεκριμένα, στο κεφάλαιο που ακολουθεί θα μελετηθεί ολοκληρωτικά μία περίπτωση διανομής προϊόντων με την χρήση ενός ετερογενούς στόλου περιορισμένης χωρητικότητας ενώ όλοι οι πελάτες έχουν συγκεκριμένα χρονικά περιθώρια εξυπηρέτησης.

ΚΕΦΑΛΑΙΟ 3: ΠΑΡΟΥΣΙΑΣΗ, ΜΟΡΦΟΠΟΙΗΣΗ ΚΑΙ ΑΡΧΙΚΗ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ CVRP-TW

3.1. Γενικά χαρακτηριστικά του προβλήματος

Στα πλαίσια αυτής της εργασίας το πρόβλημα που μελετάται και επιλύεται είναι μια πραγματική περίπτωση Προβλήματος Δρομολόγησης Οχημάτων με Περιορισμένη Χωρητικότητα στα Οχήματα, χρήση Ετερογενούς Στόλου Οχημάτων και περιορισμό στα Χρονικά Περιθώρια εξυπηρέτησης (CVRP-TW with Heterogeneous Fleet). Πιο συγκεκριμένα, γίνεται επιλογή οχημάτων και διαδρομών για την κατά το δυνατό βέλτιστη διανομή των παραγγελιών ενός πλήθους πελατών. Στο κεφάλαιο αυτό θα αναλυθεί η διαδικασία που ακολουθήθηκε για την εύρεση των διαδρομών που θα πραγματοποιηθούν για να επιτευχθεί ο ελάχιστος χρόνο κάλυψης όλων των απαιτήσεων. Ειδικότερα:

- Οι πελάτες βρίσκονται σε διάσπαρτα σημεία στην περιοχή της Αττικής (βλέπε Σχήμα 2.1).
- Για το κάθε ένα είναι γνωστή η ζήτησή του σε αγαθά σε μονάδες όγκου και βάρους.
- Κάθε πελάτης χαρακτηρίζεται από ένα χρονικό παράθυρο εξυπηρέτησης, μέσα στο οποίο θα δέχεται επίσκεψη για την παραλαβή της παραγγελίας του.
- Οι χρόνοι ταξιδιού από τον έναν πελάτη στον άλλο είναι γνωστοί και θεωρούνται σταθεροί.
- Τέλος, δίνεται ένας στόλος οχημάτων προς αξιοποίηση, με το κάθε όχημα να έχει διαφορετική μέγιστη χωρητικότητα προϊόντων ως προς βάρος και όγκο από τα υπόλοιπα.



Σχήμα 3.1 . Οι θέσεις των πελατών στη περιοχή της Αττικής.

3.2. Δεδομένα προς αξιοποίηση για την επίλυση του προβλήματος

Μετά την αναγνώριση του είδους του προβλήματος, σειρά έχει η συλλογή των στοιχείων που θα αξιοποιηθούν. Τα συγκεκριμένα δεδομένα μπορούν να χωριστούν σε δύο κατηγορίες: αυτά των πελατών και αυτά των φορτηγών.

Τα στοιχεία των πελατών:

- Η ζήτηση του προσφερόμενου προϊόντος του κάθε πελάτη (σε βάρος και όγκο).
- Όλες οι αποστάσεις (σε χρόνο) για την μετάβαση από ένα σημείο διανομής σε οποιοδήποτε άλλο (καθώς και με τη βάση των φορτηγών του στόλου).
- Τα χρονικά περιθώρια εξυπηρέτησης του κάθε πελάτη.

Τα στοιχεία των φορτηγών:

- Χωρητικότητα του κάθε φορτηγού σε βάρος (Kg).
- Χωρητικότητα του κάθε φορτηγού σε όγκο (m^3).

3.2.1. Τα στοιχεία των πελατών

Όπως σε όλα τα προβλήματα VRP, για να επιλυθεί το πρόβλημα θα πρέπει μετά την ολοκλήρωση των προγραμματισμένων δρομολογίων ο κάθε πελάτης να έχει παραλάβει εξ ολοκλήρου τα προϊόντα που έχει ζητήσει. Στο πρόβλημα δόθηκε ένας πίνακας διαστάσεων 3 X 171 που περιέχει για τον κάθε πελάτη τη παραγγελία του, μεταφρασμένη σε μονάδες όγκου και βάρους που καταλαμβάνουν για την μεταφορά τους. Επειδή πρόκειται για ένα πρόβλημα δρομολόγησης με οχήματα περιορισμένης χωρητικότητας, τα δεδομένα των ζητήσεων επηρεάζουν άμεσα και σημαντικά την επίλυση. Στο Πίνακα 3.1 που ακολουθεί δίνεται ένα παράδειγμα των ζητήσεων των δέκα πρώτων σε αρίθμηση πελατών.

Πίνακας 3.1. Οι ζητήσεις των δέκα πρώτων σε αρίθμηση πελατών.

Αριθμός Πελάτη (ID)	Παραγγελία σε κιλά(kg)	Παραγγελία σε όγκο (m^3).
1	2098	102
2	518	20
3	450	47
4	141	13
5	129	12
6	118	10
7	297	25
8	1055	60
9	1228	67
10	533	28

Επόμενο στοιχείο των πελατών είναι οι χρονικές και χωρικές αποστάσεις μεταξύ τους αλλά και από την αποθήκη. Για να επιλυθεί ένα τέτοιο πρόβλημα πρέπει να γνωρίζουμε το κόστος των δρομολογίων, εκφρασμένο σε μονάδες χρόνου ή μέτρων. Επειδή ο σκοπός του προβλήματος αυτού είναι η ελαχιστοποίηση του συνολικού χρόνου που απαιτείται για την ολοκλήρωση των διαδρομών, δίνεται μεγαλύτερη έμφαση στις χρονικές αποστάσεις. Στους Πίνακες 3.2 και 3.3 παρακάτω παρουσιάζεται ένα δείγμα των δεδομένων αυτών για τους πρώτους 10 σε αρίθμηση πελάτες, καθώς και η απόστασή τους από την αποθήκη (σημείο 0). Στους πίνακες που ακολουθούν σε κάθε τετράγωνο περιέχεται η απόσταση από το σημείο i στο σημείο j.

Πίνακας 3.2. Αποστάσεις σημείων εκφρασμένες σε μονάδες χρόνου (Seconds).

i/j	0	1	2	3	4	5	6	7	8	9	10
0	0	1500	1860	1800	2220	2040	1860	2040	1740	1740	1320
1	1560	0	900	1080	840	660	1020	720	420	1920	1500
2	1860	1020	0	240	900	480	240	780	660	1740	720
3	1920	1200	240	0	960	660	60	960	840	1680	600
4	2280	900	960	1020	0	600	960	840	720	2280	1500
5	1980	720	660	780	600	0	720	420	360	2280	1260
6	1920	1200	240	60	900	600	0	960	780	1680	600
7	1980	720	780	960	840	540	900	0	360	2340	1380
8	1680	420	660	840	660	420	780	420	0	2040	1320
9	1740	1860	1740	1680	2160	2160	1680	2400	2100	0	1140
10	1560	1560	600	480	1320	960	540	1320	1200	1320	0

Πίνακας 3.3. Αποστάσεις σημείων εκφρασμένες σε μέτρα (Meters).

i/j	0	1	2	3	4	5	6	7	8	9	10
0	0	24286	22707	23721	34905	24675	24223	23667	23102	23861	20454
1	24618	0	5105	6343	4540	3580	5951	2782	2217	23143	19736
2	23631	5651	0	1575	5039	2738	1514	3330	3652	11312	4636
3	24294	7106	1656	0	5653	3836	505	4688	5010	10797	4121
4	26421	4815	5166	6010	0	2838	5676	3262	3379	19407	9404
5	24626	3831	3715	4351	3154	0	3959	1164	1794	14378	7702
6	24429	6654	1426	415	5324	3322	0	4156	4478	10932	4256
7	23578	2783	3126	4274	3693	1939	3882	0	746	22103	7603
8	23178	2325	3636	4872	3693	2277	4480	1117	0	21703	8173
9	23965	23253	11412	10655	19291	13592	11157	22634	22069	0	7388
10	21835	9472	4025	3268	7968	6205	3770	7039	7361	8338	0

Τέλος για το κάθε πελάτη πρέπει να γνωρίζουμε τα χρονικά περιθώρια μέσα στην ημέρα που θα δέχεται τις επισκέψεις αλλά και το χρόνο που θα χρειαστεί να δαπανήσει το όχημα σε αυτόν για τη διαδικασία εκφόρτωσης. Αυτά δίνονται αντίστοιχα για τους πρώτους 10 πελάτες στο Πίνακα 3.3. Στον πίνακα αυτόν δίνονται οι χρόνοι σε λεπτά, και επειδή η δρομολόγηση θα πραγματοποιηθεί εντός μίας ημέρας εκφράζουν το διάστημα μεταξύ ποιων ωρών μέσα στην ημέρα θα είναι διαθέσιμοι οι πελάτες για παραλαβή.

Πίνακας 3.4. Χρονικά Παράθυρα και Χρόνοι Εξυπηρέτησης πελατών.

Πελάτης	Αρχή Παραθύρου	Τέλος Παραθύρου	Χρόνος εκφόρτωσης
1	480	840	72
2	480	870	25
3	480	810	23
4	480	810	14
5	480	810	13
6	480	810	13
7	480	810	18
8	480	870	41
9	480	930	46
10	0	1440	26

3.2.2. Τα στοιχεία των οχημάτων

Για τον στόλο των φορτηγών που είναι διαθέσιμα για δρομολόγηση, στο συγκεκριμένο πρόβλημα, δόθηκαν οι χωρητικότητές τους σε βάρος και όγκο που μπορούν να μεταφέρουν όπως παρουσιάζονται στον Πίνακα 3.4 για τα πρώτα 8 φορτηγά του στόλου.

Πίνακας 3.5 Μέγιστες χωρητικότητες οχημάτων στόλου.

Φορτηγό	Μέγιστο Βάρος Μεταφοράς	Μέγιστος Όγκος Μεταφοράς
1	3655	250
2	1470	200
3	3290	300
4	3880	300
5	3840	300
6	3760	300
7	3680	300
8	3360	300

3.3. Μορφοποίηση του Μαθηματικού Μοντέλου για την επίλυση του προβλήματος

Για να επιλυθεί το πρόβλημα πρέπει πρώτα να εκφραστεί στη μορφή ενός μοντέλου γραμμικού προγραμματισμού. Για να γίνει αυτό, θα μεταφραστούν κάποια στοιχεία και δεδομένα σε νέα μαθηματική μορφή. Έτσι, πλέον έχουμε τα εξής :

- **Qai** θα εκφράζει την ζήτηση του σημείου i σε αγαθά μετρημένο σε Kg (i £ N , i ∈ (1, 171)).
- **Qbi** με την σειρά του την αντίστοιχη ζήτησε του εκάστοτε σημείου i μετρημένο σε m^3 (i £ N , i ∈ (1, 171)).
- **(ai , bi)** το χρονικό περιθώριο εξυπηρέτησης του κάθε πελάτη i (i £ N , i ∈ (1 , 171)).
- **Ti,j** τον απαιτούμενο χρόνο για την μεταφορά από ένα σημείο διανομής i σε ένα σημείο διανομής j (i, j £ N , i, j ∈ (0 , 171)) όπου για τιμή 0 συμβολίζεται το depot.
- **Tdi** τον χρόνο που χρειάζεται ο κάθε πελάτης i για να παραλάβει την παραγγελία του αφότου φτάσει το φορτηγό σε αυτόν (i £ N , i ∈ (1 , 171)).
- **CAP1k** την μέγιστη χωρητικότητα του φορτηγού k σε kg . (k ∈ N , k ∈ (1, 105))
- **CAP2k** την μέγιστη χωρητικότητα του φορτηγού k σε m^3. (k ∈ N , k ∈ (1, 105))

Για να μπορέσει το πρόβλημα να μορφοποιηθεί γίνεται χρήση τεσσάρων μεταβλητών απόφασης που θα καθορίσουν αλλά και θα εκφράσουν την λύση του προβλήματος. Οι μεταβλητές αυτές είναι οι παρακάτω και εκφράζουν τα εξής:

- **Xi,j,k:** Δυική 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 οχήματα. Στην περίπτωση όπου ο δείκτης αυτός παίρνει την τιμή 1 αυτό σημαίνει πως το όχημα k εκτελεί δρομολόγιο από τον πελάτη i στον πελάτη j και 0 εάν δεν εκτελείται αυτός ο συνδυασμός δρομολογίου-οχήματος.
- **Wi,k**: Η μεταβλητή αυτή περιέχει ακέραιες τιμές μεταξύ των αριθμών 0 και 1440 και περιέχει τις χρονικές τιμές που το αντίστοιχο όχημα k ξεκινά την εξυπηρέτηση στον πελάτη i (εκφρασμένο σε λεπτά). Όταν το όχημα k δεν εξυπηρετεί τον i τότε παίρνει την τιμή 0.
- **TWijk**: Πρόκειται για βοηθητική μεταβλητή της οποίας η ανάγκη εμφανίστηκε κατά την διαδικασία επίλυσης και σχετίζεται με την φύση του προβλήματος με Time-Windows. Σε κάποιες λύσεις τα οχήματα μπορεί να φτάσουν στον προορισμό τους πριν το «άνοιγμα» των επιτρεπόμενων χρονικών περιορισμών και συνεπώς αναγκάζεται να περιμένει εκεί. Αυτός ο χρόνος αναμονής όμως είναι απαραίτητος για το πρόβλημα και επηρεάζει την λύση του.

- **Yi:** Πρόκειται για μια βοηθητική μεταβλητή της οποίας η χρήση είναι στην επίλυση του γνωστού προβλήματος sub-touring που εμφανίζεται σε προβλήματα VRP. Η χρήση της γίνεται στον αλγόριθμο των Miller-Tucker-Zemlin (1960) και παρουσιάζεται παρακάτω.

Για να βρεθεί η βέλτιστη επιλογή οχημάτων που θα χρησιμοποιηθούν και τα αντίστοιχα δρομολόγια που θα τους ανατεθούν θα πρέπει να περιοριστεί ο χώρος των λύσεων που επιτρέπεται μέσω περιορισμών. Στα προβλήματα αυτά δοκιμάζονται όλοι οι πιθανοί συνδυασμοί εφικτών λύσεων κρατώντας πάντα σαν μέτρο σύγκρισης την πιο αποδοτική με βάση το ζητούμενο της μελέτης. Οι περιορισμοί που αναλύονται στην πορεία προκύπτουν είτε από τη φύση του προβλήματος και των συνθηκών που αντιμετωπίστηκαν είτε για προγραμματιστικούς λόγους.

1) $\sum_{i=0}^{imax} \sum_{k=1}^{kmax} Xi,j,k = 1$ $\forall$ j ∈ N (1, jmax)

2) $\sum_{j=1}^{jmax} Xo,j,k \leq 1$ $\forall$ k ∈ N (1, kmax)

3) $\sum_{i=0}^{imax} Xihk - \sum_{j=0}^{jmax} Xhjk = 0$ $\forall$ k,h , h,k ∈ N (1, max)

4) $Wj,k - Wi,k - Tdi - Ti,j \leq M(1 - Xi,j,k)$ $\forall$ i, j, k ∈ N και M πολύ μεγάλοι αριθμοί

5) $TWi,j,k \geq Wj,k - Ti,j - Wi,k - Tdi - M * (1 - Xijk)$ $\forall$ i, j, k ∈ N (1, max)

6) $aj * \sum_{i=1}^{max} Xi,j,k \leq Wj,k$ $\forall$ j ,k ∈ N (1, max)

$Wj,k \leq bj * \sum_{i=1}^{max} Xi,j,k$

7) $\sum_{j=0}^{jmax} Wj,k \leq M * \sum_{i=0}^{imax} \sum_{j=0}^{jmax} Xi,j,k$ $\forall$ k

8) $\sum_{i=0}^{171}\{\sum_{j=0}^{171} Qai * Xi,j,k\} \leq CAP1k$ $\forall$ k

$\sum_{i=0}^{171}\{Qbi * \sum_{j=0}^{171} Xi,j,k\} \leq CAP2k$ $\forall$ k

9) $\sum_{i=0}^{171} \sum_{j=0}^{171}\{Xi,j,k * [Ti,j + Tdi] + TWi,j,k]\} \leq 480$ $\forall$ k ∈ N (εκφρασμένο σε λεπτά εργασίας)

10) $\sum_{i=1}^{imax} \sum_{j=0}^{jmax} Xijk \leq M * \sum_{h=1}^{hmax} X0,h,k$ $\forall$ k ∈ N

11) $Xijk = 0$ $\forall$ i, k ∈ (0, max)

12) $Yi - Yj + (N - 1) * Xi,j,k \leq N - 2$ $\forall$ i≠j ∈ N (1,imax) , k ∈ N (1, kmax)

Τέλος, η αντικειμενική συνάρτηση του προβλήματος αυτού είναι η εξής:

$$\text{Minimize (Z)} = \sum_{i=0}^{imax} \sum_{j=0}^{jmax} \sum_{k=1}^{kmax} Xi,j,k * [Ti,j + Tdj] + TWi,j,k$$

Ανάλυση των περιορισμών:

- Η πρώτη ομάδα περιορισμών του προβλήματος (περιορισμοί 1, 2 και 3) εξασφαλίζουν την δρομολόγηση του στόλου οχημάτων της εταιρείας. Πιο συγκεκριμένα, ο $1^{ος}$ περιορισμός εξασφαλίζει ότι όλα τα σημεία θα εξυπηρετηθούν 1 φορά ακριβώς από ένα όχημα. Ο δεύτερος περιορισμός αναγκάζει τα οχήματα που τελικά θα δρομολογηθούν, να φύγουν από την βάση (depot) το πολύ μία φορά. Έπειτα, ο $3^{ος}$ περιορισμός του πακέτου δρομολόγησης εξασφαλίζει πως τα επιλεγμένα οχήματα για διανομή προϊόντων θα επιστρέψουν στην βάση μετά το τέλος του δρομολογίου τους.

- Στη δεύτερη ομάδα (περιορισμοί 4, 5, 6 και 7) δίνεται η χρονική αξία του προβλήματος. Πιο συγκεκριμένα, με τον $4^{ο}$ περιορισμό, δίνεται η χρονική αλληλουχία των τιμών της μεταβλητής Wi,k, δηλαδή πως ο χρόνος εξυπηρέτησης για παράδειγμα του $3^{ου}$ εξυπηρετούμενου πελάτη από το φορτηγό k πρέπει να είναι μεγαλύτερος από αυτόν του $2^{ου}$ εξυπηρετούμενου πελάτη από το ίδιο φορτηγό, αλλά και την σχέση του με τις τιμές των χρόνων μεταφοράς και εξυπηρέτησης. Με τον $5^{ο}$ περιορισμό, εισάγεται η έννοια του χρόνου αναμονής (wait time), που μπορεί να έχει κάποιο όχημα εάν φτάσει στον προορισμό του πριν τον επιτρεπόμενο χρόνο. Ο $6^{ος}$ περιορισμός περνάει στο πρόβλημα τα δοθέντα χρονικά περιθώρια εξυπηρέτησης του κάθε πελάτη. Τέλος, ο $7^{ος}$ περιορισμός, με την σειρά του περιορίζει την μεταβλητή Wi,k, ενώ μηδενίζει την τιμή της στην περίπτωση όπου ο πελάτης i δεν εξυπηρετείται από το όχημα k.

- Από τα δεδομένα του προβλήματος, λόγω της περιορισμένης χωρητικότητας του κάθε οχήματος και την ανάγκη εργασίας μέχρι 8 ώρες για κάθε οδηγό, προκύπτουν αντιστοίχως οι περιορισμοί 8 και 9, όπου βεβαιώνουν την τήρηση των αντίστοιχων αναγκών για ορισμένη χωρητικότητα που δεν παραβιάζεται αλλά και απαραβίαστο 8-ωρο.

- Τέλος, η χρήση των περιορισμών 10, 11 και 12 προέκυψαν στην διαδικασία επίλυσης του προβλήματος, και σχετίζονται με τεχνικές δυσκολίες που εμφανίζει η επίλυση του προβλήματος αυτού στον υπολογιστή. Ο πρώτος από αυτούς εξασφαλίζει πως κάθε διαδρομή (route) που θα εκτελεστεί, θα ξεκινήσει από την βάση. Η ανάγκη για τον περιορισμό 11 προέκυψε λόγω της εμφάνισης αυτό-δρομολογίων (self-route) κατά την επίλυση (από το σημείο 1 στο 1). Ο τελευταίος περιορισμός που χρησιμοποιήθηκε είναι αυτός της επίλυσης των υπο-διαδρομών (sub-tour elimination constraint). Για την επίλυσή του χρησιμοποιήθηκε ο αλγόριθμος των Miller-Tucker-Zemlin (1960).

3.4. Πρώτη επίλυση του προβλήματος με χρήση κατάλληλου λογισμικού

Μετά την σωστή μορφοποίηση του προβλήματος, σειρά έχει η επιλογή κατάλληλου λογισμικού για να περαστούν τα στοιχεία του προβλήματος αλλά και να γίνει η επίλυση του. Στα πλαίσια της εργασίας αυτής επιλέχτηκε η χρήση της γλώσσας C++ με βιβλιοθήκες της c-plex. Η υλοποίηση των προγραμμάτων έλαβε χώρα σε προσωπικό υπολογιστή. Τα χαρακτηριστικά του υπολογιστή που χρησιμοποιήθηκε είναι τα εξής:

- Επεξεργαστής: Intel(R) Core I3, 3.00GHz,
- Εγκατεστημένη μνήμη: 4 GB,
- Λογισμικό: Windows 7 Ultimate 64-bit.

Με την μορφοποίηση του προβλήματος σαν ένα πρόβλημα προγραμματισμού βελτιστοποίησης δίνεται η εντολή στον υπολογιστή να μελετήσει όλες τις πιθανές εφικτές λύσεις και να τις συγκρίνει με την πιο αποδοτική που έχει βρεθεί. Εάν ένας νέος συνδυασμός πελατών, δρομολογίων και οχημάτων δίνει μία πιο επιθυμητή λύση τότε αυτή σώζεται και παίρνει τον ρόλο της νέας λύσης με την οποία θα συγκριθούν οι υπόλοιπες. Η διαδικασία αυτή συνεχίζεται μέχρι να ελεγχθούν όλες οι πιθανές περιπτώσεις συνδυασμών δρομολογίων, πελατών ανά δρομολόγιο και οχημάτων που θα εξυπηρετήσουν. Λόγω του όγκου και της φύσης του προβλήματος και του τεράστιου υπολογιστικού χώρου που απαιτείται, το πρόγραμμα μελετήθηκε και λύθηκε με χρήση υπό-προβλημάτων σε πρώτο στάδιο. Αυτό συμβαίνει εξαιτίας του υπερβολικά μεγάλου αριθμού (πρακτικά άπειρων) πιθανών πεδίων λύσεων όπου πρέπει να μελετηθεί για να καταλήξουμε στη βέλτιστη. Για κάθε έναν πελάτη που προστίθεται στο πρόβλημα, η διάσταση του πεδίου λύσεων αυξάνεται εκθετικά. Το ίδιο προφανώς συμβαίνει και σε μεγαλύτερο βαθμό για τον αριθμό των πιθανών οχημάτων διανομής καθώς οι χωρητικότητες του στόλου οχημάτων διαφέρουν για το καθένα.

Για να επιτευχθεί μια λογική προσέγγιση του προβλήματος έγινε μια κατηγοριοποίηση των δεδομένων των φορτηγών αλλά και των πελατών σε 13 υπό-ομάδες και ακολούθησε η λύση των προβλημάτων αυτών. Θυσιάστηκε ένα κομμάτι «βελτιστότητας» της λύσης για την εύρεση μιας (αρχικής) εφικτής λύσης. Για να γίνει αυτό απαιτείται αρχικά ο ποιοτικός διαχωρισμός των φορτηγών σε ομάδες (clusters). Είναι απαραίτητο όμως η κάθε μια ομάδα από αυτές να έχει παρόμοια συνολική χωρητικότητα στόλου δηλαδή η $1^{η}$ ομάδα οχημάτων αθροιστικά να μπορεί να μεταφέρει ίδιες ποσότητες με τα υπόλοιπα. Για παράδειγμα, εάν στην $1^{η}$ ομάδα οχημάτων περαστούν 5 οχήματα με συνολική χωρητικότητα 10 m^3 (χωρητικότητα σε όγκο) και βάρος 2.000 Kg (χωρητικότητα σε βάρος) τότε είναι υποχρεωτικό και στις υπόλοιπες ομάδες οι τιμές αυτές να είναι όσον το δυνατόν πιο κοντά γίνεται. Βέβαια, ενώ αυτή η

συνθήκη είναι αναγκαία, δεν είναι επαρκής από μόνη της, ειδικά στην περίπτωση όπου ο διαθέσιμος στόλος είναι ανομοιογενής. Για να πλησιάσει η πρώτη λύση του προβλήματος αυτή όσο το δυνατόν την βέλτιστη, θα πρέπει η κάθε ομάδα οχημάτων να μπορεί να έχει διαθέσιμα προς χρησιμοποίηση φορτηγά από όλα τα μεγέθη χωρητικοτήτων (μεγάλα, μεσαία, μικρά).

Για να μπορεί η λύση που θα βρεθεί να πλησιάζει την βέλτιστη, είναι αναγκαίο το κάθε υπό-πρόβλημα που θα επιλύεται να έχει παρόμοιο στόλο προς αξιοποίηση με τα υπόλοιπα υπό-προβλήματα. Για να επιτευχθεί αυτό έγιναν κάποιες δοκιμές με βέλτιστα αποτελέσματα να δίνει η εξής διαδικασία:

Βήμα 1. Μελέτη των στοιχείων του στόλου

Εδώ παρατηρήθηκε ότι ως προς το χαρακτηριστικό Quantity2 (βάρος) τα φορτηγά παρουσιάζουν κοντινές τιμές ,ενώ διαχωρίζονται σε 6 ομάδες ανάλογα με τη χωρητικότητά τους ως προς αυτό. Αντίθετα όμως, ως προς το χαρακτηριστικό Quantity1 (ποσότητα όγκου) παρατηρείται τεράστια διαφοροποίηση. Συνεπώς, η ομαδοποίηση που πραγματοποιήθηκε επικεντρώνεται κυρίως ως προς την εξισορρόπηση όλων των ομάδων για τον όγκο που μπορούν να μεταφέρουν.

Βήμα 2. Επιλογή μεγέθους ομάδων

Στο επίπεδο που βρίσκεται ο κώδικας και λόγω της περιορισμένης μνήμης του υπολογιστή, τα ιδανικά μεγέθη για γρήγορη εκτέλεση και εμφάνιση αποτελεσμάτων είναι σε 13 ομάδες – υπό-προβλήματα -- προς μελέτη. Έτσι, 12 ομάδες οχημάτων θα έχουν από 8 οχήματα και μία ομάδα θα περιέχει 9.

Βήμα 3. Εισαγωγή οχημάτων στις ομάδες

Αρχικά ταξινομούνται τα οχήματα ως προς τη χωρητικότητά τους (σε Quantity1) σε αύξουσα σειρά. Στη πρώτη ομάδα φορτηγών τοποθετείται το φορτηγό με την μικρότερη χωρητικότητα. Έπειτα στη $2^{η}$ ομάδα τοποθετείται το όχημα με την αμέσως επόμενη μικρότερη χωρητικότητα μέχρι να έχουν και οι 13 ομάδες από 1 όχημα στον στόλο τους. Όταν ολοκληρωθεί η διαδικασία αυτή, επαναλαμβάνεται άλλες 7 φορές μέχρι τα υπολειπόμενα οχήματα (στην περίπτωση αυτή 1) να είναι λιγότερα από τον αριθμό των διαχωρισμένων ομάδων.

Βήμα 4. Διορθωτικές (λογικές) αλλαγές

Επειδή η $1^{η}$ ομάδα θα είναι και η ομάδα που θα περιέχει λιγότερα οχήματα με μικρότερη χωρητικότητά της προστίθεται ένα όχημα ακόμα ενώ στις ομάδες 2 και 3 στην προσθήκη οχήματος στην πρώτη επανάληψη της ομαδοποίησης, αντί για το $2^{ο}$ και $3^{ο}$ μικρότερο, για τον ίδιο λόγο, προστίθενται το $1^{ο}$ και $2^{ο}$ μεγαλύτερο όχημα του στόλου. Σε αυτό το σημείο μεγάλο ρόλο για την επιλογή των ομάδων αυτών είχε το γεγονός πως υπάρχει ένα πλήθος είκοσι διαθέσιμων οχημάτων πολύ μεγάλης χωρητικότητας που

θα προτιμηθούν στην πλειοψηφία των υπο-προβλημάτων καθώς ο σκοπός μας σε πρώτο στάδιο είναι η εξοικονόμηση χρόνου.

Τελικές ομάδες οχημάτων

Ακολουθώντας τα βήματα που περιγράφηκαν παραπάνω για μία πρώτη αντικειμενική ομαδοποίηση των δεδομένων, προέκυψαν τα αποτελέσματα του Πίνακα 3.5 που παρατίθεται παρακάτω:

Πίνακας 3.6. Ομαδοποίηση οχημάτων για αρχική επίλυση του προβλήματος.

Ομάδες Οχημάτων	Οχήματα k (k αρίθμηση τους με βάση τα δεδομένα)
1	2 , 43, 70 , 21, 49, 24 ,59, 91, 104
2	105, 68, 5, 55, 74, 58, 84, 92
3	36, 1, 39, 80, 31, 83, 18, 93
4	61, 35, 64, 13, 32, 19, 52, 94
5	3, 60, 4, 47, 14, 53, 77, 95
6	37, 7, 33, 38, 72, 48, 78, 96
7	62, 41, 63, 22,73,17, 34, 97
8	8, 66, 10, 56, 16, 51, 85, 98
9	42, 6, 44, 81, 50, 76, 86, 99
10	67, 40, 69, 12, 75, 20, 87, 100
11	26, 65, 28, 46, 23, 54, 88, 101
12	27, 11, 29, 71, 57, 79, 89, 102
13	9, 45, 30, 15, 82, 25, 90, 103

Η αρίθμηση των οχημάτων στον πίνακα 3.5 έχει γίνει με βάση τον χαρακτηριστικό αριθμό του κάθε οχήματος στα δεδομένα που δόθηκαν. Έτσι στην πρώτη ομάδα οχημάτων, σαν παράδειγμα, έχουν ενταχθεί τα οχήματα με χαρακτηριστικό αριθμό (ID) 2, 43, 70, 21, 49, 24, 59, 91, 104.

Επειδή όλα τα οχήματα τοποθετήθηκαν σε 13 ομάδες, τα σημεία διανομής θα χωριστούν και αυτά σε 13 ομάδες ώστε να βρεθεί μία πρώτη λύση που μελετά όλα τα δεδομένα. Στις ομάδες αυτές, σε πρώτο στάδιο, θα δημιουργηθούν 12 ομάδες των 13 πελατών, και μία τελευταία ομάδα με 15 πελάτες. Λόγω του όγκου δεδομένων των πελατών, σε πρώτο στάδιο οι πελάτες θα περαστούν με την σειρά που δίνονται τα δεδομένα στην αντίστοιχη ομάδα, δηλαδή στη $1^{η}$ ομάδα τοποθετούνται οι πρώτοι 12 πελάτες με αριθμούς 1 έως 12, στη $2^{η}$ αντίστοιχα οι πελάτες με αριθμούς 13 με 24, και συνεχίζει έτσι μέχρι τη τελευταία, που του δίνονται οι τελευταίοι 15 ($157^{ος}$ μέχρι $171^{ος}$).

Πλέον, αντί για ένα μεγάλο πρόβλημα 171 πελατών μέσω 105 οχημάτων, επιλύονται τα 13 νέα πιο μικρά προβλήματα. Για κάθε ένα σύνολο πελατών, υπάρχει μία ομάδα οχημάτων να το εξυπηρετήσει. Λόγω του τρόπου που έγινε ο διαχωρισμός των οχημάτων σε ομάδες, έτσι ώστε να υπάρχει ποικιλομορφία, η λύση που θα προκύψει δεν επηρεάζεται από την ανάθεση μίας ομάδας πελατών σε συγκεκριμένη ομάδα οχημάτων. Για τον λόγο αυτό, για λόγους ευκολίας στην μετάφραση των αποτελεσμάτων, θα ανατεθεί στην κάθε ομάδα πελατών η αντίστοιχη ομάδα οχημάτων, δηλαδή η $3^{η}$ ομάδα πελατών θα εξυπηρετηθεί από την $3^{η}$ ομάδα οχημάτων, η $7^{η}$ ομάδα πελατών η $7^{η}$ ομάδα οχημάτων και ούτω καθ' εξής. Μοναδική εξαίρεση αποτελεί η $1^{η}$ ομάδα των οχημάτων που ανατέθηκε στην $13^{η}$ ομάδα πελατών, καθώς σε αυτές περιέχονται ένα παραπάνω όχημα για αξιοποίηση και τρεις παραπάνω πελάτες. Τα πρώτα αποτελέσματα ανάθεσης δίνονται μέσω της κύριας μεταβλητής απόφασης που χρησιμοποιήθηκε, της X_{ijk} ,μέσω της οποίας δίνονται αποτελέσματα για τα οχήματα που θα δρομολογηθούν, τους πελάτες που θα εξυπηρετήσουν αλλά και την σειρά που θα γίνει η κάλυψη των παραγγελιών τους. Τα αποτελέσματα της πρώτης λύσης δίνονται στον Πίνακα 3.6 που ακολουθεί.

Πίνακας 3.7. Αποτελέσματα λύσης των υπο-προβλημάτων που μελετήθηκαν.

ΟΜΑΔΕΣ ΠΕΛΑΤΩΝ	ΟΜΑΔΕΣ ΦΟΡΤΗΓΩΝ	ΔΡΟΜΟΛΟΓΗΣΗ
1	13 *	Xijk(0,10,7)=1 Xijk(1,2,7)=1 Xijk(2,4,7)=1 Xijk(3,6,7)=1 Xijk(4,0,7)=1 Xijk(5,11,7)=1 Xijk(6,8,7)=1 Xijk(7,12,7)=1 Xijk(8,1,7)=1 Xijk(9,3,7)=1 Xijk(10,5,7)=1 Xijk(11,7,7)=1 Xijk(12,9,7)=1
2	2	Xijk(0,12,5)=1 Xijk(3,6,5)=1 Xijk(5,9,5)=1 Xijk(6,0,5)=1 Xijk(9,3,5)=1 Xijk(10,5,5)=1 Xijk(12,10,5)=1 Xijk(0,11,7)=1 Xijk(1,2,7)=1 Xijk(2,4,7)=1 Xijk(4,8,7)=1 Xijk(7,1,7)=1 Xijk(8,0,7)=1 Xijk(11,7,7)=1
3	3	Xijk(0,6,7)=1 Xijk(1,2,7)=1 Xijk(2,4,7)=1 Xijk(3,7,7)=1 Xijk(4,8,7)=1 Xijk(5,11,7)=1 Xijk(6,12,7)=1 Xijk(7,1,7)=1 Xijk(8,0,7)=1 Xijk(9,3,7)=1 Xijk(10,5,7)=1 Xijk(11,9,7)=1 Xijk(12,10,7)=1
4	4	Xijk(0,11,7)=1 Xijk(1,2,7)=1 Xijk(2,7,7)=1 Xijk(3,6,7)=1 Xijk(4,8,7)=1 Xijk(5,4,7)=1

		Xijk(6,12,7)=1 Xijk(7,0,7)=1 Xijk(8,1,7)=1 Xijk(9,3,7)=1 Xijk(10,5,7)=1 Xijk(11,9,7)=1 Xijk(12,10,7)=1
5	5	Xijk(0,2,0)=1 Xijk(1,10,0)=1 Xijk(2,4,0)=1 Xijk(3,6,0)=1 Xijk(4,8,0)=1 Xijk(5,9,0)=1 Xijk(6,12,0)=1 Xijk(7,0,0)=1 Xijk(8,1,0)=1 Xijk(9,3,0)=1 Xijk(10,5,0)=1 Xijk(11,7,0)=1 Xijk(12,11,0)=1
6	6	Xijk(0,3,4)=1 Xijk(1,2,4)=1 Xijk(2,4,4)=1 Xijk(3,6,4)=1 Xijk(4,11,4)=1 Xijk(5,8,4)=1 Xijk(6,12,4)=1 Xijk(7,0,4)=1 Xijk(8,1,4)=1 Xijk(9,10,4)=1 Xijk(10,5,4)=1 Xijk(11,7,4)=1 Xijk(12,9,4)=1
7	7	Xijk(0,11,7)=1 Xijk(1,2,7)=1 Xijk(2,4,7)=1 Xijk(3,6,7)=1 Xijk(4,0,7)=1 Xijk(5,10,7)=1 Xijk(6,12,7)=1 Xijk(7,9,7)=1 Xijk(8,1,7)=1 Xijk(9,3,7)=1 Xijk(10,7,7)=1 Xijk(11,5,7)=1 Xijk(12,8,7)=1
8	8	Xijk(0,8,0)=1 Xijk(1,2,0)=1 Xijk(2,4,0)=1 Xijk(3,6,0)=1

		Xijk(4,12,0)=1 Xijk(5,10,0)=1 Xijk(6,11,0)=1 Xijk(7,5,0)=1 Xijk(8,1,0)=1 Xijk(9,3,0)=1 Xijk(10,0,0)=1 Xijk(11,7,0)=1 Xijk(12,9,0)=1
9	9	Xijk(0,8,1)=1 Xijk(1,9,1)=1 Xijk(2,7,1)=1 Xijk(3,11,1)=1 Xijk(4,12,1)=1 Xijk(5,10,1)=1 Xijk(6,0,1)=1 Xijk(7,1,1)=1 Xijk(8,2,1)=1 Xijk(9,3,1)=1 Xijk(10,4,1)=1 Xijk(11,5,1)=1 Xijk(12,6,1)=1
10	10	Xijk(0,5,7)=1 Xijk(1,0,7)=1 Xijk(2,4,7)=1 Xijk(3,6,7)=1 Xijk(4,8,7)=1 Xijk(5,10,7)=1 Xijk(6,12,7)=1 Xijk(7,2,7)=1 Xijk(8,1,7)=1 Xijk(9,11,7)=1 Xijk(10,3,7)=1 Xijk(11,7,7)=1 Xijk(12,9,7)=1
11	11	Xijk(0,3,6)=1 Xijk(1,2,6)=1 Xijk(2,4,6)=1 Xijk(3,6,6)=1 Xijk(4,0,6)=1 Xijk(5,11,6)=1 Xijk(6,12,6)=1 Xijk(7,8,6)=1 Xijk(8,1,6)=1 Xijk(9,10,6)=1 Xijk(10,5,6)=1 Xijk(11,7,6)=1 Xijk(12,9,6)=1
12	12	Xijk(0,2,3)=1 Xijk(1,0,3)=1

		Xijk(2,4,3)=1 Xijk(4,8,3)=1 Xijk(8,1,3)=1 Xijk(0,5,6)=1 Xijk(3,6,6)=1 Xijk(5,10,6)=1 Xijk(6,12,6)=1 Xijk(7,9,6)=1 Xijk(9,3,6)=1 Xijk(10,11,6)=1 Xijk(11,7,6)=1 Xijk(12,0,6)=1
13	1 *	Xijk(0,3,6)=1 Xijk(3,6,6)=1 Xijk(6,12,6)=1 Xijk(7,13,6)=1 Xijk(9,0,6)=1 Xijk(12,7,6)=1 Xijk(13,9,6)=1 Xijk(0,5,7)=1 Xijk(5,10,7)=1 Xijk(10,0,7)=1 Xijk(0,14,8)=1 Xijk(1,2,8)=1 Xijk(2,4,8)=1 Xijk(4,8,8)=1 Xijk(8,0,8)=1 Xijk(11,1,8)=1 Xijk(14,11,8)=1

Όπως αναφέρθηκε και στην ανάλυση των μεταβλητών απόφασης του προβλήματος, η δυική μεταβλητή X_{ijk} παίρνει την τιμή 1 μόνο εάν το αντίστοιχο k φορτηγό πραγματοποιήσει μία διαδρομή από το σημείο i στο σημείο j. Στο Πίνακα 3.7 δίνονται πιο αναλυτικά και κατανοητά οι αντιστοιχίσεις πελατών και οχημάτων που προέκυψαν από την προγραμματιστική λύση καθώς και η σειρά που εκτελέστηκαν τα δρομολόγια.

Πίνακας 3.8 Ανάλυση των αποτελεσμάτων που προέκυψαν.

Ομάδες Πελατών	Φορτηγό(α) που χρησιμοποιήθηκαν	Σειρά εξυπηρέτησης
1 (πελάτες 1 έως 13)	103	0-10-5-11-7-12-9-13-3-6-8-1-2-4-0
2(πελάτες 14 έως 26)	58, 92	0-24-22-17-21-15-26-18-0 0-23-19-25-14-16-20-0
3(πελάτες 27 έως 39)	93	0-30-36-34-38-29-35-33-27-31-37-39-28-32-0
4(πελάτες 40 έως 52)	94	0-50-48-42-45-51-49-52-44-43-47-40-41-46-0
5(πελάτες 53 έως 65)	3	0-55-57-60-53-62-57-61-55-58-64-63-65-59-0
6(πελάτες 66 έως 78)	72	0-68-71-77-74-75-70-73-66-67-78-69-76-72-0
7(πελάτες 79 έως 91)	97	0-90-83-89-85-87-81-84-90-86-79-80-82-0
8(πελάτες 92 έως 104)	8	0-99-92-93-95-103-104-101-94-97-102-98-96-101-0
9(πελάτες 105 έως 117)	6	0-112-106-111-105-113-107-115-109-114-108-117-116-110-0
10(πελάτες 118 έως 130)	100	0-122-127-120-123-129-126-128-124-119-121-125-118-0
11(πελάτες 131 έως 143)	88	0-133-136-142-139-140-135-141-143-137-138-131-132-134-0
12(πελάτες 144 έως 156)	71, 89	0-145-147-151-144-0 0-148-153-154-156-150-152-146-149-155-0
13(πελάτες 157 έως 171)	59, 81, 104	0-159-162-168-163-169-165 0-161-166-0 0-170-171-167-157-158-160-164-0

3.5. Συμπεράσματα με βάση τα αποτελέσματα

Με την πραγματοποίηση της ομαδοποίησης:

- Θυσιάστηκε ένα μέρος της «βελτιστότητας» για να μπορέσει να βρεθεί μία εφικτή λύση που να την πλησιάζει.
- Στην πλειοψηφία των μικρών προβλημάτων που επιλύθηκαν η δρομολόγηση ενός μεγάλου φορτηγού δίνει καλύτερα αποτελέσματα από την δρομολόγηση πολλαπλών μικρότερων, καθώς οι αποστάσεις μεταξύ των σημείων δεν είναι πολύ μεγάλες.
- Στην περίπτωση του $2^{ου}$, του $12^{ου}$ και του $13^{ου}$ υπό-προβλήματος προτιμήθηκαν μεγάλα φορτηγά πάλι για τις περισσότερες διαδρομές, με κάποια μικρότερα να καλύπτουν την υπόλοιπη ζήτηση.
- Πιο συγκεκριμένα, στην τελευταία υποομάδα πελατών, επιστρατεύτηκαν δύο φορτηγά πολύ μεγάλης χωρητικότητας και ένα μεσαίας.
- Αυτό το αποτέλεσμα οφείλεται στο είδος της παραγγελίας συγκεκριμένων πελατών της ομάδας όπου ο όγκος της παραγγελίας τους είναι πολύ μεγάλος, αλλά και το γεγονός πως οι τοποθεσίες

τους απέχουν αρκετά μεταξύ τους, σε βαθμό που η δρομολόγηση ενός παραπάνω οχήματος είναι πιο επιθυμητή.

Η λύση που βρέθηκε με αυτήν την διαδικασία είναι εφικτή και πλησιάζει σε έναν βαθμό την βέλτιστη. Από τα αποτελέσματα που προέκυψαν όμως, είναι φανερό πως με την πραγματοποίηση ενός πιο ποιοτικού διαχωρισμού των πελατών σε ομάδες, τα αποτελέσματα που θα προκύψουν θα είναι εμφανώς καλύτερα. Συνεπώς, έχοντας τελειώσει με την δομή του προβλήματος το επόμενο στάδιο για να πλησιάσει η λύση μας την βέλτιστη ακόμα πιο πολύ, είναι η πραγματοποίηση μιας ομαδοποίησης με τρόπο τέτοιον που θα εξοικονομείται ακόμα περισσότερος χρόνος. Αυτό μπορεί να επιτευχθεί με τη χρήση κατάλληλων αλγορίθμων ομαδοποίησης των δεδομένων. Στο επόμενο κεφάλαιο της εργασίας αυτής αναλύονται κάποιοι από αυτούς τους αλγορίθμους. Πραγματοποιείται επίσης εφαρμογή τους και γίνεται σύγκριση των αποτελεσμάτων.

ΚΕΦΑΛΑΙΟ 4: ΟΜΑΔΟΠΟΙΗΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΜΕΣΩ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ Κ-ΜΕΣΩΝ ΚΑΙ ΕΠΙΛΥΣΗ

4.1. Τεχνικές ομαδοποίησης δεδομένων - Εφαρμογή αλγορίθμων

4.1.1. Χρήση αλγορίθμων και τεχνικών στα προβλήματα

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, προβλήματα τέτοιου μεγέθους δεν είναι πάντα ικανά να επιλύονται απευθείας μετά το τέλος της μορφοποίησης τους. Αυτό κυρίως συμβαίνει όταν ο αριθμός των παραμέτρων (περιορισμών) που πρέπει να μελετηθούν είναι αρκετά μεγάλος, δημιουργώντας έτσι ένα μεγάλο πεδίο διαφορετικών περιπτώσεων (λύσεων).

Για την αντιμετώπιση αυτών των προβλημάτων υπάρχουν συγκεκριμένες διαδικασίες επίλυσης ανάλογα την περίπτωση του προβλήματος προς αντιμετώπιση δηλαδή τη δημιουργία αλγοριθμικών διαδικασιών (δημιουργία αλγορίθμων). Ως αλγόριθμος ορίζεται μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, με σκοπό την επίλυση ενός προβλήματος. Πιο απλά (αλγόριθμο) ονομάζουμε μία σειρά από εντολές που έχουν αρχή και τέλος, είναι σαφείς και έχουν ως σκοπό την επίλυση κάποιου προβλήματος. Στην μελέτη προβλημάτων επιχειρησιακής έρευνας και βελτιστοποίησης, οι κύριες κατηγορίες αλγορίθμων επίλυσης οι ευρετικοί (heuristics) και οι μετευρετικοί αλγόριθμοι (meta-heuristics).

Heuristics: Τα Heuristics είναι τεχνικές που εξαρτώνται από το πρόβλημα. Ως εκ τούτου προσαρμόζονται συνήθως στο πρόβλημα και προσπαθούν να αξιοποιήσουν πλήρως τις ιδιαιτερότητες του. Ωστόσο επειδή είναι συχνά υπερβολικά άπληστοι, συνήθως παγιδεύονται σε ένα τοπικό βέλτιστο και έτσι έχουν πιθανότητα αποτυχίας στο να αποκτήσουν τη ολική βέλτιστη λύση. Η εφαρμογή τους παρατηρείται κυρίως σε μεγάλα και δύσκολα (βάσει υπολογιστικού χώρου και χρόνου που απαιτείται) προβλήματα και επικαλούνται αλγορίθμους για εμφάνιση αρχικών εφικτών λύσεων που στην πορεία προσπαθούν να βελτιώσουν. Συνήθως βρίσκουν καλές λύσεις για μεγάλα προβλήματα δεν εγγυούνται όμως βελτιστότητα.

Meta-Heuristics: Ο όρος meta-heuristic προτάθηκε από τον Glover στα μέσα της δεκαετίας του '80 ως οικογένεια αλγορίθμων αναζήτησης ικανών να ορίσουν λύσεις υψηλού επιπέδου που χρησιμοποιείται για να καθοδηγήσει άλλα heuristics για μια καλύτερη εξέλιξη στον χώρο αναζήτησης. Τα μετα-ευρετικά είναι τεχνικές ανεξάρτητες από το πρόβλημα. Ως εκ τούτου, δεν επωφελούνται από οποιαδήποτε ιδιαιτερότητα του προβλήματος και μπορούν να χρησιμοποιηθούν ως μαύρα κουτιά. Σε γενικές γραμμές, δεν είναι άπληστοι. Στην πραγματικότητα, μπορούν ακόμη και να δεχτούν μια προσωρινή επιδείνωση της

λύσης (βλέπε για παράδειγμα την τεχνική προσομοίωσης - ανόπτησης) που τους επιτρέπει να διερευνήσουν πιο διεξοδικά το χώρο λύσης και έτσι να βρουν μια καλύτερη (που μερικές φορές θα συμπίπτει με ολικό βέλτιστο). Παρόλο που τα Meta-Heuristics είναι τεχνική ανεξάρτητη από προβλήματα είναι απαραίτητο να γίνει κάποια τελειοποίηση των εγγενών παραμέτρων της, προκειμένου να προσαρμοσθεί η εκάστοτε τεχνική στο πρόβλημα. Συνεπώς, για να εφαρμοστεί κάποια από τις παραπάνω διαδικασίες, είναι απαραίτητο να γνωρίζουμε από την αρχή ποιος ακριβώς είναι ο σκοπός μας όταν καλούμαστε να επιλύσουμε ένα τέτοιο πρόβλημα βελτιστοποίησης.

Στη συνέχεια του κεφαλαίου αυτού εφαρμόζονται ομαδοποιήσεις των δεδομένων μας (καταναλωτικά κέντρα), μελετώνται οι παραδοχές που θα γίνουν και συγκρίνονται αποτελέσματα από δύο διαφορετικές διαδικασίες επίλυσης που εφαρμόστηκαν.

4.1.2. Ομαδοποίηση των δεδομένων με τον αλγόριθμο K-means

Όπως αναφέρθηκε ήδη στα προηγούμενα κεφάλαια, το κύριο θέμα που έχουμε να αντιμετωπίσουμε στην περίπτωση του προβλήματος μας είναι πως ο χώρος λύσεων είναι πάρα πολύ μεγάλος. Γνωρίζουμε ότι υπάρχουν 171 διαφορετικά σημεία των οποίων οι ζητήσεις πρέπει να ικανοποιηθούν καθώς και το ότι ο διαθέσιμος στόλος περιέχει 105 οχήματα που χαρακτηρίζονται από ετερογένεια ως προς τη χωρητικότητα του κάθε οχήματος και από περιορισμένη συνολική χωρητικότητα. Καταλαβαίνει κανείς εύκολα ότι για να μελετηθεί όλο το πεδίο των εφικτών λύσεων και συνδυασμών απαιτείται πάρα πολύς χρόνος (αλλά και υπολογιστικές δυνατότητες).

Επανερχόμαστε λοιπόν στην στρατηγική που αναφέραμε στο τέλος του 3^{ου} κεφαλαίου, δηλαδή στην ομαδοποίηση των δεδομένων μας και την επίλυση μικρότερων προβλημάτων. Αυτήν την φορά όμως η ομαδοποίηση δεν θα είναι μυωπική αλλά θα γίνει με χρήση του κατάλληλου αλγορίθμου, στην περίπτωση αυτήν του αλγορίθμου k-means.

Ένας αλγόριθμος ομαδοποίησης προσπαθεί να αναλύσει φυσικές ομάδες δεδομένων με βάση κάποια ομοιότητα. Εντοπίζει το κέντρο της ομάδας των σημείων δεδομένων. Για την πραγματοποίηση αποτελεσματικής ομαδοποίησης ο αλγόριθμος αξιολογεί την απόσταση μεταξύ κάθε σημείου από το κέντρο του συμπλέγματος. Ο στόχος της ομαδοποίησης είναι να προσδιοριστεί η εγγενής ομαδοποίηση σε ένα σύνολο μη επισημασμένων δεδομένων.

Ο αλγόριθμος k-means (κ-μέσων) εφαρμόζεται για να πραγματοποιηθεί μια αξιόπιστη ομαδοποίηση σημείων - δεδομένων με βάση ένα κύριο χαρακτηριστικό τους. Θα αναφερόμαστε στην διαδικασία αυτήν

σαν clustering στην συνέχεια. Πριν προχωρήσουμε στην ανάλυση του αλγορίθμου k-means, ας απαντήσουμε κάποιες βασικές ερωτήσεις :

Τι είναι το clustering;

- Με τον όρο clustering χαρακτηρίζουμε την διαδικασία ομαδοποίησης σημείων δεδομένων σε ομογενείς ομάδες ή αλλιώς clusters.
- Τα σημεία στην ίδια ομάδα είναι όσο το δυνατόν πιο παρόμοια ως προς το στοιχείο για το οποίο γίνεται η ομαδοποίηση.
- Τα σημεία σε διαφορετική ομάδα είναι όσο το δυνατόν πιο ανόμοια ως προς το στοιχείο για το οποίο γίνεται η ομαδοποίηση.

Εφαρμογές του clustering μέσω του αλγορίθμου k-means:

- Ταξινόμηση εγγράφων.
- Προβλήματα βελτιστοποίησης για καταστήματα παράδοσης.
- Προσδιορισμός περιοχών εγκληματικότητας.
- Τμηματοποίηση πελατών.
- Ανάλυση στατιστικών σε αθλήματα.
- Ανάλυση ιστορικού κλήσεων.
- Καταγραφή ιστορικότητας καιρού.

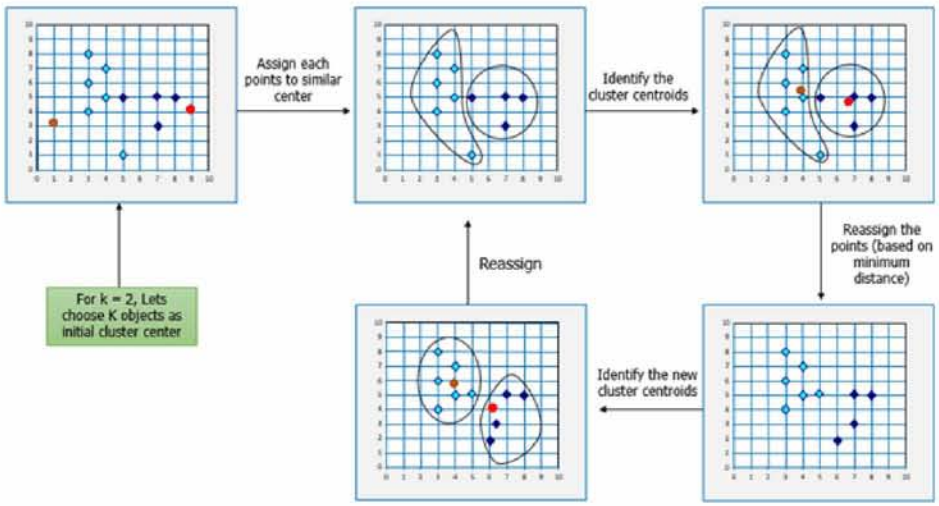
4.1.3. Λειτουργία αλγορίθμου K-means.

Ο αλγόριθμος ομαδοποίησης Κ-μέσων χρησιμοποιεί επαναληπτική βελτίωση για να παραχθεί ένα τελικό αποτέλεσμα. Η είσοδος του αλγορίθμου είναι ο αριθμός των ομάδων Κ και το σύνολο των δεδομένων. Το σύνολο δεδομένων είναι μια συλλογή χαρακτηριστικών για κάθε σημείο των δεδομένων του προβλήματος. Ο αλγόριθμος αρχίζει με αρχικές αναθέσεις για τα Κ κέντρα (centroids), τα οποία μπορούν είτε να παραχθούν τυχαία είτε να επιλεγούν τυχαία από το σύνολο δεδομένων και ακολουθεί τα εξής βήματα:

Βήμα 1: Ορίζουμε έναν Κ αριθμό τυχαίων κέντρων (centroids). Κάθε centroid αντιστοιχεί σε ένα cluster
Βήμα 2: Ανάθεση όλων των σημείων στο πιο κοντινό τους κέντρο. Όταν τελειώσει η ανάθεση έχουμε τα πρώτα clusters.
Βήμα 3: Υπολογισμός νέων κέντρων – centroid για κάθε cluster. Αυτό γίνεται με τη λήψη του μέσου όλων των σημείων.

Βήμα 4: Ο αλγόριθμος επαναλαμβάνεται μεταξύ των βημάτων 2 και 3 μέχρις ότου ικανοποιηθούν τα κριτήρια στάσης (δηλαδή κανένα σημείο δεδομένων δεν αλλάζει τα σύνολα, το άθροισμα των αποστάσεων ελαχιστοποιείται ή επιτυγχάνεται κάποιος μέγιστος αριθμός επαναλήψεων).

Αυτός ο αλγόριθμος εγγυάται ότι θα συγκλίνει σε ένα αποτέλεσμα. Το αποτέλεσμα μπορεί να είναι μια τοπική βέλτιστη (δηλαδή όχι απαραίτητα το καλύτερο δυνατό αποτέλεσμα) που σημαίνει ότι η εκτίμηση περισσοτέρων του ενός διαδρομών του αλγορίθμου με τυχαιοποιημένα αρχικά centroids μπορεί να δώσει ένα καλύτερο αποτέλεσμα. Η διαδικασία του K-means περιγράφεται με το παρακάτω σχήμα (Σχήμα 4.1).



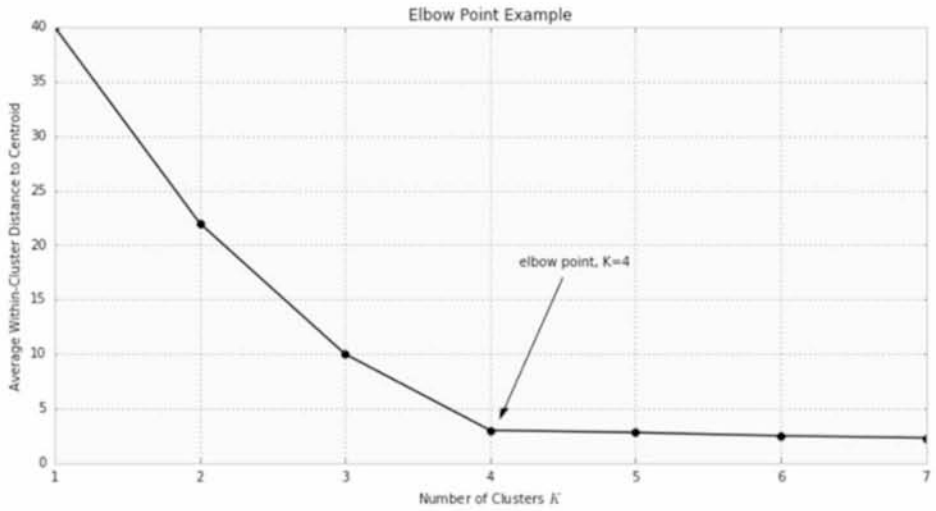
Σχήμα 4.1. Σχηματική απεικόνιση της λειτουργίας του αλγορίθμου k-means.

Ο αλγόριθμος που περιεγράφηκε παραπάνω εντοπίζει τα clusters και τα κέντρα τους για ένα συγκεκριμένο προεπιλεγμένο K. Για να βρεθεί ο αριθμός των συστάδων στα δεδομένα ο χρήστης πρέπει να εκτελέσει τον αλγόριθμο ομαδοποίησης K-means για ένα εύρος τιμών K και να συγκρίνει τα αποτελέσματα. Γενικά, δεν υπάρχει μέθοδος για τον προσδιορισμό της ακριβούς τιμής του K αλλά μια ακριβής εκτίμηση μπορεί να ληφθεί χρησιμοποιώντας τις ακόλουθες τεχνικές.

Μία από τις μετρήσεις που χρησιμοποιούνται συνήθως για τη σύγκριση των αποτελεσμάτων με διαφορετικές τιμές του K είναι και αυτή της μέσης απόστασης μεταξύ των σημείων της ομαδοποίησης και του τυχαίου κέντρου (centroid) της ομάδας τους. Δεδομένου ότι η αύξηση του αριθμού των συστάδων θα μειώνει πάντα την απόσταση στα σημεία δεδομένων, η αύξηση του K θα μειώνει πάντα αυτή τη μέτρηση, με ακρίβεια να φτάνει στο μηδέν όταν το K είναι το ίδιο με τον αριθμό των σημείων

δεδομένων. Έτσι, αυτή η μέτρηση δεν μπορεί να χρησιμοποιηθεί ως ο μοναδικός στόχος. Αντ' αυτού, η μέση απόσταση από το centroid ως συνάρτηση του K είναι γραφική παράσταση και το "σημείο αγκώνα" όπου ο ρυθμός μείωσης μετατοπίζεται απότομα μπορεί να χρησιμοποιηθεί για να καθορίσει χονδρικά το K, όπως φαίνεται και στο Σχήμα 4.7.

Υπάρχουν διάφορες άλλες τεχνικές για την επικύρωση του K, συμπεριλαμβανομένης της διασταυρούμενης επικύρωσης (cross validation), των κριτηρίων πληροφόρησης, της μεθόδου θεωρητικής άλλης πληροφόρησης (the information theoretic jump method), της μεθόδου σιλουέτας και του αλγορίθμου G-mean. Επιπλέον, η παρακολούθηση της διανομής των σημείων δεδομένων σε όλες τις ομάδες παρέχει μια εικόνα για το πώς ο αλγόριθμος διαιρεί τα δεδομένα για κάθε K.



Σχήμα 4.2. Διάγραμμα Μέσης Απόστασης σημείων από centroids, ως προς τον συνολικό αριθμό των centroids.

Για το συγκεκριμένο πρόβλημα χρησιμοποιήθηκε σαν μέτρο σύγκρισης η μέση απόσταση του συνόλου των σημείων που ανήκουν σε μία ομάδα από το αντίστοιχο centroid. Με την μέθοδο αυτή καταλήγουμε πως ο ιδανικός αριθμός ομάδων είναι 8 ή 9 ομάδες πελατών.

Παρακάτω παρουσιάζονται τα αποτελέσματα που προέκυψαν από την διαδικασία που περιεγράφηκε παραπάνω ενώ στην συνέχεια του κεφαλαίου γίνεται επίλυση του προβλήματος επιλύοντας τα μικρότερα πλέον προβλήματα με τις ομάδες που δημιουργήθηκαν για οκτώ και για εννιά ομάδες. Στους πίνακες 4.1 και 4.2 δίνονται τα αποτελέσματα της ομαδοποίησης για οκτώ και εννέα K μέσους. Επίσης, μέσω του προγράμματος QGIS 2018 μπορούμε να έχουμε μία εικόνα για την σωστή λειτουργία της ομαδοποίησης.

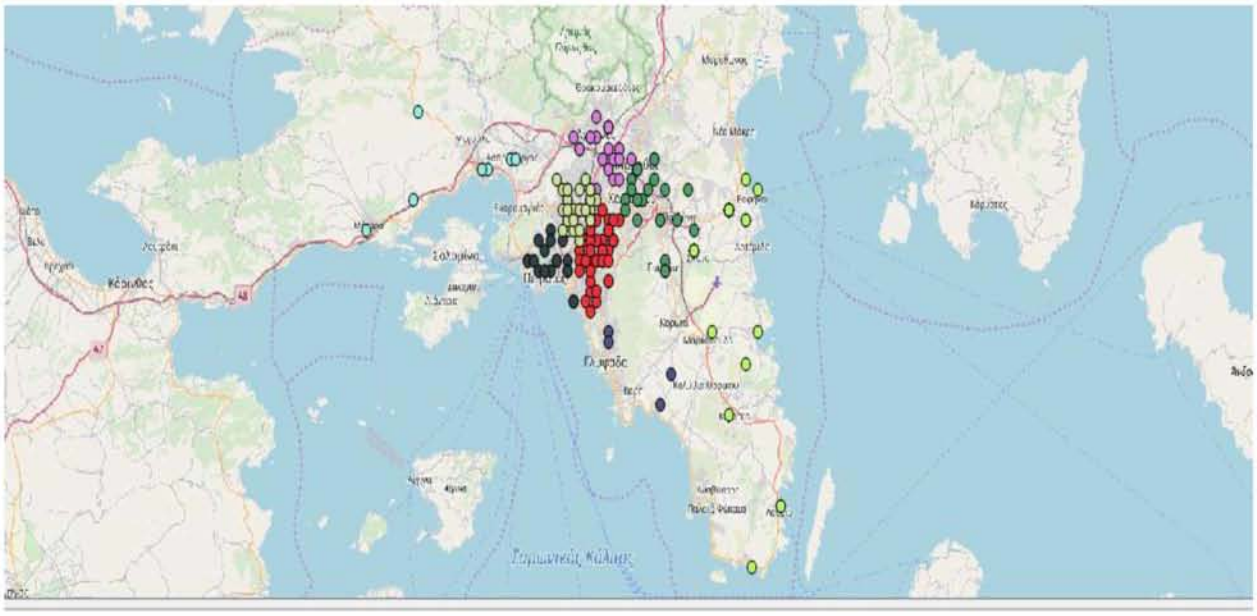
Τέλος για την λειτουργία του αλγορίθμου, πέρα από τα στοιχεία που πρέπει να ομαδοποιηθούν είναι απαραίτητο να θέσουμε εμείς τα πρώτα centroids των ομάδων. Στην μελέτη που έγινε, τα στοιχεία που τέθηκαν σαν πρώτα κέντρα είναι οι συντεταγμένες των πρώτων οκτώ και εννιά πελατών αντίστοιχα.

Πίνακας 4.1. Αποτελέσματα ομαδοποίησης με τον αλγόριθμο k-means για οκτώ (8) ομάδες.

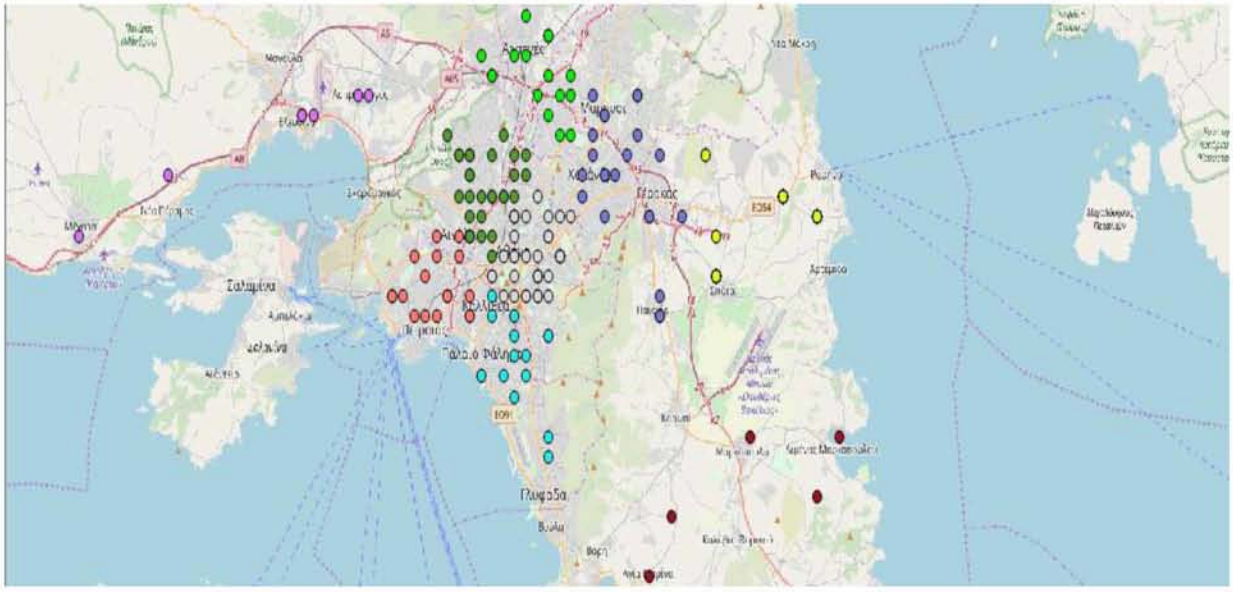
Ομάδες Πελατών	Αριθμός Πελατών ανά cluster	ID πελατών των clusters	Συντεταγμένες των centroids
Ομάδα 1	8	91,96,97,102, 103, 111, 160, 161	(23.72, 37.98)
Ομάδα 2	19	11,12,13,41,44,55,68,70,81,109,122,130,142,145,148,150, 151 ,156, 157	(23.76, 38)
Ομάδα 3	22	45,46,56,57,64,78,82,84,93,121,127,155,158,162,163,164, 165,166,167,168,169,170,171	(23.78, 38)
Ομάδα 4	4	22, 27, 98, 158	(23.74, 37.96)
Ομάδα 5	44	1,2,3,4,5,6,7,8,16,17,18,19,20,21,23,29,30,31,32,33,35,43, 49,50,51,52,59,63,73,92,100,101,107,110,112,113,116, 117,118,123, 124,126,128,136	(23.75, 37.98)
Ομάδα 6	23	9,10,48,53,61,71,77,79,80,83,104,105,106,119,120,129, 134,135,141,146,149,154,159	(23.77, 38)
Ομάδα 7	18	36,37,38,39,40,42,65,67,75,85,86,87,88,89,137,138,143, 147	(23.74, 37.98)
Ομάδα 8	33	14,15,24,25,26,28,34,47,54,58,60,62,66,69,74,76,90,94,95 , 99, 108,114, 115,125,131,132,133,139,140,144,152,153	(23.73, 37.98)

Πίνακας 4.2. Αποτελέσματα ομαδοποίησης με τον αλγόριθμο k-means για εννιά (9) ομάδες.

Ομάδες Πελατών	Αριθμός Πελατών ανά cluster	ID πελατών των clusters	Συντεταγμένες των centroids
Ομάδα 1	8	91,96,97,102, 103, 111, 160, 161	(23.72, 37.98)
Ομάδα 2	17	11,12,13,41,44,55,68,81,122,130,142, 145,148, 150, 151, 156, 157	(23.76, 38)
Ομάδα 3	9	22, 45, 46,78, 82, 84, 93, 155, 158	(23.78, 38)
Ομάδα 4	16	27, 29, 35, 49, 50, 51,65, 73 , 98, 100, 101, 107, 123, 126 , 128,138	(23.74, 37.96)
Ομάδα 5	35	1,2,3,4,5,6,7,8,16,17,18,19,20,21,23,30,31,32,33,43,52, 59,63,92,94,108, 110,112,113,114,116,117,118,124,136,	(23.75, 37.98)
Ομάδα 6	22	9,10,48, 53, 61,71,77, 80, 83,104,105,106,109,119,120,129,134,135,141,146,154, 159	(23.77, 38)
Ομάδα 7	17	36,37,38,39,40,42,60,67,75,85,86,87,88,89,137,143,147	(23.74, 37.98)
Ομάδα 8	30	14,15,24,25,26,28,34,47,54,58,62,66,69,70,72,74,76,90, 95,99,115,125,131,132,133,139,140,144,152,153	(23.73, 37.98)
Ομάδα 9	17	56,57,64,79,121,127,149,162,163,164,165,166,167,168, 169,170,171	(23.84, 38.06)



Σχήμα 4.3. Ομάδες πελατών μετά την ομαδοποίηση με αλγόριθμο K-Means, για κ =8.



Σχήμα 4.4. Ομάδες πελατών μετά την ομαδοποίηση με αλγόριθμο K-Means, για κ =9.

4.2. Επίλυση του προβλήματος

Στο στάδιο που αυτό, τα καταναλωτικά κέντρα έχουν ομαδοποιηθεί αναλόγως με τη τοποθεσία τους και επόμενο βήμα είναι η επιλογή του κατάλληλου τρόπου για να γίνει η επίλυση. Επειδή ο στόλος που είναι διαθέσιμος στην περίπτωση μας χαρακτηρίζεται από πληθώρα αριθμού και χωρητικότητας οχημάτων, είναι εφικτό να ομαδοποιηθεί με τρόπο τέτοιο ώστε να μην απειλείται η απόδοση της λύσης. Σε αυτό συμβάλλει σε μεγάλο βαθμό το ότι δεν έχουμε να λάβουμε υπόψη παραμέτρους όπως η ελαχιστοποίηση του κενού φορτίου σε συνδυασμό με την επιρροή που έχει στην κατανάλωση και το κόστος των καυσίμων. Είναι δηλαδή εφικτό, κάθε ομάδα υπό-προβλήματος να έχει μια ποικιλία οχημάτων διαθέσιμα για αξιοποίηση.

Στην πορεία θα αναλύσουμε δύο διαφορετικούς τρόπους λύσεων:

- Ο πρώτος με αναλυτική επίλυση του προβλήματος με τεχνικές μεικτού ακεραίου προγραμματισμού (MIP) με χρήση του ILOG Cplex Optimization Studio.
- Ο δεύτερος τρόπος επίλυσης είναι με την χρήση του αλγορίθμου του Κοντινότερου Γείτονα (Nearest Neighbor Algorithm).

4.2.1. Αναλυτική επίλυση των Προβλημάτων

Στο κεφάλαιο 3 της εργασίας αναφέρθηκε πως το σημαντικότερο πρόβλημα στην αντιμετώπιση προβλημάτων σαν και αυτό είναι ο τεράστιος όγκος δεδομένων και πιθανών λύσεων. Πραγματοποιώντας την ομαδοποίηση που αναφέρθηκε παραπάνω το πρόβλημα αυτό θα αντιμετωπιστεί πλέων σαν μικρότερα υπό-προβλήματα VRP για τα οποία μπορούμε να έχουμε καλές λύσεις σε ικανοποιητικό χρόνο. Το νέο πρόβλημα που θα αντιμετωπιστεί μπορεί να χαρακτηριστεί και σαν mTSP διατηρώντας όμως τις ιδιομορφίες και τους περιορισμούς που πρέπει να ικανοποιούνται.

Για την αναλυτική λύση θα εφαρμοστεί το ίδιο μοντέλου που αναλύθηκε στο κεφάλαιο 2 με την διαφορά ότι πλέον οι διαστάσεις και ο αριθμός των λύσεων είναι μικρότερος. Για την επίλυση των προβλημάτων μέσω ILOG Cplex Optimization Studio υπήρξε όριο αποτυχημένων λύσεων από 200.000 μέχρι και 400.000 ανάλογα το μέγεθος της ομάδας.

Τέλος, λόγω της ομαδοποίησης των σημείων με δοκιμασμένο αλγοριθμικό τρόπο περιμένουμε καλύτερα αποτελέσματα από αυτά της ανάθεσης που είχαμε στο κεφάλαιο 3.

4.2.2. Αποτελέσματα από την Αναλυτική επίλυση

Εισάγοντας τα στοιχεία του προβλήματος στο ILOG Cplex Optimization Studio επιτεύχθηκαν τα παρακάτω αποτελέσματα που παρουσιάζονται στο Πίνακα 4.3:

Πίνακας 4.3. Χρόνοι Δρομολόγησης Οχημάτων 1ης ομάδας πελατών. Ο αριθμός συμβολίζει το λεπτό όπου το φορτηγό έχει φτάσει στον αντίστοιχο πελάτη και έχει ξεκινήσει την διαδικασία εξυπηρέτησης.

Truck 1	Truck 2	Truck 3	Truck 4
753	0	0	0
711	0	0	0
0	0	0	813
641	0	0	0
0	0	0	790
728	0	0	0
0	0	766	0
0	0	0	0

Ο παραπάνω είναι σε λεπτά, οπότε μετατρέποντας τον για ευκολότερη κατανόηση σε ώρες (Πίνακας 4.4):

Πίνακας 4.4. Χρόνοι Δρομολόγησης Οχημάτων 1ης ομάδας πελατών. Στα στοιχεία του πίνακα περιέχονται οι ώρες όπου το φορτηγό έχει φτάσει στον αντίστοιχο πελάτη και έχει ξεκινήσει την διαδικασία εξυπηρέτησης.

Truck 1	Truck 2	Truck 3	Truck 4	Truck 5
12:33	0	0	0	0
11:51	0	0	0	0
0	0	0	12:53	0
10:41	0	0	0	0
0	0	0	12:30	0
12:08	0	0	0	0
0	0	12:46	0	0
0	0	0	0	0

Πέρα από τα στοιχεία της μεταβλητής απόφασης W στο μοντέλο που αναλύθηκε στο Κεφάλαιο 2, μέσω της μεταβλητής απόφασης Xi,j,k, τα αποτελέσματα του προβλήματος δίνονται και στην σειρά ανάθεσης τους και εξυπηρέτησης των πελατών, όπως φαίνεται στον Πίνακα 4.5.

Πίνακας 4.5. Αποτελέσματα μεταβλητής ανάθεσης X

Δείκτης σημείου i	Δείκτης σημείου j	Δείκτης Οχήματος k	Τιμή Xi,j,k
8	0	1	1
7	0	3	1
6	1	1	1
5	0	4	1
4	8	1	1
3	5	4	1
2	6	1	1
1	2	1	1
0	7	3	1
0	4	1	1
0	3	4	1

Η τιμή 1 στην θέση της μεταβλητής Xi,j,k συμβολίζει την πραγματοποίηση του δρομολογίου από το σημείο i στο σημείο j μέσω του k φορτηγού. Αν και ο αριθμός των πελατών στην πρώτη ομάδα είναι μόνο 8 πελάτες, προκύπτει ότι είναι χρονικά πιο ωφέλιμο να δρομολογηθούν 3 φορτηγά από τα 5 που του δίνονταν να επιλέξει. Επίσης, εφόσον στο συγκεκριμένο πρόβλημα η δρομολόγηση ενός οχήματος δεν προσθέτει κάποιο κόστος, σε όλες τις λύσεις υπάρχουν φορτηγά τα οποία θα εξυπηρετήσουν μόνο έναν ή δύο πελάτες. Στα δρομολόγια αυτά θα αναθέτονται πάντα τα πολύ μικρά φορτηγά του στόλου.

Για την ομάδα αυτή των πελατών η τιμή της αντικειμενικής συνάρτησης είναι 339 λεπτά, δηλαδή χρειάστηκαν 5 ώρες και 39 λεπτά για να εξυπηρετηθούν όλες οι παραγγελίες και να γυρίσουν τα φορτηγά πίσω στην βάση.

Ένα ακόμα παράδειγμα από τα αποτελέσματα που αξίζει να σχολιαστεί, είναι η $7^{η}$ ομάδα πελατών από την περίπτωση των 9 ομάδων πελατών (clusters). Σε αυτήν την ομάδα, τα χρονικά παράθυρα εξυπηρέτησης των πελατών έχουν λίγα κοινά σημεία. Η συγκεκριμένη περίπτωση παρουσιάζει ενδιαφέρον γιατί αν και στα αποτελέσματα θα δρομολογηθούν 7 φορτηγά, το ILOG Studio δυσκολεύτηκε να βρει μία πρώτη εφικτή λύση μέχρις ότου αυξηθεί ο αριθμός των διαθέσιμων φορτηγών σε 10. Αυτό

είναι ένα αντιπροσωπευτικό παράδειγμα για την σημασία των χρονικών παραθύρων σε ένα πρόβλημα VRP αλλά και της δυσκολίας στην επίλυση που φέρνουν. Τα αποτελέσματα της συγκεκριμένης ομάδας παρατίθενται συνοπτικά στους Πίνακες 4.6 και 4.7 .

Πίνακας 4.6. Χρόνοι Δρομολόγησης Οχημάτων 7ης ομάδας πελατών περίπτωσης 9 ομάδων. Στα στοιχεία του πίνακα περιέχονται οι ώρες όπου το φορτηγό έχει φτάσει στον αντίστοιχο πελάτη και έχει ξεκινήσει την διαδικασία εξυπηρέτησης.

ID / Truck #	Truck 1	Truck 2	Truck 4	Truck 6	Truck 7	Truck 8	Truck 9
1	9:03	0	0	0	0	0	0
2	0	0	0	0	6:29	0	0
3	0	0	0	0	6:00	0	0
4	7:45	0	0	0	0	0	0
5	0	8:00	0	0	0	0	0
6	0	0	10:04	0	0	0	0
7	0	0	0	0	0	0	9:12
8	0	0	9:44	0	0	0	0
9	10:12	0	0	0	0	0	0
10	0	0	0	7:55	0	0	0
11	0	0	0	0	0	12:33	0
12	9:44	0	0	0	0	0	0
13	0	0	0	7:00	0	0	0
14	0	0	9:00	0	0	0	0
15	0	8:34	0	0	0	0	0
16	0	0	0	0	0	0	9:35
17	0	0	0	0	0	12:00	0
18	0	0	0	0	0	0	8:30
19	8:45	0	0	0	0	0	0

Πίνακας 4.7. Αποτελέσματα μεταβλητής ανάθεσης X για την περίπτωση της 7ης ομάδας πελατών περίπτωσης 9 ομάδων

Δείκτης σημείου i	Δείκτης σημείου j	Δείκτης Οχήματος k	Τιμή Xi,j,k
0	3	7	1
0	4	1	1
0	5	2	1
0	13	6	1
0	14	4	1
0	17	8	1
0	18	9	1
1	12	1	1
2	0	7	1
3	2	7	1
4	19	1	1
5	15	2	1
6	0	4	1
7	16	9	1
8	6	4	1
9	0	1	1
10	0	6	1
11	0	8	1
12	9	1	1
13	10	6	1
14	8	4	1
15	0	2	1
16	0	9	1
17	11	8	1
18	7	9	1
19	1	1	1

Τέλος, η τιμή της αντικειμενικής συνάρτησης στην περίπτωση αυτή είναι 781 λεπτά, ή αλλιώς 13 ώρες και 1 λεπτό. Τόσος χρόνος απαιτείται αθροιστικά από όλα τα φορτηγά που θα αξιοποιηθούν για την ικανοποίηση των ζητήσεων.

Έχοντας πλέον τρέξει το μοντέλο για όλες τις ομάδες πελατών για την περίπτωση των 8 αλλά και 9 ομάδων, τα αποτελέσματα της αντικειμενικής τους συνάρτησης συνοψίζονται στους πίνακες 4.8 και 4.9 παρακάτω.

Πίνακας 4.8. Αποτελέσματα αντικειμενικών συναρτήσεων προβλημάτων περίπτωσης K=8 ομάδων πελατών.

Ομάδα Πελατών για K=8 ομάδες	Τιμή Αντικειμενικής Συνάρτησης σε λεπτά	Αριθμός Φορτηγών που Χρησιμοποιήθηκαν
1 (8 πελάτες)	339	3
2 (19 πελάτες)	702	5
3 (22 πελάτες)	815	5
4 (4 πελάτες)	139	1
5 (44 πελάτες)	1558	9
6 (23 πελάτες)	780	5
7 (18 πελάτες)	826	4
8 (33 πελάτες)	1223	7
Σύνολο: 171 πελάτες	6482 λεπτά (107 ώρες και 52 λεπτά)	39

Πίνακας 4.9. Αποτελέσματα αντικειμενικών συναρτήσεων προβλημάτων περίπτωσης K=9 ομάδων πελατών.

Ομάδα Πελατών για K=8 ομάδες	Τιμή Αντικειμενικής Συνάρτησης σε λεπτά	Αριθμός Φορτηγών Που Χρησιμοποιήθηκαν
1 (8 πελάτες)	358	2
2 (17 πελάτες)	663	5
3 (9 πελάτες)	411	3
4 (16 πελάτες)	708	4
5 (35 πελάτες)	1427	8
6 (22 πελάτες)	850	6
7 (17 πελάτες)	692	4
8 (30 πελάτες)	1182	7
9 (17 πελάτες)	755	4
Σύνολο: 171 πελάτες	7046 λεπτά (117 ώρες και 47 λεπτά)	41

4.2.3. Συμπεράσματα από την Αναλυτική επίλυση με ομαδοποίηση μέσω του αλγορίθμου K-Means

Από την επίλυση των δύο περιπτώσεων η λύση για οκτώ ομάδες πελατών είναι αυτή που δίνει τα καλύτερα αποτελέσματα ως προς τον χρόνο που θα χρειαστεί για την κάλυψη των ζητήσεων αλλά και ως προς τον αριθμό των φορτηγών που θα χρειαστούν. Και οι δύο λύσεις πάντως, είναι αρκετά καλύτερες σε σχέση με την αναλυτική λύση που είχε πραγματοποιηθεί με μυωπική ομαδοποίηση (8327 λεπτά - 138 ώρες και 40 λεπτά). Αποδεικνύεται και στην πράξη δηλαδή η αξία των αλγορίθμων στην ποιότητα της λύσης σε τέτοια προβλήματα.

4.3. Επίλυση με τον αλγόριθμο του Κοντινότερου Γείτονα (NNA)

Όπως αναφέρθηκε και νωρίτερα, πολλά προβλήματα βελτιστοποίησης εμφανίζουν πρόβλημα στο να βρούν μία καλή λύση σε ικανοποιητικό χρόνο. Γι' αυτό ευθύνεται κατά κύριο λόγο ο τεράστιος όγκος καταστάσεων – λύσεων που πρέπει να μελετηθούν για να ληφθεί μία απόφαση. Τέτοια προβλήματα μπορούν να αντιμετωπιστούν με την χρήση κατάλληλων αλγορίθμων. Ο αλγόριθμος που προτιμήθηκε να εφαρμοστεί στην εργασία αυτή είναι ο αλγόριθμος Nearest Neighbor Algorithm (NNA). Από τη φύση του είναι ένας "άπληστος" αλγόριθμος, εστιάζει δηλαδή στην γρήγορη και αποτελεσματική ανάθεση. Λόγω της φύσεως και της πολυπλοκότητας του προβλήματος CVRP, θα πρέπει πριν την χρήση του να γίνουν οι σωστές παραδοχές. Για την χρήση του αλγόριθμου αυτού σε προβλήματα VRP υπάρχουν δύο περιπτώσεις:

Περίπτωση Α: Στη περίπτωση αυτή δημιουργούμε μία διαδρομή (route) αναθέτοντας με 'άπληστο' τρόπο κάθε φορά το κοντινότερο κόμβο σαν επόμενο σημείο εξυπηρέτησης. Όταν ολοκληρωθεί η δημιουργία της διαδρομής, ένα επιλεγμένο όχημα ξεκινά την διαδικασία δρομολόγησης του. Στη συνέχεια, το πρώτο όχημα με πλήρη χωρητικότητα ξεκινά από το σημείο 0 που αντιστοιχεί στην βάση των οχημάτων και εξυπηρετεί τους κόμβους κατά σειρά της διαδρομής. Όταν ολοκληρώσει την εξυπηρέτηση των πελατών που του έχουν ανατεθεί, το όχημα επιστρέφει στην βάση και δρομολογείται το επόμενο επιλεγμένο όχημα για να συνεχίσει την εξυπηρέτηση από το τελευταίο σημείο της διαδρομής που έχει χαραχθεί στο $1^{ο}$ βήμα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να έχουν ικανοποιηθεί όλες οι ζητήσεις. Η λειτουργία του αλγορίθμου αυτού μπορεί να συνοψιστεί με την εξής διαδικασία:

Βήμα 1: Αρχικοποίηση. Πέρασμα των στοιχείων του προβλήματος όπως ο πίνακας αποστάσεων, χωρητικότητα των οχημάτων και πιθανές ζητήσεις στο πρόγραμμα που έχει επιλεγεί για επίλυση.

Βήμα 2: Υπολογισμός της «γιγαντιαίας διαδρομής» (Hamiltonian Circuit). Με βάση τον NNA, ξεκινάμε από το κέντρο διανομής και υπολογίζουμε τον κοντινότερο κόμβο. Στη συνέχεια από τον επιλεγμένο κόμβο υπολογίζουμε το κοντινότερο κόμβο που δεν του έχει δρομολογηθεί και εισέρχεται στην διαδρομή που χαράσσεται. Η διαδικασία αυτή επαναλαμβάνεται μέχρι την ανάθεση όλων των σημείων στην διαδρομή.

Βήμα 3: **Υπολογισμός του κόστους της γιγαντιαίας διαδρομής.** Με βάση το κριτήριο προς βελτιστοποίηση στο πρόβλημα υπολογίζουμε ποιο είναι το συνολικό κόστος για την διαδρομή.

Βήμα 4: **Επιλογή οχημάτων.** Υπολογίζουμε την συνολική ζήτηση των σημείων της διαδρομής και διαιρούμε με την χωρητικότητα των οχημάτων. Στην περίπτωση που ο στόλος μας παρουσιάζει ετερογένεια ως προς την χωρητικότητα, γίνεται μία παραδοχή ότι θα χρησιμοποιηθούν τα οχήματα με την μεγαλύτερη χωρητικότητα, μέχρι το σημείο που ένα όχημα μικρότερης χωρητικότητας θα μπορεί να καλύψει την ζήτηση των εναπομεινάντων σημείων.

Βήμα 5: Όσο η συνολική ζήτηση είναι μεγαλύτερη του μηδενός, επαναλαμβάνουμε :

Βήμα 5.1: Δρομολόγηση νέου οχήματος για εξυπηρέτηση της γιγαντιαίας διαδρομής μέχρι την εξάντληση της χωρητικότητας του.

Βήμα 5.2: Επιστροφή του οχήματος στο κέντρο διανομής.

Βήμα 5.3: Υπολογισμός κόστους της αντίστοιχης δρομολόγησης και προσθήκη της στο συνολικό κόστος.

Βήμα 5.4: Επανάληψη ελέγχου υπόλοιπων ζητήσεων.

Η περίπτωση του αλγόριθμου αυτού χρησιμοποιείται κυρίως όταν δίνεται βάση στην γρήγορη εύρεση αποτελεσμάτων, με ρίσκο όμως ως προς την ποιότητα της λύσης. Όταν όμως η πολυπλοκότητα του προβλήματος είναι μεγάλη (για παράδειγμα Time-Windows), και η ποιότητα της λύσης έχει αρκετή σημασία, όπως σε αυτό το πρόβλημα, προτιμάται η δεύτερη μορφή του αλγορίθμου που αναλύεται παρακάτω.

Περίπτωση Β: Η δεύτερη παραλλαγή του αλγορίθμου δίνει ελαφρώς μικρότερη βάση στην ταχύτητα της επίλυσης και εστιάζει στην βελτίωση της ποιότητας των αποτελεσμάτων. Τώρα ο αλγόριθμος NNA και οι ιδιαίτεροι περιορισμοί του προβλήματος χρησιμοποιούνται παράλληλα. Στη περίπτωση του το πρώτο όχημα ξεκινάει από το κέντρο διανομής και ακολουθεί τακτική εξυπηρέτησης κοντινότερων σημείων. Μία αλλαγή που εμφανίζει είναι στον ορισμό του κοντινότερου σημείου. Λόγω των time-windows, κατά τον υπολογισμό του επόμενου κοντινότερου πελάτη, εάν το όχημα θα πρέπει να περιμένει το «άνοιγμα» του χρονικού παραθύρου εξυπηρέτησης του, τότε εξετάζεται το ενδεχόμενο να συμφέρει χρονικά να εξυπηρετηθεί ο επόμενος κοντινότερος πελάτης. Οι μη εξυπηρετούμενοι πελάτες ή οι πελάτες που η

ζήτηση τους έχει ικανοποιηθεί μερικώς (εάν το επιτρέπει το πρόβλημα), εξετάζονται για το επόμενο όχημα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να έχει γίνει ανάθεση οχήματος για όλους τους πελάτες. Η λειτουργία του αλγορίθμου αυτού μπορεί να συνοψιστεί με τα παρακάτω βήματα:

Βήμα 1: Αρχικοποίηση. Πέρασμα των στοιχείων του προβλήματος, όπως ο πίνακας αποστάσεων, χωρητικότητα των οχημάτων και πιθανές ζητήσεις στο πρόγραμμα που έχει επιλεγεί για επίλυση.

Βήμα 2: Υπολογισμός της ζήτησης των σημείων. Όσο η ανικανοποίητη ζήτηση παραμένει μεγαλύτερη του μηδενός, επαναλαμβάνουμε τα εξής :

Βήμα 2.1: Επιλογή οχήματος για δρομολόγηση.

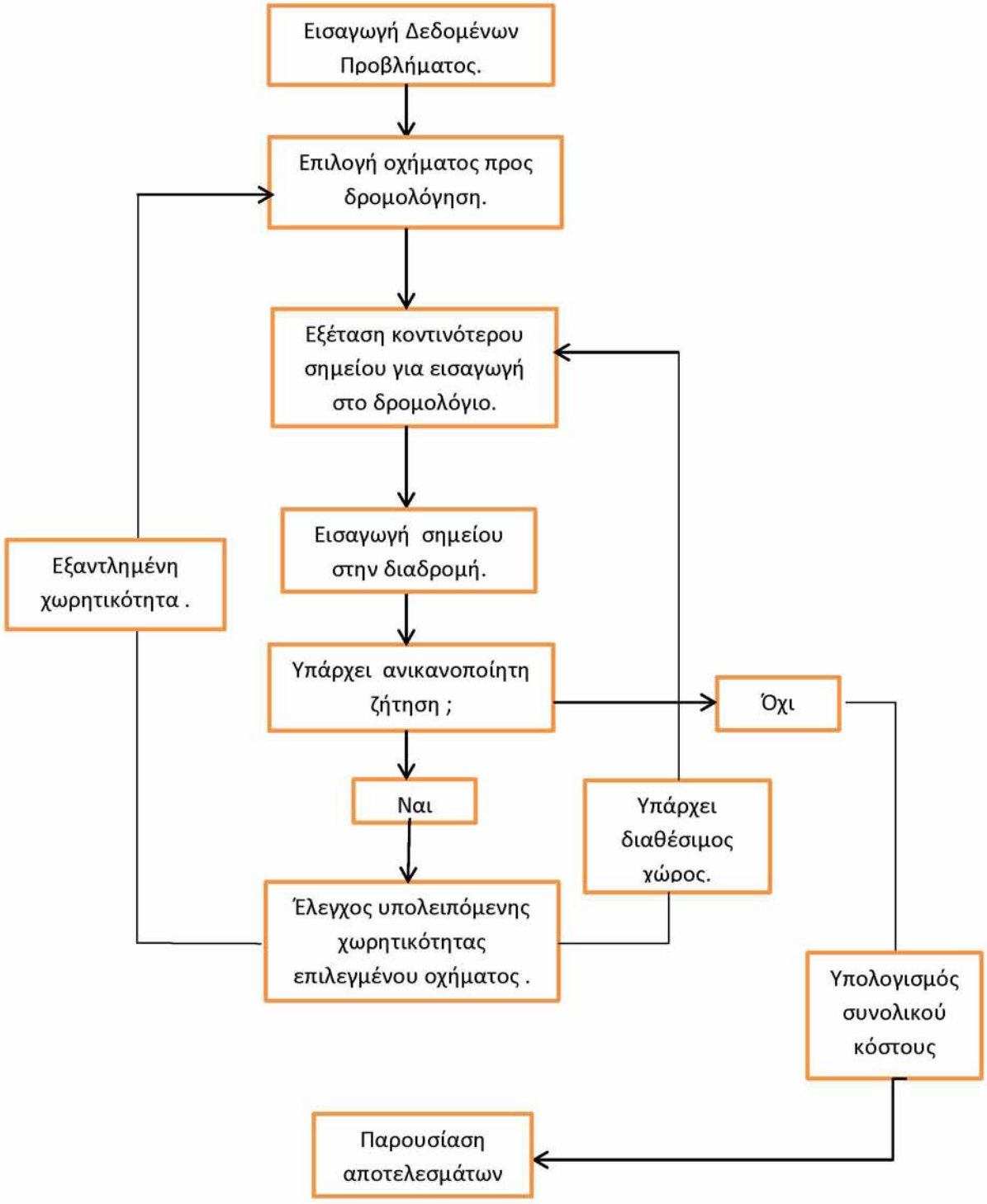
Βήμα 2.2: Υπολογισμός του κοντινότερου σημείου. Στην περίπτωση που το σημείο αυτό δεν μπορεί να εξυπηρετηθεί σε κοινό χρονικό παράθυρο με το προηγούμενο του σημείο, εξετάζεται το ενδεχόμενο να συμφέρει η τοποθέτηση ενός άλλου κοντινού σημείου στο δρομολόγιο. Η επιλογή του χρόνου έναρξης της διαδικασίας εξυπηρέτησης είναι ο μέσος χρόνος «ανοίγματος» των Time-Windows των πελατών που είναι διαθέσιμοι για εξυπηρέτηση για λιγότερο από 12 ώρες στην διάρκεια μιας ημέρας.

Βήμα 2.3: Όταν βρεθεί το σημείο αυτό, εξετάζεται η μη παραβίαση της συνολικής χωρητικότητας του οχήματος.

Βήμα 2.4: Όταν η διαθέσιμη χωρητικότητα του ανατιθέμενου οχήματος εξαντληθεί (ή έστω ένα ποσοστό της τάξης του 90%), το όχημα ξεκινάει την διαδικασία εξυπηρέτησης του δρομολογίου που έχει δημιουργηθεί, ενώ οι πελάτες που θα εξυπηρετηθούν από αυτό διαγράφονται από την λίστα της ανικανοποίητης ζήτησης.

Βήμα 3: Υπολογισμός του συνολικού κόστους των δρομολογίων που θα δημιουργηθούν.

Η διαδικασία επίλυσης του προβλήματος σε αυτή τη περίπτωση περιγράφεται και από το Σχήμα 4.5 παρακάτω.



Σχήμα 4.5. Σχηματική απεικόνιση του αλγορίθμου της περίπτωσης Β.

Μπορεί και η παραλλαγή του αλγορίθμου αυτού να ανήκει στην κατηγορία των «άπληστων» αλγορίθμων. Όμως δίνει αρκετή προσοχή στην εύρεση πιο ικανοποιητικών λύσεων σε σχέση με την περίπτωση Α όπως εξετάστηκε από τους Kulkarni and Sohani, 2014. Γι' αυτό το λόγο έγινε και η επιλογή αυτής της περίπτωσης για εφαρμογή. Στην επίλυση αυτή όμως δεν είναι δυνατόν να πάρει κανείς ιδανικά αποτελέσματα για την ανάθεση των φορτηγών καθώς δεν υπάρχει κάποιο κριτήριο για την επιλογή φορτηγών ή κάποια επιβάρυνση σε πιθανή ανάθεση. Έτσι σε αυτό τον αλγόριθμο θα επιλέξουμε εμείς τον αριθμό των φορτηγών που θα είναι διαθέσιμα ανάλογα με το μέγεθος της ομάδας και της παραγγελίας, μειώνοντας έτσι το ποσοστό του κενού φορτίου που στις προηγούμενες λύσεις ήταν μεγάλο. Τα φορτηγά που θα διατεθούν θα είναι μικρής και μεσαίας χωρητικότητας ώστε να υπάρξει πολλαπλή δρομολόγηση, καθώς υπάρχουν φορτηγά διαθέσιμα που μπορούν μόνα τους να εξυπηρετήσουν και την πιο μεγάλη ομάδα. Στην περίπτωση όμως ανάθεσης ενός μόνο φορτηγού σε κάθε ομάδα θα άλλαζε την μορφή του προβλήματος μας σε πρόβλημα πλανόδιου πωλητή (TSP). Για την ανάπτυξη του αλγόριθμου αυτού προτιμήθηκε η χρήση της πλατφόρμας της Matlab 2018b.

Στο Πίνακα 4.10 παρουσιάζονται τα δεδομένα που εισάγουμε στον αλγόριθμο ενώ στο Πίνακα 4.11 παρουσιάζεται η ανάθεση που πραγματοποιείται.

Πίνακας 4.10. Δεδομένα για επίλυση μέσω του αλγόριθμου.

Ομάδα πελατών	Συνολική Ζήτηση Ομάδας (σε βάρος)	Συνολική Ζήτηση Ομάδας (σε όγκο)	Αριθμός πελατών στην ομάδα	Αριθμός διαθέσιμων φορτηγών
1	2748	176	8	1
2	6899	466	19	2
3	5482	3023	22	2
4	3058	216	4	1
5	19155	1662	44	4
6	10888	890	23	4
7	10914	857	18	4
8	20073	1835	33	4

Πίνακας 4.11. Αποτελέσματα ομάδων μέσω χρήση του αλγορίθμου ΝΝΑ.

Ομάδα πελατών.	Αριθμός πελατών στην ομάδα.	Αριθμός χρησιμοποιηθέντων φορτηγών.	Σειρά δρομολόγησης εξυπηρέτησης.
1	8	1	0-4-8-1-6-2-3-5-7-1-0
2	19	2	0-3-1-2-17-7-11-5-18-19-0 0-8-9-12-13-4-15-6-16-14-10-0
3	22	2	0-3-4-19-20-21-22-5-13-14-15-16-17-18-0 0-11-6-12-1-2-9-7-8-10-0
4	4	1	0-1-2-4-3-0
5	44	4	0-14-19-22-43-42-7-11-8-18-29-17-0 0-2-3-4-12-32-36-16-13-20-1-9-0 0-5-1-26-27-30-6-33-34-38-39-10-0 0-21-23-24-25-44-31-40-41-28-35-15
6	23	3	0-4-2-4-5-19-14-12-18-0 0-16-15-1-8-13-23-3-22-0 0-20-7-17-9-21-10-11-0
7	18	2	0-10-4-5-15-18-8-17-6-11-13-12-14-0 0-9-7-16-3-2-1-0
8	33	4	0-9-22-23-31-11-24-33-29-30-0 0-6-5-15-17-19-21-20-0 0-3-4-8-12-32-25-27-26-0 0-7-1-2-28-7-13-14-10-18-0

4.3.1. Συμπεράσματα της επίλυσης με αλγόριθμο κοντινότερου γείτονα

Από την αρχή της επίλυσης αυτής ήταν αναμενόμενο τα αποτελέσματα να παρουσιάζουν αύξηση στον χρόνο εξυπηρέτησης. Η λύση με την διαδικασία του μεικτού ακέραιου προγραμματισμού θα βρίσκει τα βέλτιστα αποτελέσματα σε αριθμητικό επίπεδο. Ο χρόνος που χρειάζεται για την πλήρη κάλυψη των ζητήσεων στην περίπτωση της λύσης με τον αλγόριθμο ΝΝΑ είναι αυξημένος σχεδόν κατά 27%. Για την αναμενόμενη αύξηση αυτήν ευθύνονται δύο αλλαγές.

Η πρώτη είναι η φύση των «άπληστων» αλγορίθμων, η εφαρμογή των οποίων ελαχιστοποιεί τον χρόνο επίλυσης θυσιάζοντας ένα κομμάτι από την βελτιστότητα της λύσης. Σε πολλές εφαρμογές και προβλήματα η ταχύτητα εύρεσης ικανοποιητικών λύσεων αποτελεί πρώτο στόχο. Έτσι και εδώ η επίλυση πραγματοποιήθηκε σε χρόνο μειωμένο σε ένα ποσοστό της τάξης του 50%.

Η δεύτερη αλλαγή που προκάλεσε την αύξηση του χρόνου εξυπηρέτησης είναι η ανάθεση του αριθμού των φορτηγών σε κάθε υπό-ομάδα από εμάς. Στις πρώτες λύσεις με τις εφαρμογές του μικτού ακέραιου προγραμματισμού λόγω της φύσεως του προβλήματος και των δεδομένων υπήρξε μία θεωρητική

βελτιστότητα. Εδώ εμφανίζεται η σημαντικότητα της ικανότητας για ευελιξία στις λύσεις αλλά και για συμβιβασμούς ώστε η «μαθηματική βελτιστότητα» να δίνει αποτελέσματα εφικτά καθώς υπάρχει πάντα και ο ανθρώπινος παράγοντας που πρέπει να λαμβάνεται υπόψη.

4.4. Επίλυση με ομαδοποίηση δεδομένων βάσει των παραθύρων εξυπηρέτησης

Όπως αναφέρθηκε και στην αρχή αυτού του κεφαλαίου, αναλόγως της φύσης του προβλήματος που αντιμετωπίζεται κάθε φορά θα εφαρμόζονται και διαφορετικές διαδικασίες ομαδοποίησης. Στο σημείο αυτό θα εφαρμοστεί ένας ακόμα αλγόριθμος πάνω στο clustering των πελατών. Εδώ αξίζει να σημειωθεί πως ο αλγόριθμος K-Means που χρησιμοποιήθηκε μέχρι τώρα προτείνεται σαν ιδανικός για ομαδοποίηση ως προς την τοποθεσία σημείων, αρκεί ο χώρος που μελετάμε να μην είναι υπερβολικά μεγάλος (πχ για σημεία σε όλη την Ευρώπη). Αυτό συμβαίνει λόγω του ότι ο αλγόριθμος αυτός παίρνει Ευκλείδειες αποστάσεις ενώ για μεγάλες αποστάσεις υπεισέρχεται η σφαιρικότητα του πλανήτη.

Έτσι για τελευταίο πείραμα αποφασίστηκε να δοκιμαστεί μια ομαδοποίηση των σημείων με βάση νέα και διαφορετικά κριτήρια αντί για την τοποθεσία. Επιλέχθηκε έτσι να εφαρμοστεί η αρχή λειτουργίας του αλγορίθμου Διάδοσης Συγγένειας (Affinity Propagation). Στην επιστήμη της στατιστικής, ο αλγόριθμος αυτός βασίζεται στην έννοια της «διάδοσης μηνύματος» μεταξύ των σημείων των δεδομένων που θέλουμε να ομαδοποιήσουμε. Ο σκοπός αυτών των «μηνυμάτων» είναι να περιγραφεί η καταλληλόλητα των σημείων που θα χρησιμοποιηθούν σαν υπόδειγμα. Τα υποδείγματα είναι σημεία των δεδομένων που περιγράφουν ιδανικότερα τα υπόλοιπα σημεία της ομάδας τους και είναι πρακτικά τα πιο σημαντικά σε μία ομάδα δεδομένων - cluster. Κάθε ομάδα χαρακτηρίζεται από μόνο ένα υπόδειγμα. Παρακάτω περιγράφεται η διαδικασία λειτουργίας του και οι απαραίτητες αποδοχές που εφαρμόστηκαν για την λειτουργία του στο πρόβλημα VRP που αντιμετωπίζουμε.

Για δεδομένα εισόδου ενός πρότυπου αλγόριθμου διάδοσης οικογένειας είναι συνήθως :

- Ο πίνακας ευθύνης (responsibility matrix) R(I,k), ο οποίος περιγράφει την καταλληλόλητα ενός σημείου k να χρησιμοποιηθεί σαν υπόδειγμα για το σημείο i.

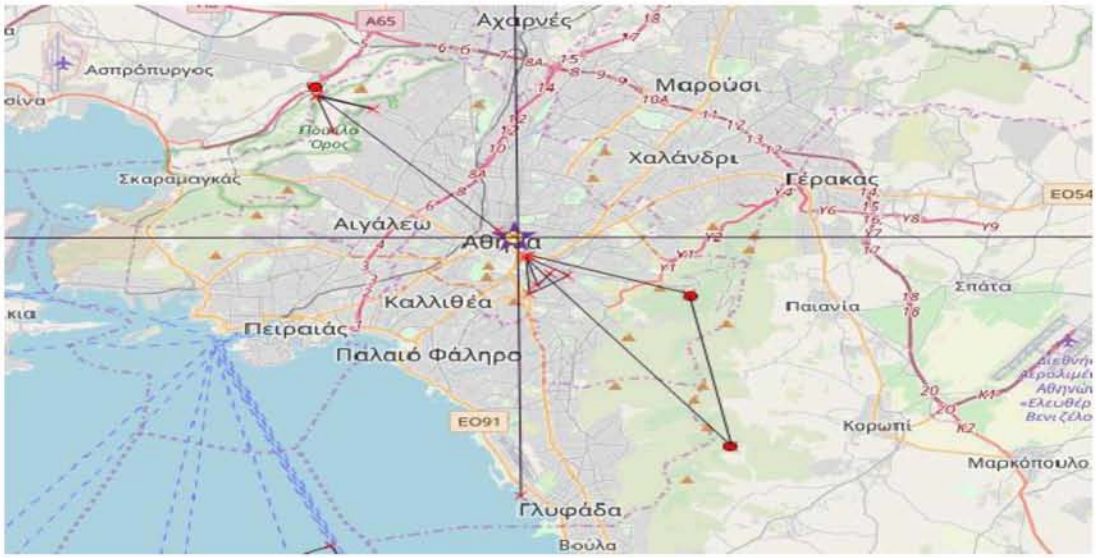
- Ο πίνακας διαθεσιμότητας (availability matrix) A(I,k) που αντίστοιχα εκφράζει το πόσο κατάλληλο μπορεί να είναι η επιλογή του k σημείου από το i σαν υπόδειγμα.

Όπως αναφέρθηκε, η ομαδοποίηση των δεδομένων μας θέλουμε να γίνει με έναν διαφορετικό κριτήριο από αυτό της τοποθεσίας. Έτσι, επειδή οι πελάτες του προβλήματος έχουν χρονικά παράθυρα μέσα στα οποία εξυπηρετούνται, θα επιδιώξουμε να δημιουργήσουμε τις συναρτήσεις και τους πίνακες ομοιότητας

με βάση τα κοινά διαστήματα των χρονικών παραθύρων. Ανάλογα δηλαδή με τους κοινούς χρόνους για τους οποίους εξυπηρετείται ο κάθε πελάτης σε συνδυασμό με το χρονικό διάστημα που είναι διαθέσιμος.

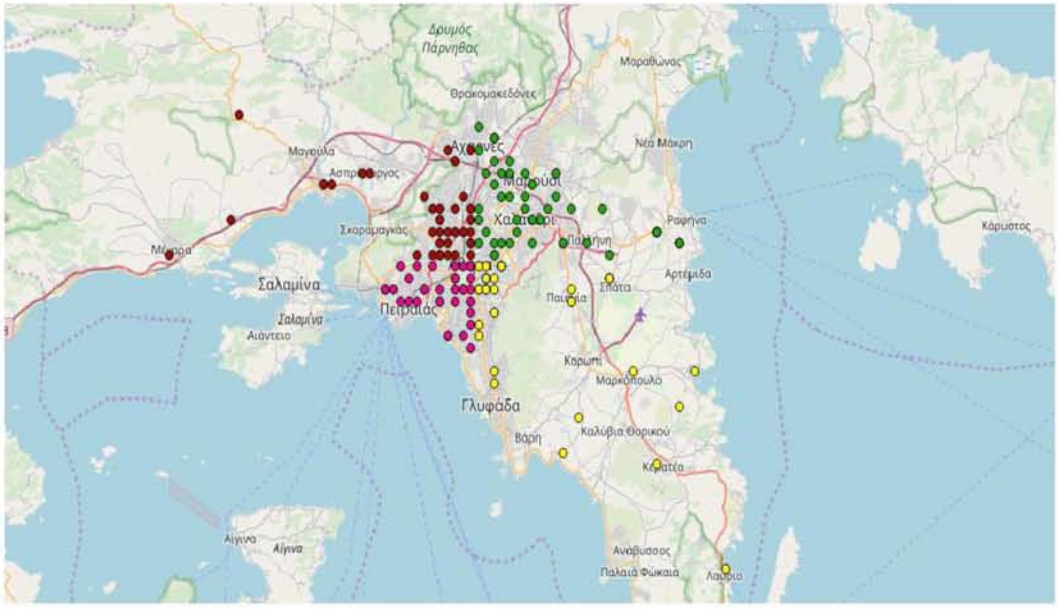
Ξεκινώντας την διαδικασία της ομαδοποίησης με βάση τα κοινά χρονικά διαστήματα εξυπηρέτησης, πρέπει να δοθεί προσοχή σε ένα πολύ λεπτό σημείο της φύσης του προβλήματος που θα εξηγηθεί στο παράδειγμα που ακολουθεί.

Ας υποθέσουμε τα σημεία που απεικονίζονται στον χάρτη του Σχήματος 4.6 με κόκκινες κουκίδες εκφράζουν τρείς πελάτες. Έστω ότι οι πελάτες αυτοί έχουν πλήρη ταύτιση στους χρόνους που δέχονται επισκέψεις. Συνεπώς, σύμφωνα με τα στοιχεία που λαμβάνουμε μέχρι τώρα υπόψη για την ομαδοποίηση, οι πελάτες αυτοί θα ανήκουν στην ίδια ομάδα. Με μπλε χρώμα απεικονίζεται η τοποθεσία της αποθήκης από την οποία ξεκινάνε οι αποστολές των οχημάτων. Είναι εμφανές ότι παρόλο που τα χρονικά παράθυρα συμπίπτουν πλήρως μεταξύ τους, η τοποθέτηση αυτών των δύο σημείων στην ίδια ομάδα και πιθανόν στο ίδιο δρομολόγιο θα επιφέρει μεγάλη καθυστέρηση στην εξυπηρέτηση τους με ένα όχημα. Συνεπώς πέρα από τα χρονικά παράθυρα είναι επιθυμητό να έχει γίνει ένας πρώτος διαχωρισμός των σημείων ανάλογα και με τη θέση τους ως προς τη βάση. Το πρόβλημα αυτό έχει μελετηθεί και από τους Gunadi and Alinda το 2002, όπου εφαρμόστηκε και ένας άπληστος αλγόριθμος «σκουπίσματος» (Sweep Algorithm) για να βελτιωθεί η ποιότητα της ομαδοποίησης. Ακολουθώντας την αρχή του αλγορίθμου αυτού γίνεται ένας γεωγραφικός διαχωρισμός των σημείων- πελατών. Αυτός ο διαχωρισμός θα γίνει σε τέσσερα κομμάτια: το βορειοανατολικό, το βορειοδυτικό, το νοτιοανατολικό και το νοτιοδυτικό τεταρτημόριο με κέντρο όλων να είναι η θέση της αποθήκης και οι γραμμές που διαχωρίζουν τα σημεία να είναι οι κάθετοι στο γεωγραφικό σύστημα συντεταγμένων που χρησιμοποιούμε (WGS 84).



Σχήμα 4.6. Παράδειγμα μη συμφέρουσας ομαδοποίησης.

Παίρνουμε έτσι τις τέσσερις ομάδες όπως απεικονίζονται στο Σχήμα 4.7 με διαφορετικό χρώμα οι πελάτες της κάθε ομάδας. Επίσης οι πελάτες της κάθε ομάδας παρουσιάζονται στο Πίνακα 4.12



Σχήμα 4.7. Γεωγραφική ομαδοποίηση ανάλογα την θέση ως προς την αποθήκη.

Πίνακας 4.12. Ομάδες Πελατών ανάλογα την θέση τους ως προς την αποθήκη.

Τεταρτημόριο Ομάδας	Πελάτες Ομάδας
Βορειανατολικό	2,3,6,9,10,11,13,16,41,44,48,52,53,55,56,57,61,64,69,70,71,77,79,80,81,105,106,109,112,119,120,121,127,129,130,134,135,136,141,142,145,146,148,149,150,156,157,159,162,163,164,165, 166,167
Βορειοδυτικό	12,14,15,24,25,26,28,47,54,58,59,60,62,66,68,72,74,75,76,90,91,94,95,96,97,99,102,103,108,111,114,115,122,125,131,132,133,139,140,144,151,152,153,160,161
Νοτιοανατολικό	4,5,7,17,18,19,20,21,22,27,30,45,46,63,78,82,83,84,92,93,98,101,104,107,113,116,118,123,124,126,154,158,168,169,170,171
Νοτιοδυτικό	1,8,23,29,31,32,33,34,35,36,37,38,39,40,42,43,49,50,51,65,67,73,85,86,87,88,89,100,110,117,128,137,138,143,147

Τώρα πρέπει να εφαρμόσουμε την μεθοδολογία του αλγορίθμου διάδοσης συγγένειας σε κάθε μια ομάδα - τεταρτημόριο ξεχωριστά. Για να γίνει αυτό, πρέπει να εκφράζουμε τις συναρτήσεις που θα εκφράζουν τον βαθμό συγγένειας των πελατών λαμβάνοντας υπόψη τα δεδομένα της κάθε ομάδας. Τα στατιστικά στοιχεία της κάθε ομάδας που θα χρησιμοποιηθούν στην ομαδοποίηση παρουσιάζονται στον Πίνακα 4.13.

Πίνακας 4.13. Στατιστικά στοιχεία Γεωγραφικών Ομάδων Πελατών.

Γεωγραφική Ομάδα Πελατών	Μέσος χρόνος ανοίγματος Time-Window	Μέσος χρόνος κλεισίματος Time-Window	Μέσος χρόνος εξυπηρέτησης	Αριθμός μελών ομάδας
Βορειοανατολική	5:48	17:33	24 λεπτά	54
Βορειοδυτική	5:57	17:24	23,5 λεπτά	45
Νοτιοανατολική	6:52	15:48	19,1 λεπτά	37
Νοτιοδυτική	7:30	14:48	28 λεπτά	35

Για αρχή πρέπει να χαρακτηρίσουμε τα στοιχεία που θα ληφθούν υπόψη στην επιλογή των πελατών – υποδειγμάτων των ομάδων. Στο σημείο αυτό απαιτείται μελέτη του στοιχείου ως προς το οποίο θα γίνει η ομαδοποίηση. Εδώ παρατηρείται πως κατά κύριο λόγο τα ωράρια των Time-Windows διακρίνονται σε μικρά (4 με 5 ώρες ανοικτού παραθύρου) τα οποία τείνουν να ξεκινούν τις πρωινές ώρες, στα πιο ευέλικτα που λειτουργούν κατά κύριο λόγο πιο αργά το πρωί έως το απόγευμα και τέλος από κάποια πιο αυθαίρετα που δεν ακολουθούν κάποιο συγκεκριμένο μοτίβο. Οι δύο πρώτες κατηγορίες χαρακτηρίζουν το 65% του πλήθους των πελατών και έτσι δεχόμαστε ένα σημείο από τον καθένα σαν σημείο υπόδειγμα. Στην συνέχεια, πραγματοποιείται ένταξη των υπόλοιπων πελατών στην ομάδα που τους χαρακτηρίζει περισσότερο μέσω συγκρίσεων των στοιχείων τους με αυτά των σημείων - υποδειγμάτων αλλά και των μέσων τιμών της κάθε γεωγραφικής ομάδας ενώ οι πελάτες με ανοιχτό ωράριο εξυπηρέτησης (24 ώρες) κατανέμονται αρχικά και στις δύο ομάδες πριν εξεταστεί η καταλληλότητα τους με κριτήριο την ομάδα των περισσότερων γειτονικών τους σημείων.

Η συνάρτηση ομοιότητας των πελατών εκφράζει το κοινό χρονικό παράθυρο που έχουν οι πελάτες με τα σημεία – υποδείγματα της κάθε τετράδας. Ανάλογα, το κοινό διάστημα αυτό και το άνω και κάτω όριο του κάθε παραθύρου εξυπηρέτησης οι πελάτες εισέρχονται στην ομάδα που τους χαρακτηρίζει.

Η συνάρτηση επιλογής των πελατών με χρονικό παράθυρο περιγράφεται ως εξής:

Similarity { i client with k exemplar} όπου i ο πελάτης που ελέγχεται και k το υπόδειγμα που εκπροσωπεί την ομάδα.

Similarity(i,k) = {MIN(TWend[i], TWend[k]) – Max(TWbegin[i] , TWbegin[k]) – Tdelay[i] } / (TWend[k] – Twbegin[k])

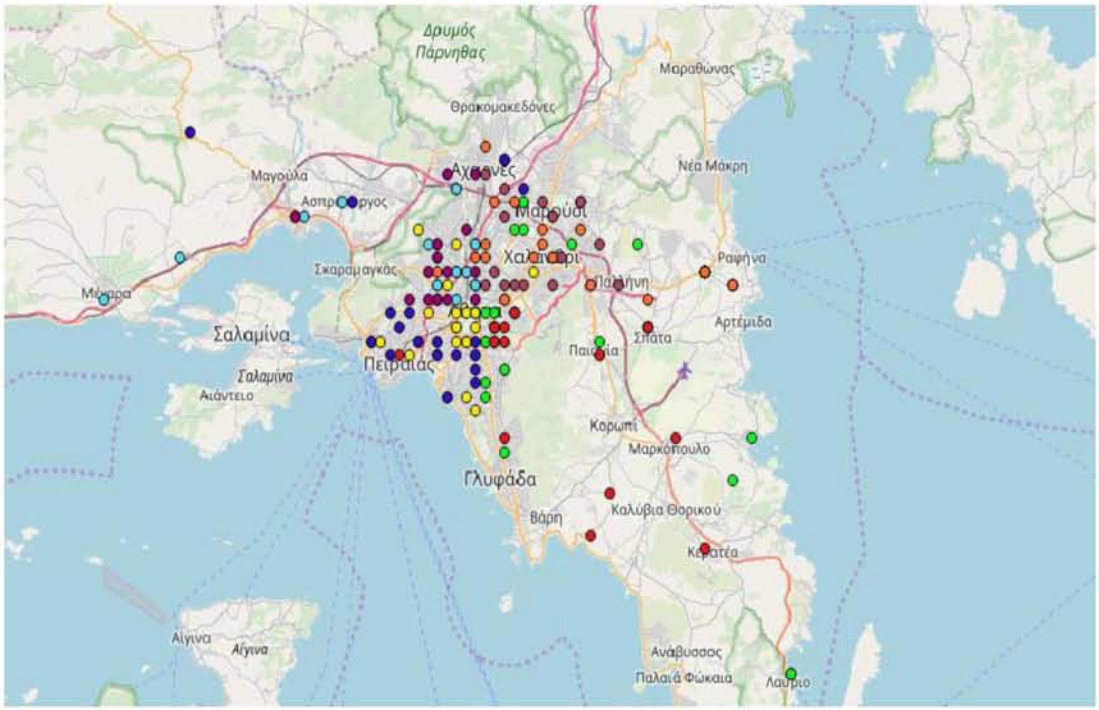
Όπου :

- TWend ο χρόνος λήξης του χρονικού παραθύρου ενός πελάτη.
- TWbegin ο χρόνος ανοίγματος του χρονικού παραθύρου ενός πελάτη.

Ανάλογα το ποσοστό συμμετοχής και κοινού χρόνου με τις ομάδες υποδείγματα οι πελάτες τοποθετούνται στην ομάδα που τους αντιστοιχεί. Στο Πίνακα 4.14 παρουσιάζονται τα αποτελέσματα της ολοκληρωμένης ομαδοποίησης που περιγράψαμε, ενώ στο Σχήμα 4.8 δίνεται και η εικόνα των νέων clusters που θα χρησιμοποιηθούν.

Πίνακας 4.14. Πίνακας ομάδων-clusters μετά την εφαρμογή του αλγορίθμου.

Ομάδα Πελατών	Γεωγραφική Ομάδα Τοποθεσίας πελατών.	ID πελατών ομάδας.	Αριθμός Πελατών
1	Βορειοανατολικό	10,13,48,52,53,55,56,57,61,64,69,70,71,77,79,81,105,112,156,157, 159, 162, 163, 164, 165, 166, 167	28
2	Βορειοανατολικό	2,3,6,9,11,16,41,44,80, 106,109, 119, 120, 121, 127,129,130,134,135,136,141,142,145,146,148, 149,150	26
3	Βορειοδυτικό	12, 14, 15, 28, 47,90, 91, 94, 95, 96,97, 99 ,102, 103 ,108, 114,115,131, 132, 139,152,153, 140, 144	24
4	Βορειοδυτικό	24, 25 ,26,54, 58 ,59, 60, 62, 66, 68, 72, 74, 75, 76, 111,133, 151, 160, 161,122,125	21
5	Νοτιοανατολικό	4,5,7,46,30,82,83,84,93,98,101,107,113,116, 118,123,126,155	18
6	Νοτιοανατολικό	17,18,19,20,21,22,27,104,92,124,154,158,168, 169,170,171,45,63,78	19
7	Νοτιοδυτικό	1,8,31,32,34,35,100,110,117,42,40,43,51,85, 138,85	16
8	Νοτιοδυτικό	23,29,33,86,87,88,137,128,89,36,37,38,39,65, 67,73,50,143,147,49	19



Σχήμα 4.8. Αποτελέσματα της διαδικασίας ομαδοποίησης με βάση την θέση ως προς το κέντρο διανομής και των χρονικών παραθύρων εξυπηρέτησης.

Με την ολοκλήρωση της δημιουργίας ομάδων με διαφορετικό πλέον κριτήριο, πραγματοποιείται επίλυση του μοντέλου που έχει αναπτυχθεί με τον solver της Cplex. Τα αποτελέσματα επιτυγχάνονται παρατίθενται στο Πίνακα 4.15 που ακολουθεί παρακάτω.

Πίνακας 4.15. Αποτελέσματα Λύσης με ομαδοποίηση πελατών ως προς τα χρονικά παράθυρα.

Ομάδα Πελατών.	Τιμή Αντικειμενικής Συνάρτησης Σε Λεπτά.	Αριθμός Πελατών Στην Ομάδα.	Αριθμός Φορτηγών Που Χρησιμοποιήθηκαν.
1	1044	28	5
2	876	26	5
3	894	24	4
4	769	21	3
5	757	18	4
6	701	19	4
7	856	16	4
8	852	19	2
Σύνολο	6619 λεπτά (112 ώρες και 33 λεπτά)	171	31 φορτηγά

4.4.1 Συμπεράσματα από την αναλυτική επίλυση με ομαδοποίηση δεδομένων βάση χρονικών παραθύρων

Τα αποτελέσματα που έδωσε η επίλυση του μοντέλου μας συγκριτικά με τα βέλτιστα αποτελέσματα που έχουν βρεθεί μέχρι τώρα παρουσιάζουν δύο κομβικά σημεία που είναι άξια σχολιασμού. Πρώτο σημείο είναι το αριθμητικό αποτέλεσμα της αντικειμενικής συνάρτησης που είναι 112 ώρες και 33 λεπτά. Το αποτέλεσμα αυτό μπορεί να χαρακτηριστεί ικανοποιητικό και ευθύνεται σε έναν βαθμό στον διαχωρισμό των δεδομένων σε γεωγραφικά τεταρτημόρια που έγινε σε πρώτο βήμα πριν την ομαδοποίηση τους βάσει χρονικών παραθύρων. Απέχει κατά σχεδόν 5 εργατοώρες από το βέλτιστο που είχε βρεθεί στην βέλτιστη λύση όμως παρουσιάζει μία σημαντική διαφορά: Η διαφορά αυτή είναι το δεύτερο σημείο που αξίζει σχολιασμό και είναι η μείωση των φορτηγών που δρομολογούνται από 39 σε 31. Στην μείωση αυτή συνέβαλε κατά κύριο λόγω το γεγονός ότι στην επίλυση αυτή οι χρόνοι αναμονής ενός οχήματος για άνοιγμα παραθύρου (δίνονται από την μεταβλητή απόφασης TWijk του μοντέλου) ενός πελάτη είναι ανύπαρκτοι και δεν προωθείται η δρομολόγηση οχημάτων για έναν ή δύο πελάτες όπως γινόταν πριν. Η λήψη μιας ολοκληρωμένης απόφασης λοιπόν για την λύση που θα προτιμηθεί από μία εταιρία θα κριθεί από τα κόστη των εργατοωρών και την πιθανή ύπαρξη επιπλέον δαπανών για δρομολόγηση φορτηγών.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1. Γενικά Συμπεράσματα

Στα πλαίσια αυτής της εργασίας έγινε μια προσπάθεια να λυθεί και να αντιμετωπιστεί ένα αρκετά περίπλοκο πρόβλημα επιχειρησιακής έρευνας, αυτό της δρομολόγησης οχημάτων για την ικανοποίηση της ζήτησης ενός πλήθους πελατών με πολλαπλά οχήματα. Η κατηγορία των προβλημάτων VRP είναι όντως από τα πιο δύσκολα προβλήματα να αντιμετωπιστούν καθώς στο καθένα από αυτά πρέπει να πραγματοποιούνται παραδοχές ώστε να προσεγγιστεί η βελτιστότητα.

Στο συγκεκριμένο πρόβλημα, ο βαθμός δυσκολίας είναι αρκετά υψηλός καθώς πέρα από την περιορισμένη χωρητικότητα των οχημάτων ως προς δύο κατηγορίες (βάρος και όγκος προϊόντων) πρέπει να προστεθεί και η διάσταση του χρόνου λόγω των χρονικών παραθύρων εξυπηρέτησης του κάθε πελάτη. Επίσης, το γεγονός ότι το κριτήριο βελτιστότητας των λύσεων είναι ο συνολικός χρόνος εξυπηρέτησης των πελατών υποχρεώνει να υπολογιστούν επιπλέον παράμετροι καθώς σε κάποιες περιπτώσεις είναι αρκετά πιο συμφέρον ένα όχημα να περιμένει ακινητοποιημένο το άνοιγμα ενός χρονικού περιθωρίου κάποιου πελάτη. Στην εργασία αυτή αποτυπώθηκαν τέσσερις διαφορετικοί τρόποι επίλυσης, με τον κάθε έναν να προσφέρει διαφορετικά και αξιόλογα αποτελέσματα.

Η πρώτη επίλυση πραγματοποιήθηκε με βάση την αναλυτική λύση που προέκυψε από την μορφοποίηση του μοντέλου. Στην περίπτωση αυτή λόγω του όγκου των δεδομένων έγινε μία ομαδοποίηση των πελατών χωρίς χρήση κάποιου αλγορίθμου και με μοναδικό κριτήριο τον αριθμό-δείκτη του κάθε πελάτη. Η λύση αυτή θα μπορούσε να χαρακτηριστεί και ως πρώτη προσέγγιση στην επίλυση του προβλήματος. Το συνολικό αποτέλεσμα είναι 8327 λεπτά, δηλαδή 138 εργατοώρες να χρειάζονται μέσα σε μία ημέρα για την κάλυψη όλων των ζητήσεων. Στο σημείο αυτό, αν και υπήρξε λύση έγινε εμφανής η ανάγκη εμφάνισης των ευρετικών και μετ-ευρετικών μεθόδων.

Η δεύτερη λύση αποτελεί κομμάτι βελτίωσης της πρώτης, καθώς ύστερα από μελέτη αντίστοιχων περιπτώσεων προβλημάτων επιλέχθηκε η εφαρμογή δοκιμασμένου αλγορίθμου ομαδοποίησης, του K-Means. Πλέον η εισαγωγή των πελατών σε ομάδες δεν θα γίνεται με αυθαίρετο τρόπο. Το μοντέλο του προβλήματος παρέμεινε ίδιο και μετά από κάποιες δοκιμές και πειράματα βρέθηκε ο ιδανικός αριθμός ομάδων καθώς και τα καινούργια σύνολα. Έτσι με οκτώ ομάδες πελατών πλέον φτιαγμένες αλγοριθμικά η βελτίωση των αποτελεσμάτων ήταν αισθητή και έπεσε στις 107 ώρες. Παρόλο που η λύση αυτή είναι που έδωσε και τα καλύτερα αποτελέσματα με βάση την αντικειμενική συνάρτηση και το κριτήριο του χρόνου, παρατηρείται η δρομολόγηση 39 φορτηγών του στόλου. Σε αυτό ευθύνεται αποκλειστικά το γεγονός πως δεν υπάρχει κάποια επιβολή κόστους σε πιθανές δρομολογήσεις πολλαπλών οχημάτων ούτε

ζήτηση για ελαχιστοποίηση των άδειων φορτίων. Έτσι δόθηκε η αφορμή για μία ακόμα προσπάθεια επίλυσης.

Στην τρίτη περίπτωση έγινε μία διαφορετική προσέγγιση του προβλήματος. Για αρχή επιλέχθηκε το μοντέλο του αλγορίθμου του κοντινότερου γείτονα (Nearest Neighbor Algorithm). Πραγματοποιήθηκαν κάποιες αλλαγές στην δομή του ώστε να μπορεί να εφαρμοστεί για τα δεδομένα του προβλήματος μας, δηλαδή το θέμα των χρονικών παραθύρων. Έπειτα, η ανάθεση των φορτηγών πραγματοποιήθηκε με τέτοιο τρόπο ώστε ο αριθμός τους να ελαττωθεί αρκετά. Για καλύτερη ποιότητα στα αποτελέσματα, το πρόβλημα μελετήθηκε πάλι σαν ομάδες προβλημάτων ενώ οι ομάδες δημιουργήθηκαν και πάλι μέσω της ομαδοποίησης K-means. Με την διαδικασία που αναλύθηκε στο Κεφάλαιο 4 πλέον γίνεται ανάθεση νέου οχήματος μόνο εάν η συνολική χωρητικότητα του προηγούμενου οχήματος καλύπτει ένα ποσοστό ή εάν ο χρόνος εξυπηρέτησης είναι μεγαλύτερος σε βαθμό που το επιβάλλει. Τα αποτελέσματα όπως αναμενόταν άλλωστε είναι χειρότερα σε θέμα χρόνου καθώς χρειάζονται 132 εργατοώρες για την κάλυψη όλων των ζητήσεων όμως ο αριθμός των δρομολογηθέντων οχημάτων είναι πλέον 22, σχεδόν τα μισά από ότι στις προηγούμενες περιπτώσεις. Αξίζει επίσης να σημειωθεί πως οι χρόνοι επίλυσης στον υπολογιστή στην περίπτωση αυτή είναι τρομερά πιο μικροί για όλες τις ομάδες. Το αποτέλεσμα με την επιβολή «άπληστου» αλγορίθμου μπορεί λοιπόν να μην προσφέρει την ίδια αξία στην λύση του, κερδίζει όμως αξία όταν απαιτείται εύρεση εφικτών λύσεων σε σύντομα χρονικά διαστήματα.

Τέλος δοκιμάστηκε να πραγματοποιηθεί μία διαδικασία ομαδοποίησης των δεδομένων με διαφορετικά κριτήρια από αυτά της τοποθεσίας τους. Στόχος είναι η ομαδοποίηση των πελατών με βάση την ομοιότητα των χρονικών παραθύρων εξυπηρέτησής τους (Time-Windows). Εφαρμόστηκε πρώτα μία ευρετική διαδικασία που ακολουθεί τη τακτική του διαχωρισμού σε γεωγραφικές περιοχές για έναν πρώτο διαχωρισμό των δεδομένων με βάση την θέση τους ως προς την βάση των επιχειρήσεων για λογική βελτίωση των αποτελεσμάτων που θα βρεθούν. Έπειτα οι πελάτες των γεωγραφικών ομάδων κατηγοριοποιήθηκαν ανάλογα με το χρόνο όπου είναι διαθέσιμοι. Τα αποτελέσματα στην περίπτωση αυτή χαρακτηρίζονται ικανοποιητικά παρόλο που απέχουν από την καλύτερη λύση ως προς τις εργατοώρες. Αυτό λόγω του ότι η απαίτηση 5 επιπλέων εργατοωρών συνδυάζεται με μείωση των φορτηγών που απαιτούνται κατά 8 γεγονός που σε περιπτώσεις είναι κρίσιμο.

Με βάση τα παραπάνω καταλήγουμε στο συμπέρασμα ότι έχοντας μοναδικό κριτήριο αυτό του χρόνου, η επιλογή μεθόδων επιχειρησιακής έρευνας αποφέρουν έμπρακτα τα καλύτερα αποτελέσματα. Εμφανίζεται όμως επιτακτική η ανάγκη εισαγωγής παραμέτρων που μπορεί να μην γίνουν εμφανή μέχρι την στιγμή της επίλυσης. Στο σημείο αυτό γίνεται αναγκαίο να ληφθούν αποφάσεις για τον βαθμό βελτιστότητας που είναι επιθυμητή. Στο πρόβλημα που μελετήθηκε το μοναδικό κριτήριο λήψης αποφάσεων είναι ο

χρόνος εξυπηρέτησης. Σε κάποια άλλη μελέτη μπορεί η δρομολόγηση μεγάλου αριθμού οχημάτων να μην είναι είτε εφικτή είτε συμφέρουσα. Είναι λοιπόν αναγκαίο όταν αντιμετωπίζουμε ένα τέτοιο πολύπλοκο πρόβλημα να κρατάμε όλες τις παραμέτρους που εμφανίζονται στην διαδικασία επίλυσης. Να λαμβάνεται δηλαδή υπόψη όχι μόνο το αποτέλεσμα σαν ένας αριθμός αλλά και ο χρόνος που απαιτεί ο σχεδιασμός ,η επίλυση του και η «ποιότητα» του αποτελέσματος που λαμβάνουμε.

5.2. Σκέψεις για μελλοντικές βελτιώσεις

Με την σημασία του κλάδου της επιχειρησιακής έρευνας να γίνεται καθημερινά όλο και πιο αναγκαία και σε συνδυασμό με την ραγδαία τεχνολογική πρόοδο στον τομέα της πληροφορικής, τα δεδομένα στα προβλήματα βελτιστοποίησης αλλάζουν συνεχώς. Προβλήματα που απαιτούσαν ώρες για μία και μόνο λύση πλέον επιλύονται σε λίγα λεπτά καθώς τα εργαλεία που έχει ένας μηχανικός παραγωγής στην διάθεση του εκσυγχρονίζονται καθημερινά. Πλέον σε πολλούς 'solvers' και βιβλιοθήκες βελτιστοποίησης όπως είναι οι κορυφαίοι Cplex, Gurobi και τα Google OR Tools προβλήματα όπως το VRP επιλύονται γρήγορα και αλγοριθμικά σε τρομακτικές ταχύτητες, με σωστή δόμηση των δεδομένων. Τα προβλήματα που εμφανίζονταν στο παρελθόν στην επίλυση αυτών των προβλημάτων εξαφανίζονται με τον χρόνο και με την εισαγωγή επιστημών όπως το 'machine learning' και 'neural networks' που κάνουν την εμφάνιση τους συνεχώς αναμένονται σημαντικές εξελίξεις στο μέλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Berhan, E., B. Beshah and D. Kitaw, 2014: Stochastic Vehicle Routing Problem: A Literature Survey Journal of Information & Knowledge Management, Vol. 13, No. 3 (2014) 1450022 (12 pages). World Scientific Publishing Co. DOI: 10.1142/S0219649214500221.

[2] Bodin, L.D., B. L. Golden, A.A. Assad, and M.O. Ball, 1983: Computers and operations Research Special Issue on the Routing and Scheduling of Vehicles and Crews, Vol. 10, no. 2, pp. 63-211.

[3] Bräysy, O. and Gendreau, M., 2005. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. Transportation Science. 39. 104-118.

[4] Capacitated Vehicle Routing Using Nearest Neighbor Algorithm in Supply Chain Sanjay Kulkarni, Dr. Nagendra Sohani (Dept of Mechanical Engineering) IET-DAVV Indore, India Neha Sehta Department Of Information Technology SDBCT Indore, India

[5] Cordeau, Jean-François & Laporte, Gilbert & Savelsbergh, Martin & Vigo, Daniele. (November 23, 2005). Vehicle Routing, book: Transportation, Publisher: Elsevier, Editors: Barnhart, Cinthya and Laporte, Gilbert, pp.195-224

[6] Derigs, U. and A. Metz. 1992: A matching-based approach for solving a delivery /pick-up VRP with time constraints. OR-Spektrum, 14:91-106, 1992.

[7] Dumas, Y., J. Desrosiers, and F. Soumis, 1991: The pickup and delivery problem with time windows. European Journal of Operational Research, 54:7-22, 1991.

[8] Eshetie Berhan*, Birhanu Beshah† and Daniel Kitaw‡(2014) Stochastic Vehicle Routing Problem: A Literature Survey Journal of Information & Knowledge Management, Vol. 13, No. 3 (2014) 1450022 (12 pages) #.c World Scientific Publishing Co. DOI: 10.1142/S0219649214500221

[9] Dantzig, G. B.., and. Ramser J.H, 1959: The Truck Dispatching Problem Author(s): Management Science, Vol. 6, No. 1 pp. 80-91.

[10] Johan Oscar Ong, Suprayogi. Vehicle Routing Problem with Backhaul, Multiple Trips and Time Window. Jurnal Teknik Industri, Vol. 13, No. 1, Juni 2011, 1-10

[11] Laporte, G., 2009: Fifty Years of Vehicle Routing. Transportation Science, 43(4), 408416.Mourgaya, M. and F. Vanderbeck, 2006: Probleme de tournees de vehicules multiperiodiques: Classification et heuristique pour la planification tactique. *RAIRO Opns Res* **40**: 169–194.

[12] Ong, J.O. and Suprayogi, 2011: Vehicle Routing Problem with Backhaul, Multiple Trips and Time Window. Jurnal Teknik Industri, Vol. 13, No. 1, Juni 2011, 1-10.

[13] Surekha, P and Sumanthi, Solution to Multi-Depot Vehicle Routing Problem Using Genetic Algorithms, World Applied Programming, Vol (1), No (3), p. 118-131, August 2011

[14] Coene, S., A. Arnout and F.C.R. Spieksma. Katholieke Universiteit Leuven, Leuven, Belgium. Published online 6 January 2010. Journal of the Operational Research Society

[15] Tai-Hsi Wu, Chinyao Low, Jiunn-Wei Bai,Heuristic solutions to multi-depot location-routing problems,Computers & Operations Research,Volume 29, Issue 10,2002,Pages 1393-1415, September 2002

[16] Toth, P., Vigo, D., 2014. Vehicle routing: Problems, Methods and Applications, Second Edition. Philadelphia: SIAM.

ΤΙΤΛΟΙ ΣΧΗΜΑΤΩΝ

ΤΙΤΛΟΙ ΠΙΝΑΚΩΝ

ΛΙΣΤΑ ΑΚΡΟΝΥΜΩΝ

Backhaul Customers: πελάτες οπίσθιας εξυπηρέτησης.

CVRP: Capacitated Vehicle Routing Problem.

K-Means Clustering: Ομαδοποίηση δεδομένων βάση του αλγορίθμου Κ-μέσων.

LIFO: Last In First Out.

Linehaul Customers: πελάτες εμπρόσθιας εξυπηρέτησης

MD-VRP: Vehicle Routing Problem with Multiple Depot – MDVRP.

NNA: Nearest Neighbor Algorithm

PVRP: Periodic Vehicle Routing Problem.

SVRP: Stochastic Vehicle Routing Problem.

TSP: Travel Salesman Problem.

VRP: Vehicle Routing Problem.

VRPB: Vehicle Routing Problem With Backhauls.

VR-PPD: Pick-up and Delivery.

VRP-TW: Vehicle Routing Problem With Time-Windows.

CODES OF THE MODELS AND SOLVERS USED

```
//KmeansClustering

//K means clustering using C++  calling data from kmeans.txt


#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>


using namespace std;


ifstream infile;

ofstream outfile;

ofstream out;


double x[171], y[171], assignedto[171];

double centroidx[10], centroidy[10];

double oldcentroidx[10], oldcentroidy[10];

int const kmax = 10;

int const imax = 171;

int const pel = 172;

int k = kmax;                  // k = number of centroids

int centroidcount[10];

int dataCount = 171;

int const M = 10000;
```

```
int K[kmax];
int j, i, b, c, h;
int id[pel];



double calculateDistance(double x, double y, double x1, double y1);
void assignCentroid(double x, double y, int point);
void calculateNewCentroid();


int main()
{
	string input = "input.txt";
	string output = "output.txt";



	outfile.open(output.c_str());


	// Read in input.
	infile.open(input.c_str());
	if (!infile)
	{
		cout << "Unable to open input." << endl;
	}
	/*while (!infile.eof())
	{
```

```
        infile >> x[dataCount] >> y[dataCount];
        dataCount++;
}*/
for (i = 0; i < imax; i++) {
        infile >> x[i] >> y[i];
}

infile.close();

// Chose initial centroids.  Hard coding for testing but can/should be random per requirements.
for (int i = 0; i<k; i++)
{
        centroidx[i] = x[i];
        centroidy[i] = y[i];

        // For debugging
        cout << "Centroid " << i << ": [" << centroidx[i] << ", " << centroidy[i] << "]" << endl;
        outfile << "Centroid " << i << ": [" << centroidx[i] << ", " << centroidy[i] << "]" << endl;
}

outfile.close();

// Assign points to centroids based on closest mean
for (int i = 0; i<dataCount; i++)
{
```

```
        assignCentroid(x[i], y[i], i);
}

//for( int i=0; i<k; i++)
//{
//      cout << centroidcount[i] << endl;
//}
// Open output

for (int i = 0; i<10; i++)
{
        calculateNewCentroid();
        for (int i = 0; i<dataCount; i++)
        {
                assignCentroid(x[i], y[i], i);
        }
}

for (int i = 0; i < k; i++) {
        for (int j = 0; j < dataCount; j++) {
                if (assignedto[j] == i) {
                        cout << j + 1 << "          :          " << i << endl;
                }
        }
}
```

```
for (int i = 0; i<imax; i++)
{
	//cout << i << ":" << assignedto[i] << endl;
	for (j = 0; j < kmax; j++) {
		if (assignedto[i] == j) K[j] = K[j] + 1;
	}
}

out.open("ClusterNum.txt");
if (out.fail())
{
	cout << "out file could not be opened" << endl;
	system("pause");
	exit(1);
}

for (i = 0; i < kmax; i++) {
	cout << K[i] << endl;
	out << K[i] << endl;
}
cout << endl;

out.close();
```

```
float pelates[5][imax + 1];


infile.open("Pelates.txt");
if (infile.fail())
{
	cout << "input file could not be opened" << endl;
	system("pause");
	exit(1);
}


for (int i = 0; i < imax; i++) {
	for (int j = 0; j < 5; j++) {
		pelates[j][i] = 0;
	}
}


infile >> pelates[0][0] >> pelates[1][0] >> pelates[2][0] >> pelates[3][0] >> pelates[4][0];
pelates[5][0] = 0;
for (int i = 1; i < imax + 1; i++) {
	infile >> pelates[0][i] >> pelates[1][i] >> pelates[2][i] >> pelates[3][i] >> pelates[4][i];
	pelates[5][i] = assignedto[i];
}


infile.close();
```

```
out.open("Pelatesnew.txt");


out << "0          0          0          0          1440" << endl;
c = 1;
id[0] = 0;
for (j = 0; j < kmax; j++) {
        for (i = 1; i < imax + 1; i++) {
                if (pelates[5][i] == j) {
                        for (b = 0; b < 5; b++) {
                                out << pelates[b][i] << " ";
                        }
                        out << endl;
                        id[c] = i;
                        c = c + 1;
                }
        }
}


out.close();


double Dij[pel][pel];
double DDij[pel][pel];


infile.open("Dij.txt");
```

```
if (infile.fail())
{
	cout << "input file could not be opened" << endl;
	system("pause");
	exit(1);
}

for (int i = 0; i < pel; i++) {
	for (int j = 0; j < pel; j++) {
		Dij[i][j] = 0;
		DDij[i][j] = 0;
	}
}

for (i = 0; i < pel; i++) {
	for (j = 0; j < pel; j++) {
		infile >> DDij[i][j];
		//cout << DDij[i][j] << "   ";
	}
	//cout << endl;
}

out.open("Dijnew.txt");

for (i = 0; i < pel; i++) {
```

```
		for (j = 0; j < pel; j++) {
			Dij[i][j] = DDij[id[i]][id[j]];
			out << Dij[i][j] << "	";
		}
		out << endl;
}

out.close();
infile.close();

double Tij[pel][pel];
double TTij[pel][pel];

infile.open("Tij.txt");
if (infile.fail())
{
		cout << "input file could not be opened (Tij)" << endl;
		system("pause");
		exit(1);
}

for (int i = 0; i < pel; i++) {
		for (int j = 0; j < pel; j++) {
			Tij[i][j] = 0;
			TTij[i][j] = 0;
```

```
            }
}


for (i = 0; i < pel; i++) {
            for (j = 0; j < pel; j++) {
                        infile >> TTij[i][j];
                        //cout << DDij[i][j] << "   ";
            }
            //cout << endl;
}


out.open("Tijnew.txt");


for (i = 0; i < pel; i++) {
            for (j = 0; j < pel; j++) {
                        Tij[i][j] = TTij[id[i]][id[j]];
                        out << Tij[i][j] << "          ";
            }
            out << endl;
}


out.close();
```

```
	cin >> x[0];
}

double calculateDistance(double x, double y, double x1, double y1)
{
	double part1 = (x - x1) * (x - x1);
	double part2 = (y - y1) * (y - y1);
	double answer = sqrt(part1 + part2);

	return answer;
}

void assignCentroid(double x, double y, int point)
{
	double smallest = 999;
	int chosenCentroid = 999;

	for (int i = 0; i<k; i++)
	{
		double distanceToCentroid = calculateDistance(x, y, centroidx[i], centroidy[i]);

		if (distanceToCentroid < smallest)
		{
			smallest = distanceToCentroid;
			chosenCentroid = i;
```

```
                }
        }
        assignedto[point] = chosenCentroid;
        centroidcount[chosenCentroid]++;
}

//void assignCentroid1(double x, double y, int point)
//{
//      double smallest = 999;
//      int chosenCentroid = 999;
//      double cluster[imax][kmax];
//
//      for (int i = 0; i<k; i++)
//      {
//              double distanceToCentroid = calculateDistance(x, y, centroidx[i], centroidy[i]);
//
//              if (distanceToCentroid < smallest)
//              {
//                      smallest = distanceToCentroid;
//                      chosenCentroid = i;
//              }
//      }
//      assignedto[point] = chosenCentroid;
//      centroidcount[chosenCentroid]++;
//}
```

```
void calculateNewCentroid()
{
	for (int i = 0; i<k; i++)
	{
		cout << endl;
		outfile << endl;

		oldcentroidx[i] = centroidx[i];
		oldcentroidy[i] = centroidy[i];

		double xsum = 0;
		double ysum = 0;
		double count = 0;

		for (int j = 0; j < dataCount; j++)
		{
			if (assignedto[j] == i)
			{
				xsum += x[j];
				ysum += y[j];
				count++;
			}
		}
		centroidx[i] = xsum / count;
```

```
                centroidy[i] = ysum / count;

                double movement = calculateDistance(oldcentroidx[i], oldcentroidy[i], centroidx[i],
centroidy[i]);

                cout << "Centroid " << i << ": [" << centroidx[i] << ", " << centroidy[i] << "]" << endl;
                outfile << "Centroid " << i << ": [" << centroidx[i] << ", " << centroidy[i] << "]" << endl;
                cout << "Centroid moved " << movement << endl;
        }
}

//C++ Model Calling Cplex Ilog libraires written in C++
//Odigies dinodai sto readmeTXT
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

//Declaration

int i, j, k, h;
const int imax = 11;
const int kmax = 5;
const int jmax = imax;
const int hmax = imax;
const int p = 0;
const int M = 100000;
const int Day = 1440;
```

```
float Q1[jmax];
float Q2[jmax];
float Tdj[jmax];
float aj[jmax];
float bj[jmax];
float CAP1[kmax];
float CAP2[kmax];
float Tij[imax][jmax];



int
main(int argc, char **argv)
{
	ifstream inFile;

	inFile.open("customers.txt");
	if (inFile.fail())
	{
		cout << "input file could not be opened1" << endl;
		system("pause");
		exit(1);
	}

	//Initialization
```

```
	for (j = 0; j < jmax; j++) {
		Q1[j] = 0;
		Q2[j] = 0;
		Tdj[j] = 0;
		aj[j] = 0;
		bj[j] = 0;
	}


	for (j = 1; j < jmax; j++) {
		inFile >> Q1[j] >> Q2[j] >> Tdj[j] >> aj[j] >> bj[j];
		cout << Q1[j] << "     " << Q2[j] << "     " << Tdj[j] << "     " << aj[j] << "     " << bj[j] << "     "
<< endl;
	}

	inFile.close();

	// CAP
	inFile.open("trucks.txt");
	if (inFile.fail())
	{
		cout << "input file could not be opened2" << endl;
		system("pause");
		exit(1);
	}
```

```
//Initialization


for (k = 0; k < kmax; k++) {
        CAP1[k] = 0;
        CAP2[k] = 0;
}


for (k = 0; k < kmax; k++) {
        inFile >> CAP1[k] >> CAP2[k];
        cout << CAP1[k] << "     " << CAP2[k] << endl;
}


inFile.close();



// Tij
inFile.open("Tij.txt");
if (inFile.fail())
{
        cout << "input file could not be opened3" << endl;
        system("pause");
        exit(1);
}
```

```
//Initialization


for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                Tij[i][j] = 0;
        }
}


for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                inFile >> Tij[i][j];
                Tij[i][j] = Tij[i][j] / 60;
                cout << Tij[i][j] << "     ";
        }

        cout << endl;
}


inFile.close();



//C-PLEX Start


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 Variable Xijk
        //Binary. Takes value 1 if route i->j is performed by truck k

        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 Move[70];
                        sprintf(Move, "Xijk(i%d,j%d,k%d)", i, j, k);
                        IloNumVar X(env, 0, 1, ILOINT, Move);
                        Xk.add(X);
                }
                Xjk.add(Xk);
        }
        Xijk.add(Xjk);
}

// Decision Variable TWijk
//Time Wait for j customer's Time Window to open, when coming from i with truck k

IloNumVarMatrix3x3 TWijk(env, 0);
for (i = 0; i<imax; i++) {
        IloNumVarMatrix2x2 TWjk(env, 0);
        for (j = 0; j<jmax; j++) {
                IloNumVarArray TWk(env, 0);
                for (k = 0; k<kmax; k++) {
                        char TimeWait[70];
                        sprintf(TimeWait, "TWijk(i%d,j%d,k%d)", i, j, k);
                        IloNumVar TW(env, 0, Day, ILOFLOAT, TimeWait);
```

```
                    TWk.add(TW);
                }
                TWjk.add(TWk);
            }
            TWijk.add(TWjk);
}

//Decision Variable Wjk
//Time that truck k starts to serve customer j

IloNumVarMatrix2x2 Wjk(env, 0);
for (j = 0; j<jmax; j++) {
            IloNumVarArray Wk(env, 0);
            for (k = 0; k<kmax; k++) {
                        char TimeServe[70];
                        sprintf(TimeServe, "Wjk(j%d,k%d)", j, k);
                        IloNumVar W(env, 0, Day, ILOFLOAT, TimeServe);
                        Wk.add(W);
            }
            Wjk.add(Wk);
}



//Subtour Elinimination Variable Y
//Variable that helps apply the Miller, Tucker and Zemlin Subrout Elimination Algorithm
```

```
IloNumVarArray Yj(env, 0);
for (j = 0; j < jmax; j++) {
        char subtour[70];
        sprintf(subtour, "Yj(j%d)", j);
        IloNumVar Y(env, 0, IloInfinity, ILOFLOAT, subtour);
        Yj.add(Y);
}



//CONSTRAINTS



//Constraint 1
//Everyone visited exactly once

IloRangeArray Visitj(env, 0);
for (j = 1; j < jmax; j++){
        IloExpr expr(env, 0);
        for (i = 0; i < imax; i++){

                if (j != i) {

                        for (k = 0; k < kmax; k++) {
```

```
                                        expr += Xijk[i][j][k];
                                }
                        }
                }
                char AllServed[60];
                sprintf(AllServed, "Visitj(j%d)", j);
                float LB = 1, UB = 1;
                IloRange Visit(env, LB, expr, UB, AllServed);
                model.add(Visit);
                Visitj.add(Visit);
                expr.end();
}

//Constraint 2
//Every truck leaves depot max once

IloRangeArray leave_oncek(env, 0);

for (k = 0; k < kmax; k++) {
                IloExpr expr(env, 0);

                for (j = 1; j < jmax; j++) {

                        expr += Xijk[p][j][k]; //p=0
```

```
        }
        char leave[60];
        sprintf(leave, "leave_oncek(k%d)", k);
        float LB = -IloInfinity , UB = 1;
        IloRange leave_once(env, LB, expr, UB, leave);
        model.add(leave_once);
        leave_oncek.add(leave_once);
        expr.end();
}



// Constraint 3
//If goes to i, it leaves i (Come & Go)

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 Leaveijk[60];
                sprintf(Leaveijk, "Leavehk(h%d,k%d)", h, k);
                float LB = 0, UB = 0;
                IloRange Leave(env, LB, expr, UB, Leaveijk);
                model.add(Leave);
                Leaveh.add(Leave);
                expr.end();
        }
        Leavehk.add(Leaveh);
}


//Constraint 4
//Time value to Wi

IloRangeMatrix3x3 TimeVijk(env, 0);

for (i = 0; i < imax; i++) {
        IloRangeMatrix2x2 TimeVjk(env, 0);

        for (j = 1; j < jmax; j++) {

                IloRangeArray TimeVk(env, 0);
```

```
                for (k = 0; k < kmax; k++) {

                        IloExpr expr(env, 0);

                        expr += Wjk[j][k] - Wjk[i][k] - Tdj[i] - Tij[i][j] + M * (1 - Xijk[i][j][k]);
                        char TimeValue[60];
                        sprintf(TimeValue, "TimeV(i%d,j%d,k%d)", i, j, k);
                        float UB = IloInfinity, LB = 0;
                        IloRange TimeV(env, LB, expr, UB, TimeValue);
                        expr.end();
                        model.add(TimeV);
                        TimeVk.add(TimeV);
                }

                TimeVjk.add(TimeVk);
        }
        TimeVijk.add(TimeVjk);
}




////Constraint 5
////Wait to serve
```

```
IloRangeMatrix3x3 TimeSijk(env, 0);

for (i = 0; i < imax; i++) {
        IloRangeMatrix2x2 TimeSjk(env, 0);

        for (j = 1; j < jmax; j++) {

                IloRangeArray TimeSk(env, 0);

                for (k = 0; k < kmax; k++) {

                        IloExpr expr(env, 0);

                        expr += Wjk[j][k] - Wjk[i][k] - Tdj[i] - Tij[i][j] - TWijk[i][j][k] -
M*(1-Xijk[i][j][k]);
                        char TimeServe[60];
                        sprintf(TimeServe, "TimeS(i%d,j%d,k%d)", i, j, k);
                        float UB = 0, LB = -IloInfinity;
                        IloRange TimeS(env, LB, expr, UB, TimeServe);
                        expr.end();
                        model.add(TimeS);
                        TimeSk.add(TimeS);
                }

                TimeSjk.add(TimeSk);
```

```
        }
        TimeSijk.add(TimeSjk);
}


//Constraint 6
//Time Windows

//Lower bound

IloRangeMatrix2x2 TimeWinLjk(env, 0);

for (j = 1; j < jmax; j++) {
        IloRangeArray TimeWinLk(env);
        for (k = 0; k < kmax; k++) {

                IloExpr expr(env, 0);

                for (i = 0; i < imax; i++) {
                        expr += aj[j] * Xijk[i][j][k];
                }
                expr -= Wjk[j][k];
                char TimeWindowsL[60];
                sprintf(TimeWindowsL, "TimeWinL(j%d,k%d)", j, k);
                float LB = -IloInfinity, UB = 0;
```

```
                IloRange TimeWinL(env, LB, expr, UB, TimeWindowsL);
                expr.end();
                model.add(TimeWinL);
                TimeWinLk.add(TimeWinL);

        }
        TimeWinLjk.add(TimeWinLk);

}

//Upper bound

IloRangeMatrix2x2 TimeWinUjk(env, 0);

for (j = 1; j < jmax; j++) {
        IloRangeArray TimeWinUk(env);
        for (k = 0; k < kmax; k++) {

                IloExpr expr(env, 0);

                for (i = 0; i < imax; i++) {
                        expr += bj[j] * Xijk[i][j][k];
                }
                expr -= Wjk[j][k];
                char TimeWindowsU[60];
```

```
                sprintf(TimeWindowsU, "TimeWjnU(j%d,k%d)", j, k);
                float LB = 0, UB = IloInfinity;
                IloRange TimeWinU(env, LB, expr, UB, TimeWindowsU);
                expr.end();
                model.add(TimeWinU);
                TimeWinUk.add(TimeWinU);

        }
        TimeWinUjk.add(TimeWinUk);

}

//Constraint 7
//Secure that Wjk=0 if k not deployed

IloRangeArray NoDeployijk(env, 0);
for (k = 0; k<kmax; k++) {
        IloExpr expr(env, 0);
        IloExpr expr1(env, 0);
        IloExpr expr2(env, 0);
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        expr1 += Xijk[i][j][k];
                }
        }
```

```
		for (j = 0; j < jmax; j++) {
			expr2 -= Wjk[j][k];
		}
		expr = expr1 * M + expr2;
		char ZeroWjk[60];
		sprintf(ZeroWjk, "NoDeployij(k%d)", k);
		float LB = 0, UB = IloInfinity;
		IloRange NoDeployij(env, LB, expr, UB, ZeroWjk);
		model.add(NoDeployij);
		NoDeployijk.add(NoDeployij);
		expr.end();
}



//Constraint 8
//Capacities

//Capacity 1

IloRangeArray Q1ijk(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 += Q1[j] * Xijk[i][j][k];
```

```
                }
        }
        char Quantity1[60];
        sprintf(Quantity1, "Q1ijk(k%d)", k);
        float LB = 0, UB = CAP1[k];
        IloRange Q1ij(env, LB, expr, UB, Quantity1);
        model.add(Q1ij);
        Q1ijk.add(Q1ij);
        expr.end();
}

//Capacity 2

IloRangeArray Q2ijk(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 += Q2[j] * Xijk[i][j][k];

                }
        }
        char Quantity2[60];
        sprintf(Quantity2, "Q2ijk(k%d)", k);
```

```
        float LB = 0, UB = CAP2[k];
        IloRange Q2ij(env, LB, expr, UB, Quantity2);
        model.add(Q2ij);
        Q2ijk.add(Q2ij);
        expr.end();
}



//Constraint 9
//8-hour Limit


IloRangeArray Shiftijk(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] + Tdj[j]) +TWijk[i][j][k];
                }
        }
        char Shift[60];
        sprintf(Shift, "Shiftij(k%d)", k);
        float LB = 0, UB = 480;
        IloRange Shiftij(env, LB, expr, UB, Shift);
        expr.end();
        model.add(Shiftij);
```

```
        Shiftijk.add(Shiftij);
}

//Constraint 10
//All trucks leave from Depot

IloRangeArray DEPOTijk(env, 0);
for (k = 0; k<kmax; k++) {
        IloExpr expr(env, 0);
        IloExpr expr1(env, 0);
        IloExpr expr2(env, 0);
        for (i = 1; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        expr1 += Xijk[i][j][k];
                }
        }
        for (h = 1; h < hmax; h++) {
                expr2 -= Xijk[p][h][k];     //p=0
        }
        expr = expr1 + expr2 * M;
        char LeavesDepot[60];
        sprintf(LeavesDepot, "DEPOTij(k%d)", k);
        float LB = -IloInfinity, UB = 0;
        IloRange DEPOTij(env, LB, expr, UB, LeavesDepot);
        model.add(DEPOTij);
```

```
        DEPOTijk.add(DEPOTij);
        expr.end();
}


//Constraint 11
//No i->i Route from truck k

IloRangeMatrix2x2 SelfRouteik(env, 0);
for (k = 0; k<kmax; k++) {
        IloRangeArray SelfRoutei(env, 0);
        for (i = 0; i < hmax; i++) {
                IloExpr expr(env, 0);

                expr += Xijk[i][i][k];

                char NoSelf[60];
                sprintf(NoSelf, "SelfRouteik(i%d,k%d)", i, k);
                float LB = 0, UB = 0;
                IloRange SelfRoute(env, LB, expr, UB, NoSelf);
                model.add(SelfRoute);
                SelfRoutei.add(SelfRoute);
                expr.end();
        }
        SelfRouteik.add(SelfRoutei);
```

```
}


//Constraint 12
//Subtour Elinimination
//Miller, Tucker and Zemlin Subrout Elimination Algorithm

IloRangeMatrix3x3 Subtourijk(env, 0);
for (i = 1; i < imax; i++) {
        IloRangeMatrix2x2 Subtourjk(env, 0);
        for (j = 1; j < jmax; j++) {
                IloRangeArray Subtourk(env, 0);
                for (k = 0; k < kmax; k++) {
                        IloExpr expr(env, 0);
                        expr += Yj[i];
                        expr -= Yj[j];
                        expr += (imax - 1) * Xijk[i][j][k];
                        char NoSubtour[60];
                        sprintf(NoSubtour, "Subtour(i%d,j%d,k%d)", i, j,
k);
                        float LB = -IloInfinity, UB = imax - 2;
                        IloRange Subtour(env, LB, expr, UB, NoSubtour);
                        expr.end();
                        model.add(Subtour);
                        Subtourk.add(Subtour);
                }
```

```
                    Subtourjk.add(Subtourk);
            }
            Subtourijk.add(Subtourjk);
}

//Objective Function
IloExpr expr1(env);

for (i = 0; i<imax; i++) {
            for (j = 0; j<jmax; j++) {
                        for (k = 0; k<kmax; k++) {
                                    expr1 += Xijk[i][j][k] * Tij[i][j] +TWijk[i][j][k]; //+ Tdj[j]
                        }
            }
}

model.add(IloMinimize(env, expr1));
expr1.end();


cplex.extract(model);
cplex.exportModel("Montelo.lp");

cplex.solve();
```

```
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;
                        }
                }
        }
}

for (k = 0; k < kmax; k++) {
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        float z = cplex.getValue(TWijk[i][j][k]);
```

```
                                        if (z != 0) cout << "TWijk" << "(" << i << "," << j << "," << k << ")"
<< "=" << z << endl;
                                }
                        }
                }

                for (k = 0; k < kmax; k++) {
                        for (j = 0; j < jmax; j++) {
                                float z = cplex.getValue(Wjk[j][k]);
                                if (z != 0) cout << "Wjk" << "("  << j << "," << k << ")" << "=" << z << endl;
                        }

                }

        }
        catch (IloException& e) {
                cerr << "concert exception caught:" << e << endl;
        }
        catch (...) {
                cerr << "Unknown exception caught" << endl;
        }

        env.end();
        system("pause");
```

```
        return 0;
}

//Matlab Model Clustering

T= readtable(clients);
c=readtable(cluster1);
TWteam=readtable(clustertime); %2 omades
TWteamval=readtable(TimeWindows); %2 omades
ClustSize = size(cluster1);
zeros(size(A,1)-size(B,12),1);
D = zeros(clustersize,'distributed');
% estmate common time window time


myIndx = findNearestCentroid(C,Xtest);
myIndex_mex = findNearestCentroid_mex(C,Xtest);
verifyMEX = isequal(idx_test,myIndx,myIndex_mex);

[N, d] = size(data);
% init U
sampleIds = randsample(1:N, K, false);
U = data(sampleIds, :);
labels_u = zeros(N, 1);
while true
```

```
stop = true;
for i = 1:N
    x = data(i, :);
    % check label
    label = 0;
    dist = 0;
    for j = 1:K
        tmp_dist = sum((x-U(j, :)).^2);
        if label == 0 || tmp_dist < dist
            label = j;
            dist = tmp_dist;
        end
    end
    if labels_u(i) ~= label
        stop = false;
    end
    labels_u(i) = label;
end
if stop == true
    break;
end
%update U
new_U = zeros(K, d);
labels_count = zeros(K, 1);
for i = 1:N
```

```
        label = labels_u(i);
        new_U(label, :) = new_U(label, :) + data(i, :);
        labels_count(label) = labels_count(label) + 1;
    end
    for i = 1:K
        new_U(i, :) = new_U(i, :)/labels_count(i);
    end
    U = new_U;
end
E_in = 0;
for i = 1:N
    label = labels_u(i);
    u = U(label, :);
    E_in = E_in + norm(x-u);
end
E_in = E_in/N;
end
```

```
// Cplex ILOG STUDIO BOTH .MODEL and .DAT files

//.Mod File ( problem Model)
```

```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: Andreas Kallos
 * Creation Date: Feb 20, 2019 at 4:18:36 PM
 *********************************************/
using CP;


int nbAsd = 19;
range asd =0..nbAsd;
range oneasd=0..nbAsd;
range monada=1..nbAsd;


int nbqwe = 9;
range qwe = 1..nbqwe;


 int imax =19;
 int jmax=19;
 int kmax =9;
 int hmax = imax;
 int M = 10000;
 int sub = imax;



// reading from .Dat File on Cplex Ilog
int TD[oneasd]=...;
```

```
int T[asd][asd]=...;

int cap1[qwe]=...;

int cap2[qwe]=...;

int b[monada]=...;

int a[monada]=...;

int Qa[monada]=...;

int Qb[monada]=...;


range i=0..imax;

range j=0..jmax;

range k=1..kmax;

range h=1..imax;


 dvar int+ Xfinal[0..imax][0..jmax][1..kmax];

 dvar int+ W[1..imax][1..kmax];

 dvar int+ TW[0..imax][0..jmax][1..kmax];

 dvar int+ Y[2..imax];


 execute {

                cp.param.FailLimit = 130000;

}


//minimize sum(i in 1..imax,j in 0..jmax, k in 1..kmax)(Xfinal[i][j][k]*(T[i][j]+TD[i]));

minimize sum(i in 0..imax,j in 0..jmax, k in 1..kmax)(Xfinal[i][j][k]*(T[i][j]+TD[i])+TW[i][j][k]);

subject to {
```

```
        ct1:
        forall(j in 1..imax)
                        sum(i in 0..imax,k in 1..kmax)
                          (Xfinal[i][j][k])==1;

   ct2:
   forall(k in 1..kmax)
        sum(j in 1..jmax)
         (Xfinal[0][j][k])<=1;

   ct3:
   forall(k in 1..kmax,h in 1..hmax  )
   sum(i in 0..imax)(Xfinal[i][h][k])== sum(j in 0..jmax)(Xfinal[h][j][k]);

//      ct11:
//      forall(i in 0..imax,k in 1..kmax)
//      Xfinal[i][i][k]==0;

        ct4:
        forall(i in 1..imax,j in 1..jmax,k in 1..kmax)
         W[j][k]-W[i][k]-TD[i]-T[i][j]      <= M*(1-Xfinal[i][j][k]);

 ct5:
        forall(i in 1..imax,j in 1..jmax,k in 1..kmax)
```

```
TW[i][j][k]>=W[j][k]- T[i][j] -W[i][k]-TD[i]-M*(1-Xfinal[i][j][k]);

ct6:

forall(j in 1..jmax,k in 1.. kmax)

  sum(i in 0..imax)(Xfinal[i][j][k])*a[j]<= W[j][k];



ct66:

forall( j in 1..jmax, k in 1..kmax)

  W[j][k]<= sum(i in 0..imax)(Xfinal[i][j][k])*b[j];



ct7:

forall(k in 1..kmax)

  sum(j in 1..jmax)(W[j][k])<=M*sum(i in 0..imax, j in 1..jmax)(Xfinal[i][j][k]);


// ston ct 8 an den katsei to i=1 ksekina to apo i=0

ct8:

forall(k in 1..kmax)

  sum(i in 1..imax, j in 0..jmax )(Xfinal[i][j][k] * Qa[i])<= cap1[k];


ct88:

forall(k in 1..kmax)

  sum(i in 1..imax, j in 0..jmax )(Xfinal[i][j][k] * Qb[i])<= cap2[k];
```

```
        ct10:
        forall (k in 1..kmax)
          sum(i in 1..imax,j in 0..jmax)(Xfinal[i][j][k])<=M*sum(h in 1..hmax)(Xfinal[0][h][k]);

        ct9:
        forall(k in 1..kmax)
//        sum(i in 1..imax,j in 0..jmax)(Xfinal[i][j][k]*(T[i][j]+TD[i]))<=480 ;
                  sum(i in 0..imax,j in 0..jmax)(Xfinal[i][j][k]*(T[i][j]+TD[i])+TW[i][j][k])<=480 ;

        ct12:  // SubtourElimination CT
        forall ( k in 1..kmax)
        forall (i in 2..imax,j in 2..imax)
          Y[i] - Y[j] + (sub-1)*Xfinal[i][j][k] + (sub-3)*Xfinal[i][j][k] <= sub-2;

        ct13://subtour elimination v2
        forall ( i in 2..imax)
          Y[i]>=1;

        ct14: // subtour elim v3
        forall (i in 2..imax)
          Y[i]<= sub-1;
}
```

```
// TO .dAT FILE for Cplex Ilog studio



/*********************************************
 * OPL 12.8.0.0 Data
 * Author: Andreas Kallos
 * Creation Date: Feb 20, 2019 at 4:18:36 PM
 *********************************************/
SheetConnection sheet("ExcelMeClusters.xlsx");
Qa from SheetRead(sheet,"Visits8!A11:A29");
Qb from SheetRead(sheet,"Visits8!B11:B29");
a from SheetRead(sheet,"Visits8!D11:D29");
b from SheetRead(sheet,"Visits8!E11:E29");
cap1 from SheetRead(sheet"Visits9!I3:I11");
cap2 from SheetRead(sheet,"Visits9!J3:J11");
T from SheetRead(sheet,"Clust2");
TD from SheetRead(sheet,"Visits8!J3:J22");




//Cplex Ilog Result From 8k Clustering.


! Time = 314.30s, Average fail depth = 654, Memory usage = 207.5 MB
```

```
! Current bound is 110 (gap is 85.92%)
!         Best Branches  Non-fixed    W       Branch decision
          781    146k     2,774    4    9,007,199,254,740,991  = Xfinal#0#0#0
          781    242k      958     2        0  = TW#9#3#7
          781    283k     3,610    3        0  = Xfinal#0#12#1
          781    160k      376     1        0  = Xfinal#9#1#3
          781    243k      457     2        0  = Xfinal#6#16#4
          781    284k     3,562    3        0  = Xfinal#11#6#5
          781    147k     2,118    4        0  = Xfinal#17#0#6
! ------------------------------------------------------------------------------
! Search terminated by limit, 71 solutions found.
! Best objective         : 781 (gap is 85.92%)
! Best bound             : 110
! Number of branches     : 834,757
! Number of fails        : 130,134
! Total memory usage     : 208.2 MB (205.7 MB CP Optimizer + 2.6 MB Concert)
! Time spent in solve    : 317.58s (315.84s engine + 1.74s extraction)
! Search speed (br. / s) : 2,642.9
```