ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Μηχανολόγων Μηχανικών

Βελτιστοποίηση του προβλήματος δρομολόγησης οχημάτων με χρονικά παράθυρα και ετερογενή στόλο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του φοιτητή

Αγγελή Δημοσθένη

Α.Μ. 1417

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

Δημήτριος Παντελής

Κωνσταντίνος Παπαδημητρίου

Βόλος 2020

`

`

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνονται οι σπουδές μου στο προπτυχιακό πρόγραμμα σπουδών του τμήματος Μηχανολόγων Μηχανικών του Πανεπιστημίου Θεσσαλίας.

Στις σπουδές μου ήταν καθοριστική η συμβολή των καθηγητών μου στα γνωστικά αντικείμενα που παρακολούθησα, στους οποίους οφείλω να εκφράσω τις ειλικρινείς μου ευχαριστίες για τη συμβολή τους στην ολοκλήρωση των σπουδών μου.

Ιδιαίτερα επιθυμώ να ευχαριστήσω τον καθηγητή μου και επιβλέποντα για την παρούσα διπλωματική εργασία, κο Σαχαρίδη Γεώργιο, για την επιστημονική και συμβουλευτική καθοδήγηση που μου προσέφερε σε όλα τα στάδια εκπόνησης της εργασίας με τις εύστοχες και πολύ εποικοδομητικές παρατηρήσεις του.

Οφείλω, επίσης, να εκφράσω τις ευχαριστίες μου προς όλους τους συναδέλφους μου, δίχως τη βοήθεια των οποίων δε θα ήταν δυνατή η διεξαγωγή της έρευνας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για τη συμπαράσταση και υπομονή τους.

`

Περίληψη

Το εμπόριο και η μεταφορά αγαθών, όντας χιλιάδες χρόνια στη ζωή των ανθρώπων, είναι καίριας σημασίας. Τους τελευταίους δύο αιώνες έχει γνωρίσει πολύ μεγάλη πρόοδο. Η ασφάλεια των μεταφορών είναι σχεδόν αυτονόητη και οι περισσότεροι ασχολούνται με την βελτίωση των οικονομικών παραγόντων. Ένα αντικείμενο πρέπει να φτάνει στον προορισμό του σε μικρό χρονικό διάστημα, με το μικρότερο κόστος. Αυτός είναι και ο λόγος που άνθισε ο κλάδος της επιχειρησιακής έρευνας και εφοδιαστικής αλυσίδας.

Σε αυτόν τον κλάδο ανήκουν τα προβλήματα μεταφοράς αντικειμένων. Σκοπός του είναι να βρεθεί μία λύση που θα επιφέρει ικανοποιητικά αποτελέσματα σε ένα αποδεκτό διάστημα χρόνου, όπως αυτά έχουν οριστεί από τον εκάστοτε επαγγελματία και πελάτη. Στην συγκεκριμένη εργασία θα ασχοληθούμε με ένα από τα βασικότερα προβλήματα του κλάδου αυτού, του προβλήματος δρομολόγησης οχημάτων.

Πιο συγκεκριμένα, το θέμα της εργασίας αυτής είναι η επίλυση του προβλήματος δρομολόγησης οχημάτων με χρονικά παράθυρα και ετερογενή στόλο. Όλα τα ανομοιογενή οχήματα του στόλου που δρομολογούνται, ξεκινάν τη διαδρομή τους από την βάση, εξυπηρετούν τις ζητήσεις των πελατών μέσα στο δοσμένο χρονικό διάστημα και επιστρέφουν. Οι ζητήσεις των πελατών και οι χωρητικότητες των οχημάτων χαρακτηρίζονται ως προς το βάρος και τον όγκο των αγαθών που χρειάζονται και μεταφέρουν αντίστοιχα.

Πρώτο βήμα της επίλυσης είναι η δημιουργία ενός αναλυτικού μοντέλου επίλυσης του προβλήματος με τεχνικές μεικτού ακεραίου προγραμματισμού. Λόγω της πολυπλοκότητας του προβλήματος δρομολόγησης οχημάτων, που είναι NP, είναι απαραίτητη η χρήση ευρετικών μεθόδων για σμίκρυνση του μεγέθους του προβλήματος. Στη συνέχεια, λοιπόν, περιγράφεται μία ευρετική μέθοδος δημιουργίας ομάδων δρομολόγησης. Τέλος, συνδυάζονται το ευρετικό και αναλυτικό κομμάτι ώστε να δημιουργηθεί ένας υβριδικός αλγόριθμος επίλυσης του συγκεκριμένου προβλήματος, που σε μικρό χρονικό διάστημα θα προσφέρει μια αποδεκτή λύση.

Παρατηρείται ότι ο νέος αλγόριθμος έχει πολύ καλή εφαρμογή στο συγκεκριμένο πρόβλημα. Οι λύσεις που δίνει βελτιώνουν τις αρχικές, όχι μόνο όσο αφορά τις απαραίτητες αποστάσεις που πρέπει να διανύσει ο στόλος, αλλά και ως προς τον απαραίτητο χρόνο επίλυσης του προβλήματος. Προσεγγίζεται, δηλαδή, σε μεγαλύτερο βαθμό η βελτιστότητα, δίχως όμως να επιτυγχάνεται.

`

Περιεχόμενα

`

`

Τίτλοι σχημάτων

`

Τίτλοι εικόνων

`

Τίτλοι πινάκων

`

Κεφάλαιο 1: Εισαγωγή

Η ανάγκη της ανθρωπότητας για δημιουργία ασφαλών διαδρομών για την μετακίνηση ή την μεταφορά αγαθών υπήρξε από την αρχαιότητα υψίστης σημασίας πρόβλημα. Για αυτόν ακριβώς τον λόγο σχηματίστηκαν οι πρώτοι δρόμοι, τα πρώτα λιμάνια και πιο πρόσφατα τα πρώτα αεροδρόμια. Ο κλάδος της επιχειρησιακής έρευνας μπορεί να πρωτοεμφανίζεται κατά τον 17ο αιώνα σε επίσημες καταγραφές όμως ο καθένας μπορεί να αντιληφθεί πως προϋπήρχε ήδη από την αρχαιότητα.

Αυτός ο κλάδος είχε πολύ μεγάλη σημασία στην καθημερινή ζωή των ανθρώπων από την αρχαιότητα. Αυτό φαίνεται από την δημιουργία οδικών δικτύων και θαλάσσιων οδών που είχαν δημιουργήσει οι Έλληνες για την μεταφορά και ανταλλαγή αγαθών με άλλους λαούς όπως οι Φοίνικες, από τον 10ο αιώνα π.Χ., ή ακόμα και από τον σχηματισμό του δρόμου του μεταξιού. Με αυτόν τον τρόπο, υπήρχε αύξηση του ανταλλακτικού εμπορίου μεταξύ διάφορων λαών, γεγονός πολύ σημαντικό τόσο για την τότε οικονομία, όσο και για την εγκαθίδρυση και ενδυνάμωση γεωπολιτικών σχέσεων.

Με το πέρασμα του χρόνου, οι συνθήκες μεταφοράς βελτιώθηκαν σε μεγάλο βαθμό. Οι δρόμοι μεταξύ των διάφορων πόλεων έγιναν πιο βατοί και η ασφάλεια αυξήθηκε ώστε να γίνει πιο εύκολη η μετακίνηση μεταξύ τους. Αυτό δείχνει και τη σημαντικότητα της όλης διαδικασίας.

Η ραγδαία εξέλιξη στον χώρου του εμπορίου βέβαια παρατηρήθηκε μετά την βιομηχανική επανάσταση. Η ολοένα αυξανόμενη παραγωγή και ζήτηση ώθησε τον χώρο της βιομηχανίας στο να αναζητήσει ακόμα πιο εύκολες μεταβάσεις από ένα μέρος σε ένα άλλο. Σε μεγάλο βαθμό αυτό οδήγησε στην δημιουργία μεγάλων μέσων μεταφοράς φορτίου, καθώς πλέον το βασικό ζήτημα ήταν η γρήγορη μεταφορά πολύ μεγαλύτερων όγκων.

Επίσης ο κλάδος των ανεφοδιασμών αλλά και της ασφαλούς μετακίνησης είχε πάντα κομβικό ρόλο σε κάθε είδους στρατιωτική επιχείρηση. Από εκεί ξεκίνησε και επίσημα ο κλάδος της επιχειρησιακής έρευνας. Αν και υπάρχουν πολλές μελέτες στον κλάδο ήδη από τον 17ο αιώνα, ο κλάδος τράβηξε την έντονη προσοχή των ερευνητών κατά την διάρκεια του πρώτου παγκοσμίου πολέμου ενώ πλέον το 1937 ο A.P. Rowe καθώς εργαζόταν για την βελτίωση των Βρετανικών ραντάρ έδωσε μια πρώτη διάσταση στην σύγχρονη προσέγγιση της επιστήμης επιχειρησιακής έρευνας.

Τμήμα του παραπάνω κλάδου αποτελεί και η επιστήμη της εφοδιαστικής αλυσίδας (Logistics) η οποία αποτελεί και την βάση πάνω στην οποία εκπονήθηκε η παρούσα μελέτη. Η επιστήμη της εφοδιαστικής αλυσίδας μπορεί να μην ονομαζόταν έτσι πριν τα μέσα του 19ου όταν και πήρε την επίσημη ονομασία της υπήρχε όμως πάντα στην ζωή μας. Η διαδικασία διανομής των προϊόντων της εκάστοτε επιχείρησης, δηλαδή η λειτουργία των logistics, αποτελεί μία από τις βασικές δραστηριότητες της εφοδιαστικής αλυσίδας. Είναι μάλιστα η διαδικασία εκείνη η οποία καταλαμβάνει, κατά μέσο όρο, το υψηλότερο ποσοστό στο συνολικό κόστος των δραστηριοτήτων της εφοδιαστικής αλυσίδας (Ballou, 1999).

Η εφαρμογή του προβλήματος της μεταφοράς αγαθών ή και υπηρεσιών από έναν στόλο οχημάτων προς ένα σύνολο πελατών είναι το πρόβλημα δρομολόγησης οχημάτων. Συνήθως σε τέτοιου είδους προβλήματα η συνθήκη γύρω από την οποία προσπαθούμε να βελτιστοποιήσουμε το πρόβλημα είναι η ελαχιστοποίηση της απόστασης και αυτός ακριβώς ήταν και ο στόχος της παρούσας μελέτης. Ασφαλώς δεν είναι η μοναδική συνθήκη καθώς βελτιστοποίηση μπορεί να γίνει γύρω από κάθε παράγοντα του προβλήματός μας. Για την επίλυση τέτοιου είδους προβλημάτων έχουν δημιουργηθεί και αναπτυχθεί πολλοί αλγόριθμοι

`

και μαθηματικά μοντέλα βελτιστοποίησης, τα οποία στην πλειοψηφία τους βασίζονται στο πρόβλημα δρομολόγησης οχημάτων.

Τα αναλυτικά μοντέλα που υπάρχουν για την επίλυση του παραπάνω προβλήματος είναι σχεδιασμένα με τέτοιον τρόπο, ώστε να δίνουν την βέλτιστη λύση. Το πρόβλημα όμως που προκύπτει με τέτοιου είδους αλγόριθμους είναι πάντα το υπολογιστικό φορτίο και αυτό ήταν και το βασικό εμπόδιο που κληθήκαμε να αντιμετωπίσουμε στην περίπτωση μας.

Η παρούσα εργασία αφορά ένα κλασικό πρόβλημα διανομής προϊόντων από έναν αποθηκευτικό χώρο στην Αθήνα και πιο συγκεκριμένα στην περιοχή του Ασπρόπυργου, σε έναν αριθμό πελατών στην ευρύτερη περιοχή της Αττικής. Για την παράδοση των παραγγελιών του κάθε πελάτη, η επιχείρηση έχει στην διάθεση της έναν στόλο φορτηγών που μπορεί να αξιοποιήσει.

Όπως όμως προαναφέραμε, το βασικό εμπόδιο το οποίο κληθήκαμε να αντιμετωπίσουμε ήταν εκείνο του υπολογιστικού φορτίου. Σε περιπτώσεις σαν κι αυτές η λύση που χρησιμοποιείται κατά κόρον είναι η ομαδοποίηση των στοιχείων και εν συνεχεία η επίλυση μικρότερων προβλημάτων. Ο αλγόριθμος που χρησιμοποιήσαμε, προκειμένου να γίνει η ομαδοποίηση, είναι ο Nearest Point, ένας αλγόριθμος σειριακής τοποθέτησης κόμβων σε διαδρομές υπό περιορισμούς.

`

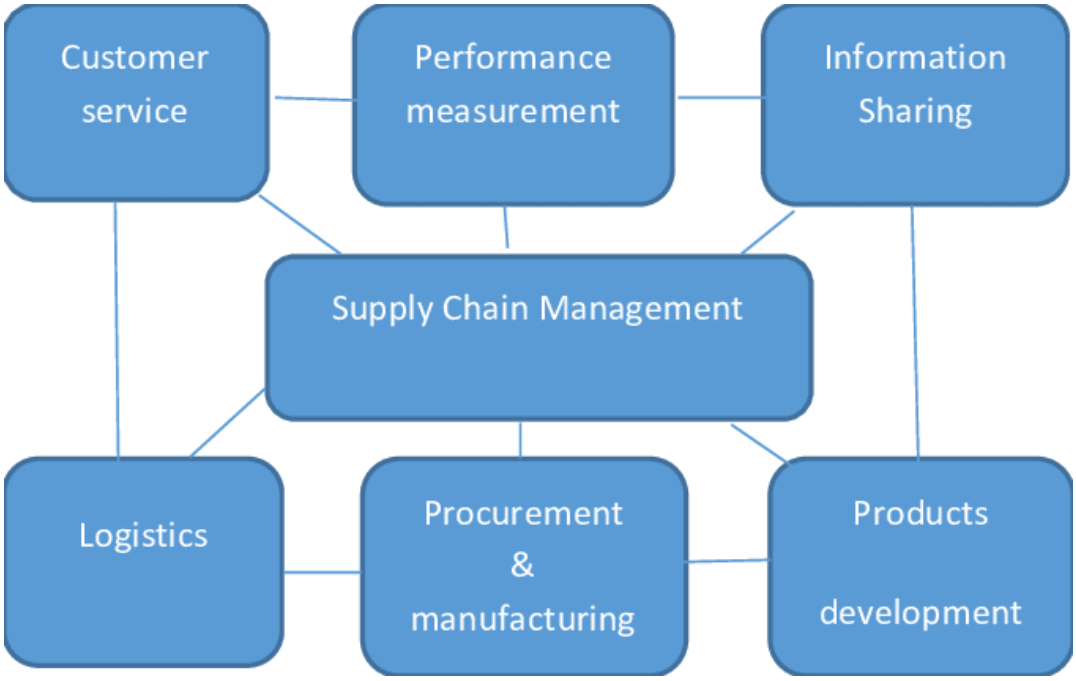
Κεφάλαιο 2: Γενικές πληροφορίες

2.1 Εφοδιαστική αλυσίδα

Η επιχειρησιακή έρευνα (Operational Research - OR) είναι ένας διεπιστημονικός κλάδος που ασχολείται με την εφαρμογή προηγμένων αναλυτικών μεθόδων για τη λήψη καλύτερων αποφάσεων. Ειδικότερα, στους κλάδους της Τεχνολογίας και της Μηχανικής, η εστίαση κατά την ανάπτυξη αυτής της επιστήμης κινήθηκε γύρω από την παραγωγή μοντέλων λύσεων. Ένας από τους τομείς της επιχειρησιακής έρευνας είναι και εκείνος της εφοδιαστικής αλυσίδας.

Η ασφαλής μετακίνηση πρώτων υλών έχει υπάρξει από την αρχαιότητα ένα από τα πιο βασικά προβλήματα που έχει κληθεί να λύσει ο άνθρωπος. Είτε αυτό ήταν για την δημιουργία ασφαλών δρόμων για το εμπόριο είτε ήταν για την εξασφάλιση τροφής κατά την διάρκεια μιας στρατιωτικής εκστρατείας, το ζητούμενο ήταν πάντα το ίδιο. Η διαχείριση τέτοιων ζητημάτων για πολλούς αιώνες βασιζόταν σε εμπειρικές λύσεις όμως από τις αρχές του 19^{ου} αιώνα παρατηρείται πως δημιουργούνται μαθηματικά μοντέλα σε μια προσπάθεια να μελετηθεί σε βάθος ο κλάδος αυτός.

Ο όρος της "εφοδιαστικής αλυσίδας" εμφανίζεται για πρώτη φορά το 1982 στους Financial Times από τον K. Oliver και έκτοτε παρατηρείται σχεδόν σε καθημερινή βάση τριγύρω μας. Η διαχείριση μιας εφοδιαστικής αλυσίδας για κάθε σύγχρονη επιχείρηση που μετακινεί πρώτες ύλες ή προϊόντα έχει πολύ μεγάλη σημασία, καθώς είναι μια διαλειτουργική προσέγγιση που περιλαμβάνει τη διαχείριση της μεταφοράς πρώτων υλών σε έναν οργανισμό, ορισμένες πτυχές της εσωτερικής επεξεργασίας υλικών σε τελικά προϊόντα και τη μετακίνηση τελικών προϊόντων εκτός του οργανισμού και προς τον τελικό καταναλωτή.



Εικόνα 2. 1. Διάγραμμα ανάλυσης τομέων του κλάδου της επιστήμης εφοδιαστικής αλυσίδας.[1]

[1]https://www.researchgate.net/figure/Logistics-and-Supply-chain-Management-Linkages-2_fig1_320235453

`

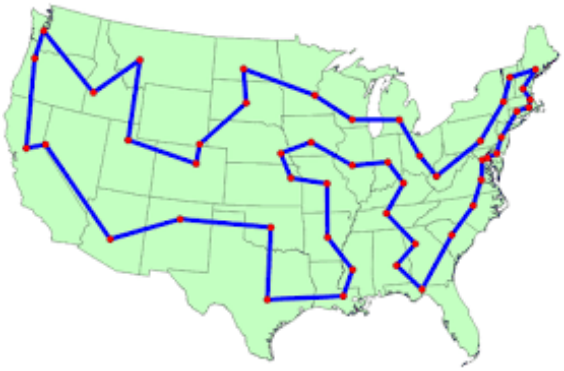
2.2 TSP

Η πρώτη αναφορά σε μελέτη προβλημάτων δρομολόγησης παρατηρείται από τον Ιρλανδό μαθηματικό W. R. Hamilton και τον Βρετανό μαθηματικό T. Kirkman στις αρχές του 1800. Το πρόβλημα το οποίο για πρώτη φορά μελετήθηκε ήταν το πρόβλημα του πλανόδιου πωλητή (Traveling salesman problem - TSP). Από τους Hamilton και Kirkman γίνεται ορισμός του προβλήματος, ενώ το 1832 εμφανίζεται εγχειρίδιο το οποίο αναφέρεται στο πρόβλημα και παρουσιάζει παραδείγματα με ταξίδια στην Γερμανία και στην Ελβετία χωρίς όμως να περιέχει κάποια μαθηματική εξήγηση.

Οι πρώτες μαθηματικές προσεγγίσεις στο πρόβλημα συναντώνται την δεκαετία του 1930 τόσο στη Βιέννη όσο και στο Harvard. Ο μαθηματικός K. Merger αποσαφηνίζει το μαθηματικό μοντέλο του TSP και παρατηρεί πως η επιλογή του κοντινότερου προορισμού μετά από κάθε ταξίδι δεν αποδίδει την βέλτιστη συνολική λύση. Τις δεκαετίες του 1950 και του 1960 το πρόβλημα αποκτά μεγάλη απήχηση στους Αμερικάνικους αλλά και στους Ευρωπαϊκούς επιστημονικούς κύκλους κυρίως λόγω της χρηματικής αμοιβής που δίνεται από την Αμερικάνικη μη κερδοσκοπική εταιρία RAND στην Santa Monica. Σημαντικές συνεισφορές δόθηκαν τότε από τους G. Dantzig, D. R. Fulkerson και S. M. Johnson οι οποίοι ήταν εργαζόμενοι τις παραπάνω εταιρίας. Οι παραπάνω εξέφρασαν το πρόβλημα σαν πρόβλημα ακέραιου προγραμματισμού και χρησιμοποίησαν την μέθοδο περιορισμού του εφικτού χώρου (Cutting planes) για την επίλυση του.

Με την χρήση αυτής της νέας μεθόδου κατάφεραν να επιλύσουν ένα πρόβλημα με 49 πόλεις αποδεικνύοντας παράλληλα ότι δεν υπήρχε καλύτερη λύση το οποίο, πάνω στην οποία έγραψαν και ένα άρθρο το οποίο θεωρείται από τα πλέον σημαντικά στην ιστορία του τομέα. Το συγκεκριμένο άρθρο μπορεί να μην δίνει έναν ξεκάθαρο αλγόριθμο για την επίλυση του προβλήματος, αλλά οι ιδέες που πρωτοεμφανίζονται εκεί θεωρούνται απαραίτητες για την μετέπειτα εξέλιξη του τομέα συνολικά αλλά και για τις αναλυτικές λύσεις που προτάθηκαν αργότερα. Οι ίδιοι χρησιμοποίησαν επίσης την μέθοδο branch and bound ενδεχομένως για πρώτη φορά στην ιστορία.

Τις επόμενες δεκαετίες, το πρόβλημα μελετήθηκε σε βάθος από ερευνητές τόσο μαθηματικών όσο και άλλων πεδίων όπως για παράδειγμα των κλάδων της φυσικής, της χημείας αλλά και της επιστήμης υπολογιστών. Μέχρι και σήμερα, πολλές διαφορετικές μέθοδοι έχουν χρησιμοποιηθεί για την επίλυση του προβλήματος, ενώ η μελέτη του υπήρξε υψίστης σημασίας καθώς αποτελεί τον προάγγελο του προβλήματος δρομολόγησης οχημάτων (Vehicle Routing Problem - VRP) το οποίο θα αναλυθεί στα επόμενα κεφάλαια και αποτελεί τον βασικό πυλώνα αυτής της μελέτης.



Εικόνα 2. 2 Χαρακτηριστικό παράδειγμα προβλήματος TSP[2]

[2] https://optimization.mccormick.northwestern.edu/index.php/Traveling_salesman_problems

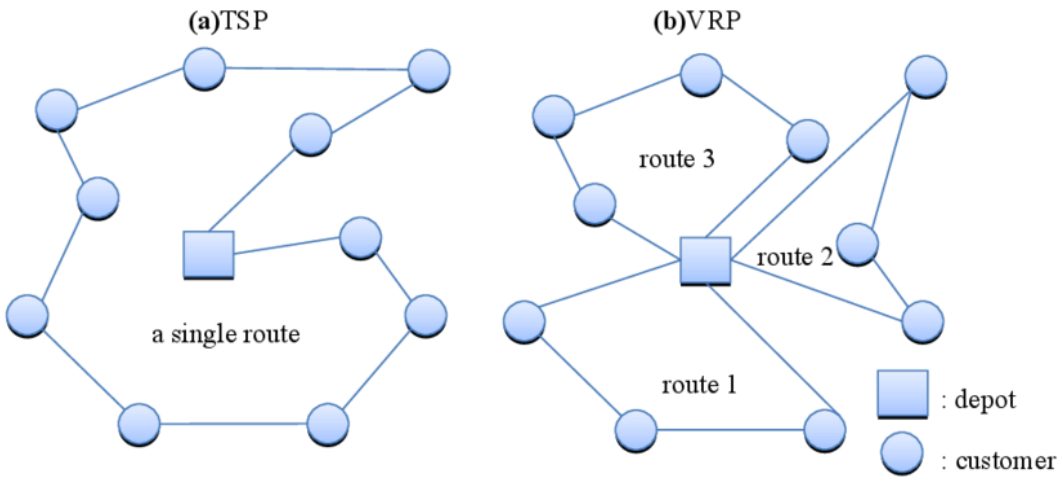
`

2.3 VRP

Το πρόβλημα δρομολόγησης οχημάτων (Vehicle routing problem – VRP) είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης που σαν στόχο έχει την εύρεση των βέλτιστων διαδρομών μεταξύ ενός συνόλου κόμβων. Σε αντίθεση με το πρόβλημα του περιπλανώμενου πωλητή που σχολιάστηκε σε προηγούμενη ενότητα, το VRP δεν περιορίζεται σε ένα μόνο δρομολόγιο και για αυτόν τον λόγο θεωρείται γενίκευσή του. Στο VRP επιτρέπεται η δημιουργία πολλαπλών διαδρομών ενώ σαν κριτήρια βελτιστοποίησης μπορούν να χρησιμοποιηθούν διάφοροι παράγοντες και όχι μόνο η ελαχιστοποίηση των αποστάσεων. Για παράδειγμα, ένα δρομολόγιο μπορεί να μελετηθεί ως προς τις ελάχιστες απαιτούμενες εργατοώρες ή την ελαχιστοποίηση του αριθμού των δρομολογίων.

Η πρώτη εμφάνιση του VRP παρατηρείται το 1959 από τους G. Dantzig και J. Ramser οι οποίοι είναι οι πρώτοι που εμφανίζουν έναν αλγόριθμο για την μεταφορά πετρελαίου. Η πιο καθιερωμένη μορφή ενός τέτοιου προβλήματος, αφορά την μετακίνηση αγαθών από μια κεντρική τοποθεσία σε έναν αριθμό τοποθεσιών. Η λήψη μιας ακριβούς λύσης για ένα τέτοιο πρόβλημα είναι συχνά δύσκολη καθώς εξαρτάται από το μέγεθός του. Πρόκειται για αλγόριθμο επίλυσης τύπου NP – hardness (non-deterministic polynomial-time hardness) καθώς δεν επιδέχεται λύσης σε πολυωνυμικό χρόνο. Κατά συνέπεια, τα εργαλεία που υπάρχουν αυτή τη στιγμή χρησιμοποιούν ευρετικούς αλγόριθμους προκειμένου να προσεγγίσουν όσο το δυνατόν καλύτερες λύσεις.

Όσον αφορά την εφαρμογή στην βιομηχανία, έχει παρατηρηθεί μείωση της τάξεως του 5% στο κόστος των μεταφορικών για ένα προϊόν. Όμως ο τομέας των μεταφορικών αποτελεί το 10% του ακαθάριστου εγχώριου προϊόντος της Ευρωπαϊκής Ένωσης και κατά συνέπεια ακόμα και μια μείωση κατά 5% έχει πολύ μεγάλο αντίκτυπο.



Σχήμα 2. 1 Διαφορά ανάμεσα σε TSP και VRP[3]

2.4 Βασικό μοντέλο VRP

Η εξέλιξη του βασικού μοντέλου του TSP που είχε δημιουργηθεί από τους Dantzig, Fulkerson και Johnson οδήγησε σε μια πρώτη αποτύπωση του μοντέλου για το VRP. Η

[3] https://www.researchgate.net/figure/Illustration-of-the-traveling-salesman-problem-TSP-and-vehicle-route-problem-VRP_fig1_277673931

`

μετάβαση αύτη όπως φαίνεται και παρακάτω αποτυπώνεται στην δημιουργία μεταβλητών με δύο δείκτες πλέον.

Η αντικειμενική συνάρτηση του προβλήματος θα είναι:

$$\min z = \sum_{i=0}^{N} \sum_{j=0,\, j\neq 1}^{N} \sum_{k=1}^{K} c_{ij} x_{ij}^{k} \qquad \forall j \in V \setminus \{0\} \tag{2.1}$$

Όπου ο όρος c_{ij} δηλώνει το κόστος μετάβασης από τον κόμβο i στον κόμβο j, ενώ ο όρος x_{ij}^{k} δηλώνει την μετακίνηση από τον κόμβο i στον κόμβο j από το όχημα k. Όσον αφορά τον όρο x_{ij}^{k}, πρόκειται για μια δυαδική μεταβλητή, η οποία όταν λαμβάνει την τιμή 1 δηλώνει ότι το δρομολόγιο από τον κόμβο i στον κόμβο j πραγματοποιείται ενώ αντίθετα όταν λαμβάνει την τιμή 0 δηλώνει ότι το παραπάνω δρομολόγιο δεν πραγματοποιείται. Ο δείκτης k χρησιμοποιείται προκειμένου να ορίζει το ακριβές όχημα που θα εκτελέσει το δρομολόγιο. Η παραπάνω αντικειμενική συνάρτηση είναι ενδεικτική καθώς η βελτιστοποίηση μπορεί να γίνει και γύρω από άλλους παράγοντες όπως για παράδειγμα την ελαχιστοποίηση της απόστασης ή οτιδήποτε άλλο ζητηθεί.

Οι βασικοί περιορισμοί του προβλήματός μας θα είναι οι παρακάτω.

- $\sum_{i=0}^{N} \sum_{k=1}^{K} x_{ij}^{k} = 1 \quad \forall j \in \{1, \ldots., \mathrm{N}\}$ (2.2)
- $\sum_{j=0}^{N} \sum_{k=1}^{K} x_{ij}^{k} = 1 \quad \forall i \in \{1, \ldots., \mathrm{N}\}$ (2.3)
- $\sum_{j=0}^{N} x_{ij}^{k} - \sum_{j=0}^{N} x_{ji}^{k} = 0 \quad \forall i \in \{1, \ldots., N\}, \quad \forall k \in \{1, \ldots., K\}$ (2.4)
- $\sum_{i=0}^{N} \sum_{j=0}^{N} x_{ij}^{k} d_i \leq Q$ (2.5)
- $\sum_{j=1}^{N} x_{0j}^{k} \leq 1 \quad \forall k \in \{1, \ldots., K\}$ (2.6)
- $\sum_{i=1}^{N} x_{i0}^{k} \leq 1 \quad \forall k \in \{1, \ldots., K\}$ (2.7)
- $x_{ij}^{k} \in \{0,1\}$ (2.8)

Οι περιορισμοί (2.2) και (2.3) δεσμεύουν τα αθροίσματα των μεταβλητών x_{ij}^{k} με τον παραπάνω τρόπο δηλώνοντας ότι ο κάθε κόμβος θα δεχθεί ακριβώς μια επίσκεψη. Αυτό φυσικά αποτελεί και το πρώτο βήμα για την δημιουργία ενός δρομολογίου κάλυψης των ζητήσεων. Στη συνέχεια ο περιορισμός (2.4) εκφράζει πως εάν ένα όχημα επισκεφτεί έναν από τους κόμβους θα πρέπει υποχρεωτικά και να αναχωρήσει από τον ίδιο κόμβο. Για τον περιορισμό (2.5) χρησιμοποιούμε τον όρο d_i ο οποίος δηλώνει την ζήτηση στον i κόμβο. Συνολικά ο περιορισμός όμως διασφαλίζει ότι η χωρητικότητα του οχήματος μας (Q) είναι αρκετή προκειμένου να καλύψει όλες τις επιμέρους ζητήσεις που καλύπτει το κάθε δρομολόγιο. Οι περιορισμοί (2.6) και (2.7) εκφράζουν ότι το κάθε φορτηγό θα εκτελέσει το πολύ ένα

`

δρομολόγιο. Ο περιορισμός (2.8) εκφράζει τον περιορισμό στις τιμές ανάμεσα στις τιμές 0-1 που μπορεί να λάβει η μεταβλητή μας.

2.5 Τα είδη του VRP.

Πέρα από την βασική του μορφή, το πρόβλημα δρομολόγησης οχημάτων μπορεί να εκφράσει πολλές ακόμα μορφές παραπλήσιων προβλημάτων. Μερικές από αυτές αναφέρονται παρακάτω.

- CVRP Capacitated Vehicle Routing Problem
- HFVRP Heterogeneous Fleet Vehicle Routing Problem
- VRPTW Vehicle Routing Problem with Time Windows
- VRPMD Vehicle Routing Problem with Multiple Depots
- VRPPD Vehicle Routing Problem with Pick-up and Delivery
- VRPB Vehicle Routing Problem with Backhauls
- SVRP Stochastic Vehicle Routing Problem
- PVRP Periodic Vehicle Routing Problem

2.5.1 Πρόβλημα δρομολόγησης με οχήματα περιορισμένης χωρητικότητας – Capacitated Vehicle Routing Problem (CVRP)

Πρόκειται για προβλήματα στα οποία ο στόλος των οχημάτων που εξυπηρετούν τις διαδρομές έχει περιορισμένες χωρητικότητες, δηλαδή υπάρχει ένα ανώτατο όριο στις ποσότητες που μπορούν να μεταφερθούν συνολικά σε κάθε δρομολόγιο. Για την επίλυσή τους χρειάζεται να προβλεφθεί η συγκεκριμένη συνθήκη κατά την μοντελοποίηση του προβλήματος, δηλαδή το άθροισμα των ζητήσεων που καλύπτει το κάθε όχημα θα πρέπει να είναι μικρότερο ή ίσο με την χωρητικότητα του οχήματος που εξυπηρετεί την διαδρομή αυτή. Ο περιορισμός αυτός όπως θα δούμε και στην αναλυτική λύση του προβλήματος στα επόμενα κεφάλαια θα είναι της μορφής (4) όπως αναφέρθηκε στην παραπάνω μοντελοποίηση.

2.5.2 Πρόβλημα δρομολόγησης ετερογενών οχημάτων - Heterogeneous fleet vehicle routing problem (HFVRP)

Η κατηγορία αυτών των προβλημάτων είναι παρόμοια με την κατηγορία των CVRP με μια όμως μεγάλη διαφορά. Η χωρητικότητα του κάθε οχήματος δεν είναι η ίδια και ως εκ τούτου η διαφοροποίηση αυτή πρέπει να αποτυπωθεί και στον αντίστοιχο περιορισμό του προβλήματος. Θα πρέπει λοιπόν να δημιουργηθεί ένας νέος περιορισμός στην θέση του περιορισμού (4) της παρακάτω μορφής:

$$\sum_{i=0}^{N}\sum_{j=0}^{N} x_{ij}^{k} d_i \le Q_k \quad \forall k \in \{1,\ldots,K\} \qquad (2.9)$$

Με αυτό τον τρόπο διασφαλίζουμε ότι το κάθε όχημα δεν θα υπερβαίνει την δική του χωρητικότητα.

2.5.3 Πρόβλημα δρομολόγησης με χρονικά παράθυρα - Vehicle Routing Problem with Multiple Depots (VRPTW)

Μια εξίσου σημαντική κατηγορία με τις προηγούμενες είναι αυτή της δρομολόγησης με χρονικά παράθυρα. Στην συγκεκριμένη κατηγορία, ο βασικός περιορισμός που πρέπει να καλύψουμε είναι εκείνος της εξυπηρέτησης του κάθε πελάτη μέσα σε διαφορετικά χρονικά περιθώρια. Σε αυτά τα προβλήματα, ανάμεσα στα δεδομένα που λαμβάνουμε από τον κάθε

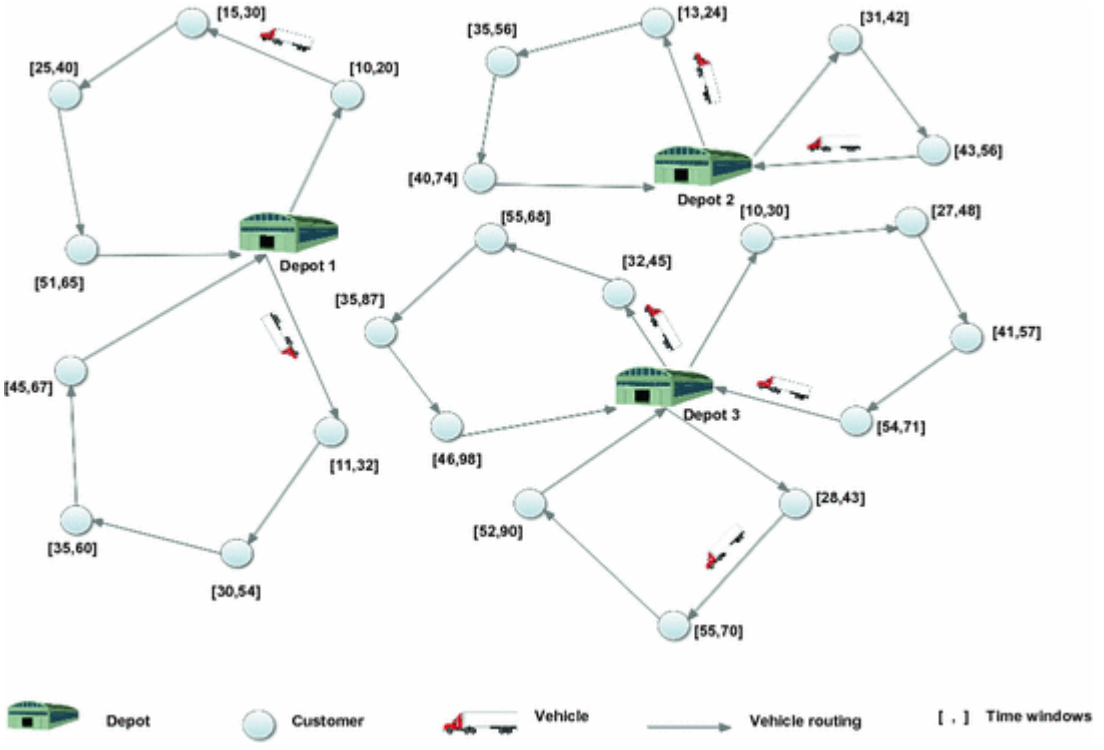
`

πελάτη (ζήτηση σε προϊόντα, γεωγραφικές συντεταγμένες κ.α.) θα πρέπει οπωσδήποτε να υπάρχει και το χρονικό παράθυρο μέσα στο οποίο ο πελάτης θέλει να εξυπηρετηθεί. Καθώς πλέον εισάγεται στο πρόβλημα και η έννοια του χρόνου είναι σαφές ότι τόσο ο αριθμός των μεταβλητών θα αυξάνεται αλλά και το πλήθος των περιορισμών θα μεγαλώνει. Κάθε αύξηση στις μεταβλητές και στους περιορισμούς σε έναν τέτοιο αλγόριθμο έχει άμεση επίπτωση στον χρόνο επίλυσης. Η μεταβλητή που εισάγουμε θα είναι της μορφής T_{ij}, και δηλώνει την διάρκεια της μετάβασης από τον i κόμβο στον j κόμβο, ενώ θα υπάρχει επίσης μια μεταβλητή της μορφής T_{di} για να εκφράζει τον χρόνο εξυπηρέτησης του κάθε πελάτη. Ο περιορισμός που θα χρειαστεί να προσθέσουμε στον μοντέλο μας θα είναι ο παρακάτω:

$$W_{j,k} - W_{i,k} - T_{di} - T_{i,j} \leq M\ (1 - X_{i,j,k}) \qquad \forall\ i,\ j,\ k \tag{2.10}$$

2.5.4 Πρόβλημα δρομολόγησης οχημάτων με πολλαπλές αποθήκες - Vehicle Routing Problem with Multiple Depots (VRPMD)

Η συγκεκριμένη κατηγορία προβλημάτων επίσης συναντάται ευρέως, κυρίως από μεγάλες εταιρίες που δραστηριοποιούνται στον χώρο της διανομής προϊόντων. Οι τοποθεσίες των αποθηκών (Depots) είναι εξ αρχής δεδομένα ενώ το ίδιο ισχύει και για τις ζητήσεις και τοποθεσίες των πελατών, όπως δηλαδή είναι και στο σύνολο των υποπεριπτώσεων που μελετάμε. Η συνθήκη που πρέπει επίσης να καλύπτεται είναι εκείνη που δεσμεύει ένα όχημα να ξεκινάει και να καταλήγει στην ίδια αποθήκη. Η βασική διαφορά στο μοντέλο επίλυσης θα έχει να κάνει με την δέσμευση των περιορισμών (3),(4),(5) και (6) για την κάθε αποθήκη ξεχωριστά, καθώς πρέπει να υποχρεώσουμε το μοντέλο σε καθαρά ανεξάρτητες διαδρομές.



Εικόνα 2. 3 Περίπτωση προβλήματος VRP με πολλαπλές αποθήκες[4]

[4] https://link.springer.com/chapter/10.1007/978-981-10-1837-4_96

`

2.5.5 Πρόβλημα δρομολόγησης οχημάτων με συλλογές κατά την παράδοση - Vehicle Routing Problem with Pick-up and Delivery (VRPPD)

Στην περίπτωση του VRP με συλλογές κατά την παράδοση ο συντελεστής της πολυπλοκότητας της επίλυσης είναι και πάλι υψηλότερος σε σχέση με ένα απλό πρόβλημα VRP. Αυτό οφείλετε στην συνθήκη του συγκεκριμένου είδους προβλημάτων, στο οποίο η συλλογή κατά την παράδοση αυξάνει τους περιορισμούς χωρητικότητας του κάθε οχήματος. Το συγκεκριμένο είδος προβλήματος μπορεί να συναντηθεί σε πολλούς συνδυασμούς προβλημάτων, όπως για παράδειγμα να συμβαίνει σε έναν ετερογενή στόλο ή να υπάρχουν πολλαπλές αποθήκες. Σύνηθες φαινόμενο είναι η κακή διαχείριση του χώρου των οχημάτων είτε κατά την παράδοση είτε κατά την επιστροφή.

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 θεμελιώδης διαφορά στους αλγόριθμους επίλυσης μεταξύ ενός απλού προβλήματος VRP και του VRP με Pick-up and Delivery είναι η προσθήκη ενός νέου δεδομένου στο σετ των ζητήσεων. Για τον κάθε πελάτη θα υπάρχουν πλέον δύο τιμές, μια για την παράδοση προϊόντων D και μια για την παραλαβή P. Θα πρέπει λοιπόν στο σύνολο των περιορισμών που ήδη υπάρχει, να προστεθεί ένας περιορισμός ούτως ώστε σε κάθε στάση του δρομολογίου να δεσμεύουμε πως η διαφορά ανάμεσα στα D_i και στα P_i (όπου i ο κάθε κόμβος στον οποίο θα γίνεται μια στάση) δεν θα δημιουργεί προβλήματα που θα οδηγούν στην υπέρβαση του ορίου χωρητικότητας του οχήματος.

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.

2.5.6 Πρόβλημα δρομολόγησης οχημάτων με “Backhauls” - Vehicle routing problem with Backhauls (VRPB)

Το πρόβλημα δρομολόγησης οχημάτων με Backhauls μοιάζει πολύ με την προηγούμενη κατηγορία που αναφέραμε, εκείνη των προβλημάτων με συλλογές κατά την παράδοση, με μια όμως βασική διαφορά. Στην συγκεκριμένη κατηγορία προβλημάτων το όχημα καλείται πρώτα να φέρει εις πέρας όλες τις παραδόσεις και μετά να αρχίσει τις παραλαβές σχηματίζοντας έτσι ένα δεύτερο κατ’ ουσίαν δρομολόγιο. 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.

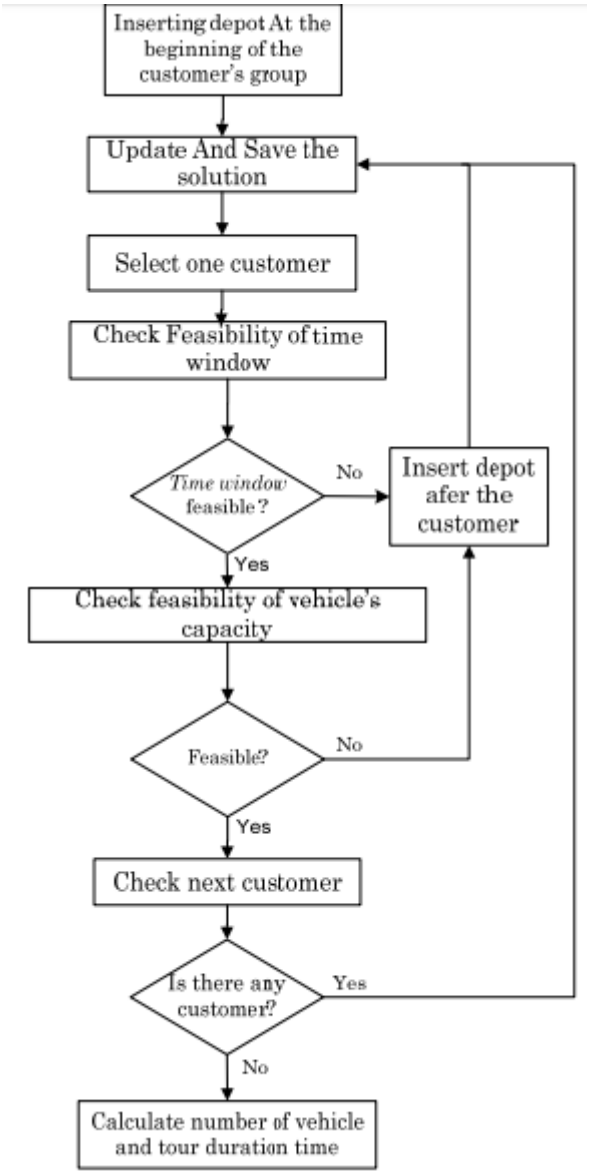
Όπως είναι αναμενόμενο τα δεδομένα και στις δύο κατηγορίες είναι ίδια καθώς βασικό ρόλο έχουν οι ποσότητες των προϊόντων που πρέπει να παραδοθούν στον πελάτη αλλά και

`

εκείνες που πρέπει να απομακρυνθούν στην δεύτερη φάση του δρομολογίου. Για αυτόν τον λόγο προχωράμε στην δημιουργία δύο συνόλων. Το πρώτο σύνολο απαρτίζουν οι πελάτες που εξυπηρετούνται για παράδοση προϊόντων (πελάτες εμπρόσθιας εξυπηρέτησης - linehaul customers) ενώ το δεύτερο απαρτίζουν οι πελάτες εκείνοι που εξυπηρετούνται κατά την παραλαβή προϊόντων (πελάτες οπίσθιας εξυπηρέτησης - backhaul customers).

Η ακολουθιακή εισχώρηση είναι η μέθοδος που χρησιμοποιείτε κατά κύριο λόγο για την επίλυση τέτοιων προβλημάτων και αυτό προκύπτει λόγω της ευκολίας στην χρήση της, την ταχύτητα με την οποία φέρνει την λύση και λόγω της εύκολης μετατροπής της για χειρισμό και μορφοποίηση δύσκολων μεταβλητών. Ο αλγόριθμος της ακολουθιακής εισχώρησης χρησιμοποιείτε για την εύρεση μιας αρχικής εφικτή λύσης. Αρχικά επιλέγει έναν linehaul πελάτη με βάση τους κανόνες προτεραιότητας, οι οποίοι είναι το μικρότερο χρονικό παράθυρο, το μεγαλύτερο χρονικό διάστημα ταξιδιού και οι πρώτες προθεσμίες

Εν συνεχεία, ακολουθείται μια δομή επίλυσης της παρακάτω λογικής.



Σχήμα 2. 2 Διάγραμμα ροής επίλυσης σε προβλήματα VRP με Backhauls

`

2.5.7 Το στοχαστικό πρόβλημα δρομολόγησης οχημάτων (SVRP)

Στο SVRP ένα ή περισσότερα στοιχεία του προβλήματος, όπως για παράδειγμα ο αριθμός των πελατών ή η ζήτησή τους, δεν θεωρούνται σταθερά, αλλά δυναμικά, δηλαδή δεν είναι γνωστά εκ των προτέρων αλλά μπορούν να αλλάξουν ανά πάσα στιγμή. Με βάση το στοχαστικό τους χαρακτηριστικό διαχωρίζονται στις εξής τρείς βασικές 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.

`

2.5.8 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.

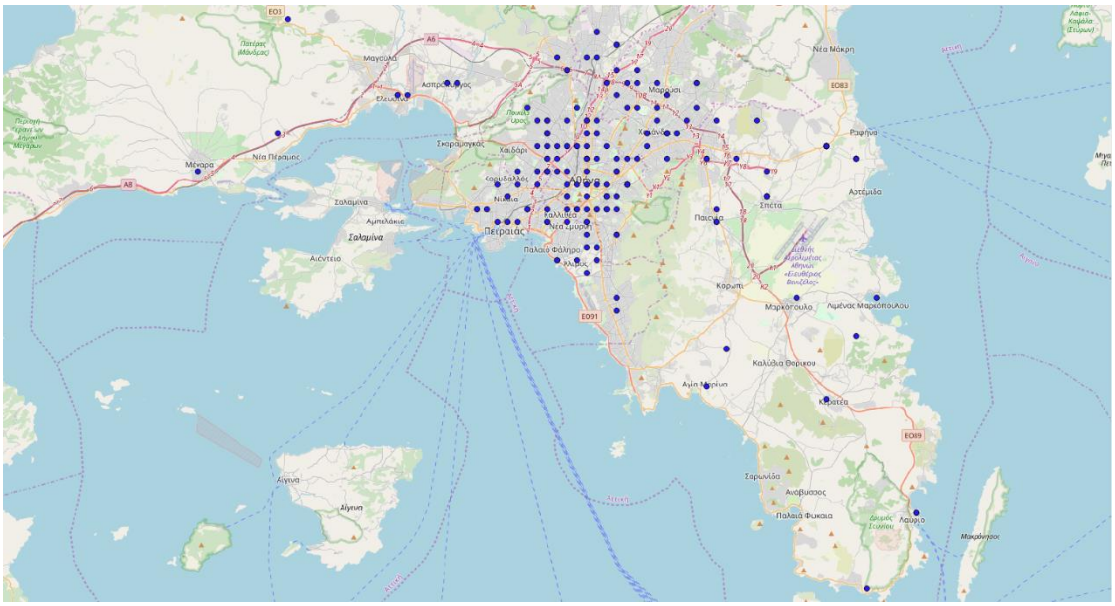
`

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

3.1. Γενικά χαρακτηριστικά του προβλήματος

Στα πλαίσια αυτής της εργασίας το πρόβλημα που μελετάται και επιλύεται είναι μια πραγματική περίπτωση του προβλήματος δρομολόγησης οχημάτων με την χρήση ετερογενούς στόλου με περιορισμένη χωρητικότητα και τον περιορισμό των χρονικών παραθύρων (CVRP-TW with Heterogenous Fleet). Πιο συγκεκριμένα, γίνεται αναζήτηση του βέλτιστου τρόπου εξυπηρέτησης της ζήτησης των πελατών από έναν συγκεκριμένο στόλο οχημάτων. Στο κεφάλαιο αυτό θα παρουσιαστούν οι παράμετροι του συγκεκριμένου προβλήματος, το αναλυτικό μοντέλο επίλυσής του και μία αρχική του λύση. Αναφορικά, οι παράμετροι του προβλήματος είναι οι εξής:

- Οι πελάτες βρίσκονται διάσπαρτοι στην περιοχή της Αττικής, όπως φαίνεται και στο σχήμα 3.1.
- Η ζήτηση του κάθε πελάτη σε αγαθά είναι γνωστή σε μονάδες όγκου και βάρους.
- Κάθε πελάτης χαρακτηρίζεται από ένα χρονικό παράθυρο εξυπηρέτησης, μέσα στο οποίο θα πρέπει να δέχεται το όχημα που τον εξυπηρετεί αλλά και να ολοκληρώνεται η διαδικασία της εξυπηρέτησης.
- Η αποστάσεις μεταξύ των πελατών και οι χρόνοι κάλυψης αυτών των αποστάσεων από κάποιο όχημα είναι γνωστοί και θεωρούνται σταθεροί.
- Ο στόλος οχημάτων είναι γνωστός και δίνονται πληροφορίες ως προς τις χωρητικότητες του κάθε οχήματος σε βάρος και όγκο προϊόντων.



Εικόνα 3. 1 Οι γεωγραφικές θέσεις των πελατών στην περιοχή της Αττικής.

`

3.2. Δεδομένα προς αξιοποίηση για την επίλυση του προβλήματος.

Μετά την αναγνώριση του προβλήματος, σειρά έχουν τα στοιχεία προς αξιοποίηση για το μοντέλο. Τα στοιχεία αυτά μπορούν να χωριστούν σε δεδομένα για τους πελάτες και δεδομένα για το στόλο οχημάτων. Η μοντελοποίηση και η επίλυση του προβλήματος στηρίζεται στην αξιοποίησή τους.

Τα στοιχεία των πελατών που δεχόμαστε ως δεδομένα είναι τα παρακάτω:

- Η ζήτηση του κάθε πελάτη ως προς το βάρος των αγαθών και ως προς τον όγκο,
- Οι γεωγραφικές αποστάσεις αναμεταξύ των πελατών αλλά και ο χρόνος μετάβασης από τον ένα στον άλλον,
- Τα χρονικά περιθώρια εξυπηρέτησης του κάθε πελάτη.
- Τα στοιχεία του στόλου που δεχόμαστε ως δεδομένα είναι τα παρακάτω:
- Χωρητικότητα του κάθε φορτηγού σε βάρος,
- Χωρητικότητα του κάθε φορτηγού σε όγκο.

Εκτός από τα δεδομένα των πελατών και του στόλου, δεχόμαστε ως δεδομένο ότι κάθε φορτηγό μπορεί να λειτουργήσει ως και οκτώ ώρες συνεχόμενες την μέρα. Σύμφωνα με το Προεδρικό Διάταγμα 27.6.1932 – Περί κωδικοποιήσεως και συμπληρώσεως των περί 8ώρου εργασίας διατάξεων, άρθρο 2, οι ώρες εργασίας κάθε εργαζομένου δεν πρέπει να υπερβαίνουν τις οκτώ ώρες τη μέρα και τις σαράντα οκτώ ανά βδομάδα.

3.2.1. Τα στοιχεία των πελατών

Για την ολοκλήρωση ενός προβλήματος δρομολόγησης οχημάτων θα πρέπει να έχουν ικανοποιηθεί κάποιοι συγκεκριμένοι όροι. Σε μία λύση του προβλήματος θα πρέπει να έχει ικανοποιηθεί εξ' ολοκλήρου η ζήτηση όλων των πελατών από τα φορτηγά που έχουν δρομολογηθεί. Κάθε πελάτης δέχεται επίσκεψη από ένα μόνο όχημα, το οποίο με αυτή τη μοναδική επίσκεψη ικανοποιεί πλήρως την ζήτησή του.

Για το πρόβλημά μας δόθηκε ένας πίνακας διαστάσεων 171*3 που περιέχει την ζήτηση του κάθε πελάτη σε μονάδες βάρους και όγκου. Τα στοιχεία αυτά επηρεάζουν σε μεγάλο βαθμό τη λύση μας λόγω της περιορισμένης χωρητικότητας των διαθέσιμων οχημάτων του στόλου μας. Στον πίνακα 3.1 φαίνονται τα πρώτα δέκα στοιχεία του δοσμένου πίνακα.

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

Πίνακας 3. 1 Οι ζητήσεις των δέκα πρώτων σε αρίθμηση πελατών.

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.

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. 2 Αποστάσεις σημείων εκφρασμένες σε μονάδες χρόνου (Seconds).

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

Πίνακας 3. 3 Αποστάσεις σημείων εκφρασμένες σε μέτρα (Meters).

Τέλος, για τον κάθε πελάτη γνωρίζουμε τα χρονικά περιθώρια μέσα στα οποία μπορεί να εξυπηρετηθεί αλλά και το χρονικό διάστημα που κάθε όχημα θα πρέπει να παραμείνει σε αυτόν ώστε να ολοκληρωθεί η διαδικασία κάλυψης της ζήτησής του. Επειδή χρησιμοποιούμε

`

αυστηρά χρονικά παράθυρα, δηλαδή τόσο η ώρα επίσκεψης του πελάτη αλλά και η εξυπηρέτησή του γίνεται εντός του δοσμένου χρονικού παραθύρου, για ευκολία πράξεων αφαιρούμε από το τέλος του κάθε παραθύρου τον χρόνο εξυπηρέτησης του συγκεκριμένου πελάτη. Στον πίνακα 3.4 δίνονται οι χρόνοι σε λεπτά των χρονικών παραθύρων, πριν από τις πράξεις, για τους πρώτους 10 πελάτες μέσα στην ίδια μέρα.

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

Πίνακας 3. 4 Χρονικά Παράθυρα και Χρόνοι Εξυπηρέτησης πελατών.

3.2.2. Τα στοιχεία των οχημάτων

Για τον στόλο των οχημάτων που είναι διαθέσιμα προς δρομολόγηση μας δόθηκαν οι μέγιστες χωρητικότητές τους σε βάρος και σε όγκο που μπορούν να μεταφέρουν. Στον πίνακα 3.5 παρουσιάζονται αυτά τα στοιχεία για τα πρώτα οκτώ φορτηγά του 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

Πίνακας 3. 5 Μέγιστες χωρητικότητες οχημάτων στόλου.

`

3.3 Μορφοποίηση του Μαθηματικού Μοντέλου για την επίλυση του προβλήματος

Για να επιλυθεί το πρόβλημα θα πρέπει να εκφραστεί σε ένα μαθηματικό μοντέλου γραμμικού προγραμματισμού. Αρχικά θα παρουσιάσουμε τους συμβολισμούς των δεδομένων που θα χρησιμοποιήσουμε και των μεταβλητών απόφασης που χρησιμοποιήθηκαν. Στη συνέχεια θα παρουσιάσουμε αναλυτικά τους περιορισμούς και την αντικειμενική συνάρτηση του μοντέλου. Τέλος, θα κάνουμε μια ανάλυση των περιορισμών. Σκοπός του μοντέλου είναι η ελαχιστοποίηση της απόστασης που θα διανύσουν τα οχήματα του στόλου μας μετά από την εξυπηρέτηση όλων των πελατών.

Οι συμβολισμοί των δεδομένων που έχουμε είναι οι εξής:

- **D1i** θα εκφράζει την ζήτηση του σημείου i σε αγαθά μετρημένο σε Kg (i ∈ (1, 171)),
- **D2i** με την σειρά του την αντίστοιχη ζήτησε του εκάστοτε σημείου i μετρημένο σε m^3 (i ∈ (1, 171)),
- **(TWSi , TWFi)** το χρονικό περιθώριο εξυπηρέτησης του κάθε πελάτη i (i ∈ (1 , 171)),
- **Ti,j** τον απαιτούμενο χρόνο για την μεταφορά από ένα σημείο διανομής i σε ένα σημείο διανομής j (i, j ∈ (0 , 171)) όπου για τις γραμμές και στήλες 0 συμβολίζεται το depot,
- **Di,j** την απαιτούμενη απόσταση μεταξύ των σημείων διανομής i και j (i, j ∈ (0 , 171)) όπου για τις γραμμές και στήλες 0 συμβολίζεται το depot,
- **Q1k** την μέγιστη χωρητικότητα του φορτηγού k σε kg (k ∈ (1, 105)),
- **Q2k** την μέγιστη χωρητικότητα του φορτηγού k σε m^3 (k ∈ (1, 105)),
- **MDi** τον χρόνο εξυπηρέτησης του πελάτη i (i ∈ N).

Για το μοντέλο μας θα χρειαστούμε τέσσερις μεταβλητές απόφασης. Αυτές είναι είτε κύριες, είτε βοηθητικές που προέκυψαν λόγω των απαιτήσεων του μοντέλου, και είναι οι 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.
- **ti,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.
- **Yk**: Πρόκειται για βοηθητική μεταβλητή της οποίας η ανάγκη εμφανίστηκε κατά την διαδικασία επίλυσης και σχετίζεται με την κάλυψη του οκταώρου. Παίρνει ακέραιες τιμές μεταξύ των αριθμών 0 και 1440 και συμβολίζει τον χρόνο τέλος της διαδρομής του οχήματος k. Στην περίπτωση που το όχημα k δεν εκτελεί κάποια διαδρομή, η συγκεκριμένη μεταβλητή παίρνει την τιμή 0.
- **Uk:** 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.

Στη συνέχεια παρουσιάζεται το αναλυτικό μαθηματικό μοντέλο του προβλήματος. Πιο συγκεκριμένα, θα αναφέρουμε τους περιορισμούς και την αντικειμενική συνάρτηση που χρησιμοποιήθηκε για την επίλυση του προβλήματος. Οι περιορισμοί προέκυψαν είτε από τη φύση του προβλήματος, είτε για προγραμματιστικούς λόγους.

`

Οι περιορισμοί είναι οι εξής:

1) $\sum_{i=0}^{imax} \sum_{k=1}^{kmax} Xi,j,k \ = 1$ ∀ j ∈ (1, jmax),
2) $\sum_{j=1}^{jmax} Xo,j,k \ \leq 1$ ∀ k ∈ (1, kmax),
3) $\sum_{i=0}^{imax} Xihk \ - \sum_{j=0}^{jmax} Xhjk$ =0 ∀ h ∈ (1, imax), k ∈ (1, kmax).

Η πρώτη ομάδα περιορισμών του προβλήματος , οι περιορισμοί (1), (2) και (3), εξασφαλίζουν την δρομολόγηση του στόλου οχημάτων της εταιρείας. Πιο συγκεκριμένα, ο πρώτος 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 κάνουν μόνο μία διαδρομή. Έπειτα, ο τρίτος περιορισμός του πακέτου δρομολόγησης εξασφαλίζει πως κάθε όχημα που επισκέπτεται κάποιον πελάτη φεύγει και από αυτόν.

4) $\sum_{i=0}^{171}\{\sum_{j=0}^{171} \mathrm{D1i} \ * Xi,j,k\} \leq \mathrm{Q1k}$ ∀ k,
$\sum_{i=0}^{171}\{\ \mathrm{D2i} \ * \sum_{j=0}^{171} Xi,j,k\} \leq \mathrm{Q2k}$ ∀ k.

Ο περιορισμός (4) εξασφαλίζει ότι οι ζητήσεις σε βάρος και όγκο που μεταφέρει το κάθε όχημα στους πελάτες που εξυπηρετεί δεν ξεπερνάνε την χωρητικότητά του.

5) $\mathrm{Ui} - \mathrm{Uj} + (\mathrm{N} - 1) * \mathrm{Xi,j,k} \ \leq \ \mathrm{N} - 2$ ∀ i≠j ∈ (1, imax), k ∈ (1, kmax).

Ο περιορισμός (5) χρησιμοποιείται για την εξάλειψη των υποδιαδρομών ενός οχήματος (sub- tour elimination constraint). Ουσιαστικά εξασφαλίζει ότι κάθε όχημα ξεκινάει από το depot, επισκέπτεται όλους του τους πελάτες και στο τέλος επιστρέφει στο depot ξανά.

6) $\mathrm{TWSi} \ * \sum_{j=0}^{max} Xi,j,k \leq ti,k$ ∀ i ∈ (1, imax), ∀ k ∈ (1, kmax),
$ti,k \leq \mathrm{TWFi} * \sum_{j=0}^{max} Xi,j,k$ ∀ i ∈ (1, imax), ∀ k ∈ (1, kmax),
7) $\mathrm{ti,k} \ \leq \mathrm{tj,k} - \mathrm{Tij} - \mathrm{MDi} + \mathrm{M} * (1 - \ \mathrm{Xi,j,k})$ ∀ i, j, k.

Οι περιορισμοί (6), (7) ικανοποιούν τις συνθήκες των χρονικών παραθύρων. Ο περιορισμός 6 εξασφαλίζει ότι η ώρα που θα επισκεφτεί το όχημα k τον πελάτη i είναι εντός του διαστήματος (TWSi,TWFi). Ο περιορισμός (7) δεν επιτρέπει στο μοντέλο να θέσει τον χρόνο επίσκεψης του πελάτη i μεγαλύτερο από το χρόνο επίσκεψης του πελάτη j, όταν γίνεται η διαδρομή από τον i στον j. Το M είναι ένας πολύ μεγάλος αριθμός.

8) $\mathrm{Yk} \geq \sum_{i=1}^{imax}(tik + Xi,0,k * (Ti,0 + MDi))$ ∀ k ∈ (1, kmax),
9) $\mathrm{Yk} \leq 1440 * \sum_{i=1}^{imax}(Xi,0,k)$ ∀ k ∈ (1, kmax),
10) $\mathrm{Yk} - \mathrm{t0,k} \leq 480$ ∀ k ∈ (1, kmax).

Οι τελευταίοι τρεις περιορισμοί προέκυψαν από την κάλυψη του οκταώρου. Οι περιορισμοί (8) και (9) βάζουν αντίστοιχα τα πάνω και κάτω όρια της βοηθητικής μεταβλητής Yk, η οποία συμβολίζει την ώρα που επισκέπτεται το φορτηγό k τον τελευταίο πελάτη της διαδρομής που ακολουθεί. Ο τελευταίος περιορισμός περιορίζει τον χρόνο του ταξιδιού του κάθε οχήματος στις οκτώ ώρες.

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

Minimize (Z)=$\sum_{i=0}^{imax} \sum_{j=0}^{jmax} \sum_{k=1}^{kmax}(Xi,j,k \ * Di,j)$,

και ελαχιστοποιεί την απόσταση που διανύουν τα οχημάτων.

3.4. πρώτη επίλυση του προβλήματος με τη χρήση κατάλληλου λογισμικού.

`

Την σωστή μορφοποίηση του προβλήματος ακολουθεί η χρήση κατάλληλου λογισμικού για την επίλυση του προβλήματος. Στα πλαίσια αυτής της εργασίας έγινε χρήση της γλώσσας C++ με βιβλιοθήκες της cplex. Η υλοποίηση του κώδικα και η επίλυση έλαβαν χώρα σε προσωπικό υπολογιστή με τα παρακάτω χαρακτηριστικά:

- Επεξεργαστής: AMD Ryzen 1600, 3.2GHz,
- Εγκατεστημένη μνήμη: 8GB,
- Λογισμικό: Windows 10 64-bit

Με την σωστή μορφοποίηση του προβλήματος, ο υπολογιστής εξετάζει όλες τις εφικτές λύσεις και τις συγκρίνει μεταξύ τους. Κάθε φορά που βρίσκει μία καλύτερη εφικτή λύση, αντικαθηστεί την παλιά. Η διαδικασία αυτή συνεχίζεται εωσότου έχει εξαντληθεί όλο το σύνολο των εφικτών λύσεων, οι οποίες περιλαμβάνουν τις διαδρομές που ακολουθεί το κάθε όχημα και την ώρα που επισκέπτεται τον κάθε πελάτη.

Λόγω του όγκου του πεδίου λύσεων και της φύσης του προβλήματος απαιτείται τεράστιος υπολογιστικός φόρτος και μνήμη. Για τον κάθε πελάτη που εισάγουμε στο πρόβλημα, η διάσταση του πεδίου λύσεων αυξάνεται εκθετικά. Για να μπορέσουμε, λοιπόν, να βγάλουμε μια συνολική πρώτη λύση για το πρόβλημα κάνουμε έναν χωρισμό του προβλήματος σε υποπροβλήματα.

Για να επιτευχθεί μια λογική προσέγγιση του προβλήματος έγινε μια κατηγοριοποίηση των δεδομένων των φορτηγών αλλά και των πελατών σε 19 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

`

19 ομάδες – υπό-προβλήματα – προς μελέτη. Έτσι, 19 ομάδες οχημάτων θα έχουν από 5 οχήματα.

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 19 ομάδες από 1 όχημα στον στόλο τους. Όταν ολοκληρωθεί η διαδικασία αυτή, επαναλαμβάνεται άλλες 4 φορές μέχρι όλες οι ομάδες να έχουν από 5 οχήματα.

Βήμα 4. Διορθωτικές (λογικές) αλλαγές

Η αρίθμηση των οχημάτων στον πίνακα 3.6 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) 1, 20, 39, 58, 78.

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

Πίνακας 3. 6 Ομαδοποίηση οχημάτων για αρχική επίλυση του προβλήματος.

Επειδή τα οχήματα τοποθετήθηκαν σε 19 ομάδες, τα σημεία διανομής θα χωριστούν και αυτά σε 19 ομάδες ώστε να βρεθεί μία πρώτη λύση που μελετά όλα τα δεδομένα. 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 9 πελάτες με αριθμούς 1 έως 9, στη $2^{η}$ αντίστοιχα οι πελάτες με αριθμούς 10 με 19, και συνεχίζει έτσι μέχρι τη τελευταία.

Πλέον, αντί για ένα μεγάλο πρόβλημα 171 πελατών μέσω 105 οχημάτων, επιλύονται τα 19 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 7η ομάδα πελατών η 7η ομάδα οχημάτων και ούτω καθ' εξής. 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 1 του παραρτήματος.

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. Στον Πίνακα 1 του παραρτήματος δίνονται πιο αναλυτικά και κατανοητά οι αντιστοιχίσεις πελατών και οχημάτων που προέκυψαν από την προγραμματιστική λύση καθώς και η σειρά που εκτελέστηκαν τα δρομολόγια. Οι τιμές των φορτηγών για την κάθε ομάδα k ανήκουν στο διάστημα [1+d, 5+d], όπου d= (k-1)*5.

3.5 Συμπεράσματα με βάση τα αποτελέσματα

Όπως είδαμε παραπάνω, κάναμε μία ομαδοποίηση των δεδομένων μας για να βρούμε μία εφικτή λύση για το πρόβλημα. Η λύση αυτή δεν είναι σε καμία περίπτωση βέλτιστη διότι η ομαδοποίηση που έγινε ήταν αυθαίρετη. Το αποτέλεσμα αυτό, όμως, μας βοηθάει να βγάλουμε κάποια συμπεράσματα ώστε να προχωρήσουμε σε μία καλύτερη λύση.

Στην πλειοψηφία των υποπροβλημάτων που επιλύθηκαν, παρατηρούμε ότι επιλέγεται ένα μεγάλο φορτηγό αντί πολλών μικρότερων. Για παράδειγμα, στις περιπτώσεις των υποομάδων 2, 12 και 13 προτιμήθηκαν από το μοντέλο μεγάλα φορτηγά για τις περισσότερες διαδρομές, με μικρότερα που καλύπτουν την υπολειπόμενη ζήτηση. Αυτό συμβαίνει γιατί τα σημεία μας έχουν μικρή διασπορά μεταξύ τους.

Στην περίπτωση, βέβαια, της τελευταίας υποομάδας παρατηρούμε ότι επιστρατεύτηκαν τρία φορτηγά, δύο μεγάλα και ένα μεσαίας χωρητικότητας. Όταν οι πελάτες έχουν πολύ μεγάλες ζητήσεις επιλέγεται η χρήση μεγάλων φορτηγών για την κάλυψή τους. Επίσης, όταν η απόσταση μεταξύ των πελατών είναι μεγάλη, το μοντέλο προτιμά να αναθέσει τη δρομολόγηση σε περισσότερα φορτηγά.

Από αυτά τα αποτελέσματα μπορούμε να συμπεράνουμε ότι η απόσταση μεταξύ των πελατών είναι αρκετά σημαντική. Πελάτες που βρίσκονται πιο κοντά μπορούν να εξυπηρετηθούν από τα ίδια οχήματα μειώνοντας την αθροιστική απόσταση που έχει να διανύσει ο στόλος μας. Συνεπώς, η πραγματοποίηση ενός γεωγραφικού διαχωρισμού των πελατών θα μας βοηθούσε να βελτιώσουμε σε μεγάλο βαθμό την λύση. Αυτό μπορεί να επιτευχθεί με την χρήση κατάλληλων αλγορίθμων ομαδοποίησης δεδομένων. Στο επόμενο κεφάλαιο αναπτύσσεται ένας τέτοιος αλγόριθμος.

`

ΚΕΦΑΛΑΙΟ 4: ΟΜΑΔΟΠΟΙΗΣΗ ΤΩΝ ΔΕΔΟΜΈΝΩΝ ΜΕΣΩ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ NEAREST POINT (NP)

4.1. Τεχνικές ομαδοποίησης δεδομένων – Εφαρμογή αλγορίθμων

4.1.1. Χρήση αλγορίθμων και τεχνικών στα προβλήματα

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, προβλήματα τέτοιου μεγέθους δεν είναι ικανό να επιλυθούν απευθείας μέσω του αναλυτικού μοντέλου. Αυτό συμβαίνει κυρίως όταν ο αριθμός των μεταβλητών απόφασης που πρέπει να πάρουν τιμή, καθώς και ο αριθμός των περιορισμών του προβλήματος, είναι αρκετά μεγάλος με αποτέλεσμα την δημιουργία ενός μεγάλου πεδίου διαφορετικών λύσεων.

Για την αντιμετώπιση των προβλημάτων αυτών υπάρχουν διαδικασίες επίλυσης ανάλογες του κάθε προβλήματος βελτιστοποίησης. Οι διαδικασίες αυτές ονομάζονται αλγόριθμοι. Ως αλγόριθμος ορίζεται μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, με σκοπό την επίλυση ενός προβλήματος. Στην μελέτη της επιχειρησιακής έρευνας και βελτιστοποίησης, οι κύριες κατηγορίες αλγορίθμων επίλυσης είναι οι ευρετικοί (heuristics) και οι μετευρετικοί (meta-heuristics).

Heuristics: Οι ευρετικές διαδικασίες που ακολουθούνται εξαρτώνται κάθε φορά από το πρόβλημα. Κάθε διαδικασία προσαρμόζεται στο πρόβλημα προσπαθώντας να αξιοποιήσει τις ιδιαιτερότητές του. Αυτές οι διαδικασίες βρίσκουν εφαρμογή σε μεγάλα και δύσκολα προβλήματα. Συνήθως βρίσκουν γρήγορα μια εφικτή λύση, την οποία προσπαθούν σε δεύτερο χρόνο να βελτιώσουν. Βέβαια, ενώ αυτές οι διαδικασίες μικραίνουν το χρόνο υπολογισμού και έχουν καλές λύσεις, δεν εγγυώνται βελτιστότητα.

Meta-Heuristics: Ο όρος Meta-Heuristics προτάθηκε από τον Glover στα μέσα της δεκαετίας του ’80 ως οικογένεια αλγορίθμων αναζήτησης ικανών να ορίσουν λύσεις υψηλού επιπέδου που χρησιμοποιείται για την καθοδήγηση άλλων heuristics σε μια καλύτερη εξέλιξη στον χώρο της 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 που τους επιτρέπει να διερευνήσουν πιο διεξοδικά το χώρο λύσης και έτσι να βρουν μια καλύτερη. Αυτή η νέα λύση μπορεί να συμπίπτει και με το 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 εφαρμόζεται ομαδοποίηση των δεδομένων μας μέσω του αλγορίθμου που αναπτύχθηκε για την μετατροπή του προβλήματός μας από CVRP με χρονικά παράθυρα σε TSP με χρονικά παράθυρα.

4.1.2. Ομαδοποίηση των δεδομένων με τον αλγόριθμο Nearest Point.

Όπως αναφέρθηκε στα προηγούμενα κεφάλαια, το κύριο θέμα που πρέπει να αντιμετωπίσουμε στο πρόβλημά μας είναι πως ο χώρος των λύσεων είναι πάρα πολύ μεγάλος. Έχουμε 171 διαφορετικά σημεία (κόμβους) των οποίων πρέπει να ικανοποιηθούν οι ζητήσεις

`

από τον ετερογενή στόλο 105 φορτηγών διαφορετικών και πεπερασμένων χωρητικοτήτων. Για να μελετηθεί όλο το πεδίο εφικτών λύσεων απαιτείται πολύς υπολογιστικός χρόνος και χώρος.

Επανερχόμαστε, λοιπόν, στην στρατηγική που προαναφέραμε στο κεφάλαιο 3, την ομαδοποίηση δηλαδή των δεδομένων μας και την επίλυση πολλών μικρότερων προβλημάτων. Η ομαδοποίηση αυτή δεν θα γίνει μυωπικά αλλά με τη χρήση του κατάλληλου αλγορίθμου ομαδοποίησης.

Οι αλγόριθμοι ομαδοποίησης αναλύουν φυσικές ομάδες δεδομένων με βάση κάποιες ομοιότητες που έχουν. Στην περίπτωσή του αλγορίθμου Nearest Point συνδέουμε τα σημεία μας με βάση την σχετική τους απόσταση. Για την αποτελεσματική ομαδοποίηση, ο αλγόριθμος αξιοποιεί την απόσταση του κάθε σημείου μας από το κέντρο της κάθε ομάδας σημείων που δημιουργείται. Στην ομαδοποίηση αυτών λαμβάνονται υπόψιν και τα χρονικά παράθυρα τους.

Στόχος της ομαδοποίησης αυτής είναι η μετατροπή του προβλήματος VRP σε TSP. Ο αλγόριθμος δημιουργεί ομάδες σημείων που έχουν τουλάχιστον μία εφικτή λύση. Βέβαια, ο αλγόριθμος αυτός είναι άπληστος. Με αυτό τον τρόπο θα μειωθεί δραματικά το πεδίο λύσεων του αναλυτικού προβλήματος που έχουμε αλλά δεν μας εγγυάται βελτιστότητα.

4.1.3. Λειτουργία αλγορίθμου Nearest Point.

Ο αλγόριθμος Nearest Point ομαδοποιεί τα σημεία μας σειριακά, λαμβάνοντας υπόψιν τις γεωγραφικές αποστάσεις τους αλλά και την επικαλυψιμότητα των χρονικών παραθύρων τους. Μετά την ολοκλήρωση της κάθε ομάδας ξεκινά η ανάθεση των κόμβων στην επόμενη. Ο σκοπός του αλγορίθμου είναι να μας δώσει τα σημεία της κάθε διαδρομής και το φορτηγό που ικανοποιεί τη ζήτησή της.

Βήμα 1: Υπολογισμός στατιστικών σταθερών του προβλήματος.

Το πρώτο βήμα είναι ο υπολογισμός του κέντρου βάρους των κόμβων του προβλήματός μας. Αρχικά, γίνεται προβολή των σημείων σε ένα καρτεσιανό επίπεδο σύμφωνα με τις γεωγραφικές συντεταγμένες τους, οι οποίες είναι δεδομένες. Ο υπολογισμός της μέσης απόστασης των σημείων γίνεται απλά, σύμφωνα με τον τύπο:

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{n} D\,ij$$

Όπου N είναι ο αριθμός των κόμβων και Dij η απόσταση μεταξύ των κόμβων i και j. Στη συνέχεια υπολογίζουμε την τυπική απόκλιση του συνόλου των αποστάσεων από τον τύπο:

$$s = \sqrt{\frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{n} (\mathrm{D\,ij} - \overline{x})^2}\ .$$

Τέλος, βρίσκουμε την τιμή του συντελεστή μεταβολής CV του συνόλου από τον τύπο:

$$CV = \frac{s}{\overline{x}}\ .$$

Οι στατιστικές σταθερές που αναφέρθηκαν παραπάνω χρησιμοποιούνται στον αλγόριθμο για την καλύτερη λειτουργεία του αλγορίθμου.

Βήμα 2: Τοποθέτηση κόμβων σε διαδρομές.

Στο δεύτερο βήμα, χρησιμοποιείται μία ευρετική επαναληπτική διαδικασία με σκοπό την ομαδοποίηση των κόμβων. Αυτό θα μειώσει σε μεγάλο βαθμό την πολυπλοκότητα του

`

αναλυτικού μοντέλου, μετατρέποντας το πρόβλημά μας από CVRP με χρονικά παράθυρα σε πολλά TSP με χρονικά παράθυρα. Κύριος παράγοντας για την ομαδοποίηση αυτή είναι η σχετική απόσταση των πελατών.

Βήμα 2.1: Ανάθεση πρώτου κόμβου.

Για να ξεκινήσει η διαδικασία χρειαζόμαστε έναν αρχικό κόμβο. Αυτός είναι, κάθε φορά, ο πιο απομακρυσμένος από το κέντρο βάρους του βήματος 1, ο οποίος δεν έχει ανατεθεί σε ήδη υπάρχουσα διαδρομή. Με αυτόν τον τρόπο, κάθε φορά που ολοκληρώνουμε μία ομάδα, έχουμε όλο και μικρότερο εύρος στο σύνολο των υπολειπόμενων κόμβων.

Αφού επιλέξουμε τον αρχικό κόμβο της ομάδας, εισάγουμε νέους πελάτες στην ομάδα με βάση την απόστασή τους από το κέντρο βάρος των κόμβων της διαδρομής.

Βήμα 2.2: Διαδικασία ανάθεσης

Αρχικά, υπολογίζουμε το κέντρο βάρος των κόμβων που έχουν ανατεθεί στην διαδρομή μας τη συγκεκριμένη χρονική στιγμή. Στη συνέχεια, επιλέγουμε τον πιο κοντινό κόμβο που δεν έχει ανατεθεί σε κάποια διαδρομή ήδη και ελέγχουμε την συμβατότητά του με την διαδρομή που δημιουργούμε.

Η συμβατότητα ενός κόμβου με μία διαδρομή έχει να κάνει με την επικαλυψημότητα των χρονικών παραθύρων των κοντινών κόμβων που έχουν ήδη ανατεθεί στη διαδρομή. Σε περίπτωση που τα χρονικά παράθυρα του υποψήφιου κόμβου και ενός από τους τρεις πιο κοντινούς κόμβους του, που έχουν ήδη ανατεθεί στην διαδρομή, έχουν επικάλυψη ίση ή μεγαλύτερη από τον χρόνο εξυπηρέτησης και των δύο, ο υποψήφιος κόμβος είναι συμβατός. Όμως, το ότι ένας κόμβος είναι συμβατός με μία διαδρομή δεν σημαίνει ότι θα εισαχθεί κιόλας σε αυτή. Για να γίνει αυτό πρέπει να γίνουν έλεγχοι για τη χωρητικότητα των φορτηγών και για την συνθήκη του οκταώρου.

Για τον πρώτο έλεγχο, προσθέτουμε τις ζητήσεις των ήδη ανατεθειμένων κόμβων αλλά και του υποψηφίου, ως προς βάρος και όγκο. Αυτά τα αθροίσματα τα συγκρίνουμε με τις αντίστοιχες χωρητικότητες του μεγαλύτερου αχρησιμοποίητου φορτηγού. Άμα οι αθροιστικές ζητήσεις είναι μικρότερες από τις χωρητικότητες του μεγαλύτερου φορτηγού αντίστοιχα ο έλεγχος χωρητικότητας είναι θετικός.

Για τον έλεγχο του οκταώρου, χρησιμοποιούμε τρεις παράγοντες: τους χρόνους εξυπηρέτησης των κόμβων, τον μέσο όρο των αποστάσεών τους αλλά και τις αποστάσεις τους από το κέντρο εξυπηρέτησης. Για τον πρώτο όρο, αθροίζουμε τους χρόνους εξυπηρέτησης των κόμβων. Για τον δεύτερο, διαιρούμε το άθροισμα των χρόνων μεταξύ του κάθε ζεύγους κόμβων της διαδρομής με τον αριθμό των κόμβων αυτής πολλαπλασιασμένο επί δύο. Αυτό συμβαίνει γιατί έχουμε υπολογίσει αμφίδρομες διαδρομές χωρίς να συμβαίνει στην πραγματικότητα. Τέλος, για τον τρίτο όρο, χρησιμοποιούμε τον μέσο χρόνο επιστροφής στο depot. Αυτός ο όρος διαιρείται με έναν σταθερό όρο που προκύπτει από τον συντελεστή μεταβολής του προβλήματος που έχουμε. Αυτό συμβαίνει για την καλύτερη δημιουργία των ομάδων και την σύσφιξη της εν δυνάμει λύσης.

Στην περίπτωση που ικανοποιούνται όλες οι συνθήκες εισάγεται ο κόμβος στην διαδρομή και επαναλαμβάνουμε την διαδικασία. Όταν δεν υπάρχουν άλλοι κόμβοι που να ικανοποιούν όλες τις συνθήκες ολοκληρώνεται η διαδρομή.

Βήμα 3: Υπολογισμός συνολικής ζήτησης

Με την ολοκλήρωση μιας διαδρομής υπολογίζουμε την συνολική ζήτηση της διαδρομής.

`

Βήμα 4: Επιλογή φορτηγού εξυπηρέτησης της συγκεκριμένης ομάδας

Στο βήμα 4 επιλέγουμε ποιο φορτηγό θα εξυπηρετήσει την ομάδα που δημιουργήθηκε. Το φορτηγό που επιλέγεται είναι το μικρότερο φορτηγό που μπορεί να καλύψει τις συγκεκριμένες ζητήσεις που υπολογίσθηκαν στο βήμα 3, χωρίς να υπερβαίνεται κάποια από τις χωρητικότητες.

Βήμα 5: Επανάληψη βημάτων 2, 3, 4.

Στην συνέχεια επαναλαμβάνουμε την διαδικασία από το βήμα 2 ως το βήμα 4. Όταν έχουμε αναθέσει όλους τους κόμβους σε διαδρομές τερματίζει ο αλγόριθμος.

Βήμα 6: Αναλυτικό μοντέλο

Τελευταίο βήμα του αλγορίθμου είναι η επίλυση του αναλυτικού μοντέλου για την κάθε ομάδα πελατών και οχήματος. Με αυτόν τον τρόπο έχουμε περιορίσει σε πολύ μεγάλο βαθμό το πλήθος των μεταβλητών απόφασης και περιορισμών που χρησιμοποιούμε. Άρα, ο υπολογιστικός φόρτος του προβλήματος είναι πολύ μικρότερος από πριν.

Το μοντέλο που χρησιμοποιούμε είναι αυτό που φαίνεται και στο κεφάλαιο (3.3), αλλά απλοποιημένο. Αρχικά, μέσω της ευρετικής διαδικασίας έχουμε καθορίσει πιο όχημα θα εξυπηρετήσει την κάθε διαδρομή. Ο δείκτης k, λοιπόν, του μοντέλου έχει σταθερή τιμή ίση με $\{\alpha\}$, όπου a είναι ο αριθμός της ομάδας που βελτιστοποιούμε τη διαδρομή. Επιπλέον, δεν χρειαζόμαστε τους περιορισμούς που σχετίζει την ζήτηση των πελατών μιας διαδρομής και την χωρητικότητα του οχήματος που την εξυπηρετεί.

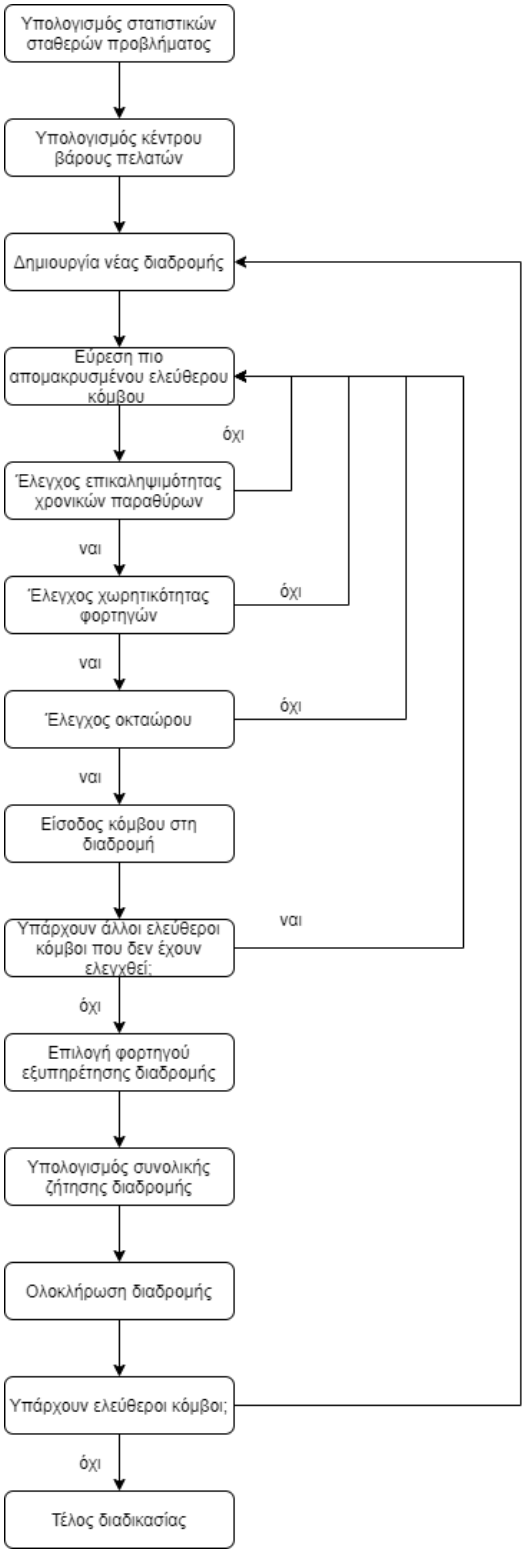
Το μετασχηματισμένο μοντέλο για την κάθε ομάδα είναι το εξής:

1)	$\sum_{i=0}^{imax}\sum_{k=1}^{kmax} Xi,j,k \ = 1$	∀ j ∈ N (1, jmax),
2)	$\sum_{j=1}^{jmax} Xo,j,k \ \leq 1$	∀ k = {α},
3)	$\sum_{i=0}^{imax} Xihk \ - \sum_{j=0}^{jmax} Xhjk$ =0	∀ h ∈ (1, imax), k = {α},
4)	$\mathrm{Ui} - \mathrm{Uj} + (\mathrm{N}-1) * \mathrm{Xi,j,k} \leq \mathrm{N}-2$	∀ i≠j ∈ (1, imax), k = {α},
5)	$\mathrm{TWSi} * \sum_{j=0}^{max} Xi,j,k \leq ti,k$	∀ i ∈ (1, imax), ∀ k = {α},
	$ti,k \leq \mathrm{TWFi} * \sum_{j=0}^{max} Xi,j,k$	∀ i ∈ (1, imax), ∀ k = {α},
6)	$\mathrm{ti,k} \leq \mathrm{tj,k} - \mathrm{Tij} - \mathrm{MDi} + \mathrm{M} * (1 - \mathrm{Xi,j,k})$	∀ i ∈ (1, imax), ∀ j ∈ (1, jmax), ∀ k={α}
7)	$\mathrm{Yk} \geq \sum_{i=1}^{imax}(tik + Xi,0,k * (Ti,0 + MDi))$	∀ k = {α},
8)	$\mathrm{Yk} \leq 1440 * \sum_{i=1}^{imax}(Xi,0,k)$	∀ k = {α},
9)	$\mathrm{Yk} - \mathrm{t0,k} \leq 480$	∀ k = {α},

με αντικειμενική συνάρτηση: Minimize (Z)=$\sum_{i=0}^{imax}\sum_{j=0}^{jmax}(Xi,j,a \ * Di,j)$.

Στο μοντέλο αυτό, η τιμή των imax, jmax παίρνει τιμή ανάλογα με το πλήθος των πελατών της κάθε ομάδας.

`



Σχήμα 4. 1 Διάγραμμα ροής για βήματα 1-5

`

4.2. Πακέτων δεδομένων

Για να δούμε ότι ο αλγόριθμος λειτουργεί σωστά αλλά και να βγάλουμε συμπεράσματα είναι απαραίτητη η δημιουργία κάποιων επιπλέον πακέτων δεδομένων. Τα πακέτα δεδομένων που θα δημιουργήσουμε πρέπει να είναι συγκρίσιμα με τα δεδομένα του φυσικού προβλήματος. Τα δεδομένα που χρειαζόμαστε για το κάθε πακέτο δεδομένων είναι τα εξής:

- Δεδομένα πελάτη:
  - ζήτηση ως προς βάρος και όγκο,
  - απαιτούμενος χρόνος εξυπηρέτησης,
  - χρονικό παράθυρο εξυπηρέτησης,
- δεδομένα φορτηγών:
  - χωρητικότητα ως προς βάρος και όγκο,
- δεδομένα προβλήματος:
  - αποστάσεις μεταξύ πελατών,
  - χρόνοι μεταξύ πελατών.

Για να είναι συγκρίσιμα τα δεδομένα πρέπει να δημιουργήσουμε ομάδες 170 πελατών και μιας βάσης. Ο κάθε πελάτης πρέπει να έχει την δικιά του ζήτηση ως προς βάρος και όγκο, καθώς, και χρονικά παράθυρα για την εξυπηρέτησή του. Τέλος, πρέπει να έχει συγκεκριμένο χρόνο εξυπηρέτησης.

Επιπλέον, τα δεδομένα αυτά πρέπει να αντικατοπτρίζουν την πραγματικότητα. Για τον σκοπό αυτό έγινε μια ανάλυση των δεδομένων του φυσικού προβλήματος ώστε να εγκαθιδρυθούν κάποιοι κανόνες για τα δεδομένα ζήτησης και χρόνου εξυπηρέτησης του κάθε πελάτη. Τα δεδομένα των χρονικών παραθύρων τα αφήσαμε ίδια διότι εισάγουν από μόνα τους μεγάλη τυχαιότητα όταν μοιραστούν στους πελάτες.

4.2.1. Δημιουργία πακέτων δεδομένων.

Πρώτο βήμα για τη δημιουργία των πακέτων δεδομένων είναι η τυχαία επιλογή γεωγραφικών σημείων εντός ενός χωρίου. Επιλέξαμε τέσσερα διαφορετικά χωρία διαφορετικών μεγεθών σε διαφορετικά μέρη του κόσμου, στα Βερολίνο, Ρώμη, Μόσχα και Παρίσι, και με μία γεννήτρια τυχαίων σημείων εντός χωρίου τα δημιουργούμε. Κάθε πακέτο δεδομένων έχει 171 σημεία, όπου 170 είναι πελάτες και το τελευταίο είναι η βάση μεταφοράς. Στον πίνακα 4.1 φαίνονται τα κατανεμημένα σημεία σε κλίμακα 1:400.000.

Για να υπολογίσουμε την απόσταση μεταξύ οποιονδήποτε δύο σημείων χρησιμοποιούμε τη φόρμουλα haversine. Η φόρμουλα αυτή υπολογίζει την απόσταση αυτών των σημείων θεωρώντας ότι η γη είναι σφαιρική, αγνοώντας τα σφάλματα από το ελλειψοειδές σχήμα της. Τα σφάλματα αυτά, επειδή η απόσταση των σημείων είναι μικρή συγκριτικά με την ακτίνα της γης, μπορούμε να τα θεωρήσουμε αμελητέα. Η απόσταση των σημείων υπολογίζεται από τον παρακάτω τύπο:

$$d = 2 * r * \arcsin(\sqrt{a}),$$

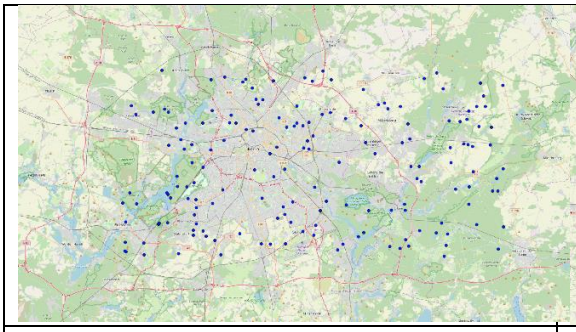
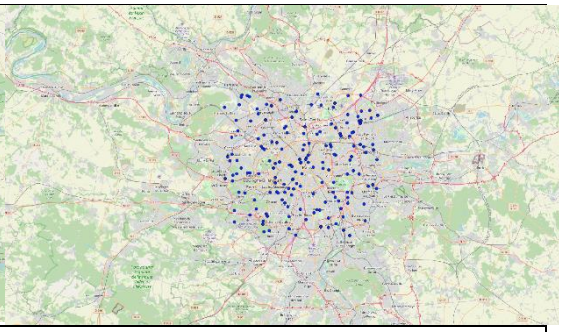
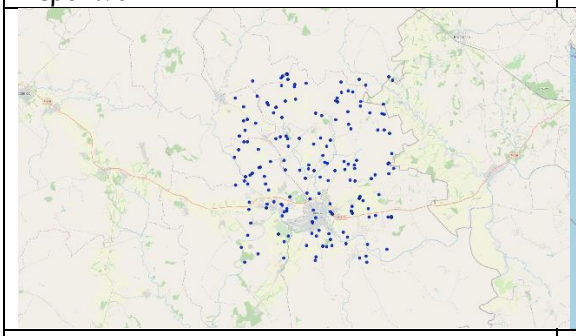
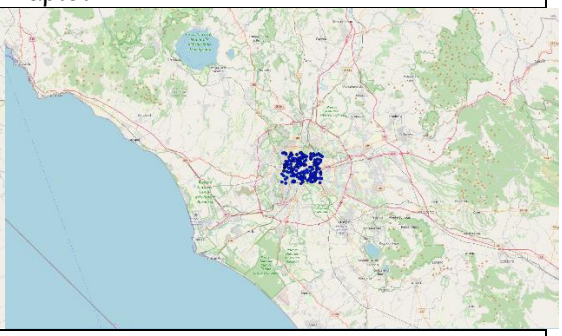
όπου: d η απόσταση, r η ακτίνα της γης και α που υπολογίζεται από τον τύπο:

$$a = \sin^2(\frac{\Delta\varphi}{2}) + \cos\varphi_1 * \cos\varphi_2 * \sin^2(\frac{\Delta\lambda}{2}),$$

όπου: Δφ η διαφορά των γεωμετρικών πλατών και Δλ η διαφορά των γεωμετρικών μηκών των δύο σημείων.

`

Αφού υπολογίσουμε με αυτόν τον τρόπο όλες τις αποστάσεις μεταξύ των σημείων του κάθε πακέτου δεδομένων, σειρά έχει ο πίνακας του χρόνου μετάβασης μεταξύ αυτών των σημείων. Αυτόν τον πίνακα τον δημιουργήσαμε απλώς διαιρώντας την απόστασή τους με μία μέση ταχύτητα των 50 χιλιομέτρων ανά ώρα.

	
Βερολίνο	Παρίσι
	
Μόσχα	Ρώμη

Πίνακας 4. 1 Χάρτες κατανεμημένων σημείων στις πόλεις.

Τέλος, έχουμε την παραγωγή των υπόλοιπων στοιχείων για τον κάθε πελάτη. Τα στοιχεία αυτά είναι η ζήτηση ως προς όγκο, η ζήτηση ως προς βάρος και ο χρόνος εξυπηρέτησης. Τα δεδομένα αυτά παράγονται τυχαία με μέγιστες τιμές τις αντίστοιχες του φυσικού προβλήματος και τον κάθε τύπο δεδομένων. Πιο συγκεκριμένα, η ζήτηση ως προς όγκο των πελατών μας παίρνει τυχαία τιμή στο διάστημα (0, 4000], η ζήτηση ως προς βάρος στο διάστημα (0, 1000] και ο χρόνος εξυπηρέτησης στο διάστημα (0, 120].

Αξίζει να σημειωθεί ότι στόλος των φορτηγών παραμένει ο ίδιος για την λύση του κάθε προβλήματος. Σκοπός μας όταν δημιουργήσαμε τα νέα δεδομένα είναι να δούμε ότι μπορεί να εφαρμοστεί το μοντέλο μας σε πολλά προβλήματα και ότι δεν παρουσιάζονται απλώς λύσεις στο συγκεκριμένο φυσικό πρόβλημα. Αυτό, λοιπόν, το σετ οχημάτων είναι αρκετά μεγάλο αλλά και διαφοροποιούμενο από όχημα σε όχημα ώστε να μην χρειάζεται αλλαγή.

4.2.2. Αρχικές λύσεις πακέτων δεδομένων.

Στον παρακάτω πίνακα φαίνονται τα δεδομένα των αρχικών λύσεων των πακέτων δεδομένων που δημιουργήθηκαν με βάση τη μεθοδολογία του κεφαλαίου 3.

`

	Αθήνα	Βερολίνο	Παρίσι	Μόσχα	Ρώμη
Συνολική απόσταση ($Km*10^3$)	2,59441	2,77985	1,19766	1,48748	0,320529
Αριθμός φορτηγών	27	31	26	29	16
Μέσος όρος χιλιομέτρων ανά φορτηγό	96,1	89,7	46,06	59,3	20,03
Μέγιστη-ελάχιστη απόσταση	204,805	291,001	102,956	132,588	23,743
	65,209	138,826	55,264	76,444	17,076
Ποσοστό κάλυψης χωρητικότητας α κατά μέσο όρο (%)	47	25	30	25	34
Ποσοστό κάλυψης χωρητικότητας β κατά μέσο όρο	59	74	78	75	42
Μέσος αριθμός πελατών ανά φορτηγό	6,30	5,45	6,5	5,83	10,56
Μέγιστος-ελάχιστος αριθμός πελατών ανά διαδρομή	9	9	9	9	9
	1	1	1	1	9
Αποτυχία επίλυσης	0	4	4	4	3
Χρόνος επίλυσης (sec)	29	38	11	14	8

Πίνακας 4. 2 Δεδομένα αρχικών λύσεων των πακέτων δεδομένων.

`

Στις γραμμές του πίνακα 4.2. φαίνονται για την κάθε πόλη κάποια στοιχεία. Τα στοιχεία αυτά είναι η συνολική απόσταση που διανύουν τα οχήματα του στόλου μας, ο αριθμός των οχημάτων που χρησιμοποιήθηκαν, χιλιομετρικός μέσος όρος της διανυθείσας απόστασης του κάθε οχήματος, η χιλιομετρικά μεγαλύτερη και μικρότερη διαδρομή, το μέσο ποσοστό κάλυψης της χωρητικότητας α και β των οχημάτων, ο μέσος αριθμός πελατών ανά όχημα, ο μέγιστος και ελάχιστος αριθμός πελατών του συνόλου των διαδρομών του κάθε πακέτου, τα προβλήματα που δεν κατάφερε το πρόγραμμα να βελτιστοποιήσει και ο χρόνος που χρειάστηκε να τρέξει το πρόγραμμα για να μας δώσει αυτά τα αποτελέσματα. Στο επόμενο κεφάλαιο, θα παρουσιαστούν τα αποτελέσματα του αλγορίθμου Nearest Point με ακριβώς την ίδια δομή ώστε να γίνει σύγκριση.

4.3. Αποτελέσματα του αλγορίθμου Nearest Point.

Στο παρών κεφάλαιο θα παρουσιαστούν τα αποτελέσματα των πακέτων δεδομένων που δημιουργήθηκαν αλλά και του φυσικού προβλήματος. Τα αποτελέσματα του αλγορίθμου προκύπτουν μετά την χρήση κατάλληλου λογισμικού. Όπως αναφέρθηκε και στο τρίτο κεφάλαιο, έγινε χρήση της γλώσσας C++ με βιβλιοθήκες cplex. Η υλοποίηση του κώδικα και η επίλυση έλαβαν χώρα σε ηλεκτρονικό υπολογιστή με τα παρακάτω χαρακτηριστικά:

- Επεξεργαστής: AMD Ryzen 1600, 3.2GHz,
- Εγκατεστημένη μνήμη: 8GB,
- Λογισμικό: Windows 10 64-bit.

Για την εξαγωγή συμπερασμάτων θα πρέπει να αναφερθούμε στα στατιστικά χαρακτηριστικά του κάθε σετ δεδομένων. Τα χαρακτηριστικά αυτά είναι η μέση απόσταση των κόμβων του κάθε πακέτου καθώς και οι τυπικές αποκλίσεις και οι συντελεστές μεταβλητότητας. Στον πίνακα 4.3.1. φαίνονται τα στοιχεία των πακέτων δεδομένων αλλά και του φυσικού προβλήματος.

Πόλη	Αριθμός πελατών	Συνολική ζήτηση (α, β)	Μέση απόσταση (m)	Τυπική απόκλιση αποστάσεων s	Συντελεστής μεταβλητότητας αποστάσεων CV
Αθήνα	171	79.271, 9.125	20.928,6	0,722572	6,8903
Βερολίνο	170	140.956, 32.156	26.848,2	0,542075	6,1683
Παρίσι	170	143.994, 28.194	13.386,7	0,483952	5,9358
Μόσχα	170	129.169, 29,569	15.297,2	0,479135	5,9165
Ρώμη	170	142.282, 29.965	3.831,08	0,485341	5,9414

Πίνακας 4.3.1. Στατιστικά και αριθμητικά χαρακτηριστικά πακέτων δεδομένων

Τα στοιχεία των πακέτων δεδομένων είναι μεγάλο βαθμό ομοιομορφίας, όπως φαίνεται και στον πίνακα 4.3.1.. Όλα τα πακέτα, ενώ διαφέρουν στη μέση απόσταση των κόμβων, έχουν τυπικές αποκλίσεις που διαφέρουν μέχρι και 7% και συντελεστές μεταβλητότητας με πολύ μικρές αποκλίσεις, της τάξης του 3%. Αυτό, βέβαια, θα αποτυπώνεται και στα αποτελέσματα, όπου υποθέτουμε ότι λόγω της καλύτερης διασποράς των κόμβων θα έχουμε πιο ομοιόμορφη κατανομή τους σε ομάδες. Στον πίνακα 4.3 φαίνονται τα αποτελέσματα των πακέτων δεδομένων αλλά και του φυσικού προβλήματος.

`

	Αθήνα	Βερολίνο	Παρίσι	Μόσχα	Ρώμη
Συνολική απόσταση ($Km*10^3$)	1,25994	2,29522	0,85749	1,34989	0,107918
Αριθμός φορτηγών	16	33	30	24	9
Μέσος όρος χιλιομέτρων ανά φορτηγό	78,746	69,552	28,583	56,245	11,990
Μέγιστη-ελάχιστη απόσταση	160,123	136,625	59,894	73,303	18,433
	45,863	30,583	4,529	14,577	2,833
Ποσοστό κάλυψης χωρητικότητας α κατά μέσο όρο (%)	45,30	19,35	22,89	10,14	49,51
Ποσοστό κάλυψης χωρητικότητας β κατά μέσο όρο	79,69	83,74	82,44	39,61	93,94
Μέσος αριθμός πελατών ανά φορτηγό	10,625	5,15	5,63	7,04	18,78
Μέγιστος-ελάχιστος αριθμός πελατών ανά διαδρομή	17	10	13	14	23
	2	1	2	1	7
Αποτυχία επίλυσης	0	0	0	0	2
Χρόνος επίλυσης (sec)	58	7	9	17	153

Πίνακας 4. 3 Αποτελέσματα αλγορίθμου Nearest Point.

Από τους πίνακες 4.2 και 4.3 φαίνεται πως έχουμε μεγάλη βελτίωση στα αποτελέσματά μας.

`

Αρχικά, έχουμε 29,95% μείωση των αποστάσεων που χρειάζονται να διανύσουν αθροιστικά τα οχήματα του στόλου μας τα οποία έχουν δρομολογηθεί. Τα οχήματα έχουν δει μείωση τα τάξης του 13,18% και διανύουν κατά μέσο όρο 21,23% μικρότερες αποστάσεις.

Οι μεγαλύτερες και οι μικρότερες διαδρομές που δρομολογούνται για το κάθε πακέτο δεδομένων έχουν μειωθεί επίσης. Πιο συγκεκριμένα, οι μεγαλύτερες διαδρομές έχουν μειωθεί κατά 40,62% ενώ οι μικρότερες διαδρομές έχουν μειωθεί κατά 72,11%. Τα παραπάνω ποσοστά είναι κατά μέσο όρο για όλα τα πακέτα δεδομένων που έχουμε δημιουργήσει.

Τα ποσοστά κάλυψης χωρητικότητας α και β των οχημάτων μας χρειάζονται αύξηση για να θεωρήσουμε ότι έχουμε θετικό αποτέλεσμα. Όσον αφορά το ποσοστό πλήρωσης των φορτηγών ως προς το βάρος, βλέπουμε μείωση κατά 8,58%, ενώ παρατηρείται αύξηση του ποσοστού πλήρωσης των οχημάτων ως προς τον όγκο κατά 14,63%.

Η μείωση των αποστάσεων υποδεικνύει πύκνωση των διαδρομών μας αφού έχουμε δρομολόγηση όλων των πελατών. Στις μεγαλύτερες διαδρομές του κάθε πακέτου δεδομένων έχουμε αύξηση των πελατών κατά 71,11% ενώ στις μικρότερες δεν έχουμε μεταβολή των τιμών. Άρα η θεωρία της πύκνωσης επαληθεύεται.

Τέλος, παρατηρείται μεγάλη μείωση των αναλυτικών προβλημάτων που δεν παίρνουν λύση. Ενώ στις αρχικές λύσεις είχαμε αθροιστικά 15 προβλήματα που δεν είχαν εφικτές λύσεις, στον αλγόριθμο nearest point έχουμε μόνο 2. Αυτό, βέβαια, έχει αντίκτυπο και στον υπολογιστικό χρόνο που χρειάζεται ο δεύτερος αλγόριθμος να υλοποιηθεί, όπου είναι 144% περισσότερος από τον πρώτο, μη διακριτό τρόπο.

Ο λόγος που παρατηρούμε μη εφικτά προβλήματα κατά την αναλυτική επίλυση των TSP είναι η χαλαρότητα με την οποία έχουμε επιλέξει τις διαδρομές. Αυτή η μη εφικτότητα μπορεί εύκολα να αναιρεθεί απλώς συσφίγγοντας τους περιορισμούς που έχουμε θέσει για την δημιουργία διαδρομών. Στον πίνακα 4.4 φαίνονται τα νέα αποτελέσματα για τα δεδομένα της Ρώμης με την μόνη αλλαγή στον περιορισμό του οκταώρου όπου δημιουργούμε διαδρομές που δεν ξεπερνούν κατά μέσο όρο τις 7 ώρες από τις αρχικές 8 που είχαμε επιλέξει.

Όπως παρατηρούμε, με μία πολύ μικρή αλλαγή έχουμε μια αρκετά καλύτερη λύση για το πρόβλημα. Η νέα λύση, όχι μόνο είναι εφικτή, αλλά λόγω της συγκεκριμένης σύσφιξης βελτίωσε την προηγούμενη χιλιομετρικά. Κατά μέσο όρο το κάθε φορτηγό εκτελεί διαδρομή 10,18% μικρότερη από την πρώτη λύση. Επιπλέον, παρατηρείται μεγαλύτερη πλήρωση στα οχήματα που δρομολογούνται. Τέλος, ο χρόνος επίλυσης του προβλήματος πήγε από τα 135 δευτερόλεπτα στα 9.

`

	Ρώμη (συνθήκη οκταώρου 480sec)	Ρώμη (συνθήκη οκταώρου 420sec)
Συνολική απόσταση ($Km*10^3$)	0,107918	0,140012
Αριθμός φορτηγών	9	13
Μέσος όρος χιλιομέτρων ανά φορτηγό	11,990	10,770
Μέγιστη- ελάχιστη απόσταση	18,433	17,040
	2,833	2,833
Ποσοστό κάλυψης χωρητικότητας α κατά μέσο όρο (%)	49,51	53,03
Ποσοστό κάλυψης χωρητικότητας β κατά μέσο όρο (%)	93,94	94,92
Μέσος αριθμός πελατών ανά φορτηγό	18,78	13
Μέγιστος- ελάχιστος αριθμός πελατών ανά διαδρομή	23	20
	7	7
Αποτυχία επίλυσης	2	0
Χρόνος επίλυσης (sec)	153	9

Πίνακας 4. 4 Συγκριτικά αποτελέσματα για την πόλη της Ρώμης με διαφορετική συνθήκη οκταώρου.

`

4.4. Αναλυτικά αποτελέσματα του αλγορίθμου Nearest Point για το φυσικό πρόβλημα.

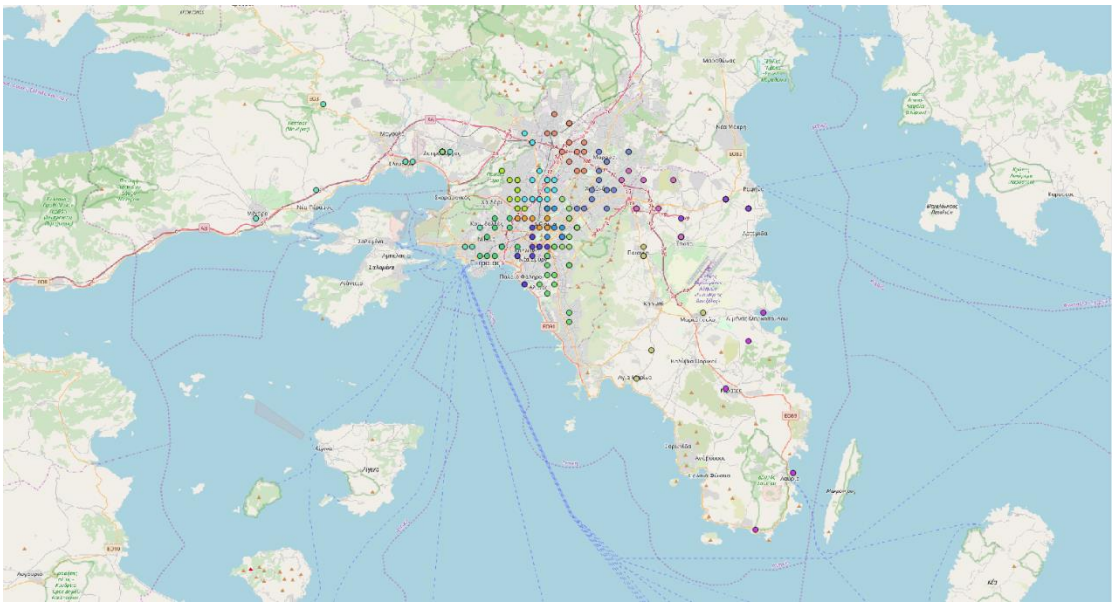
Παρακάτω παρουσιάζονται τα αποτελέσματα που προέκυψαν από την διαδικασία που περιεγράφηκε στο κεφάλαιο 4.1.2. για το φυσικό πρόβλημα. Στην συνέχεια του κεφαλαίου γίνεται επίλυση του φυσικού προβλήματος χρησιμοποιώντας το αναλυτικό μοντέλο TSP με χρονικά παράθυρα. Στον πίνακα 4.1 φαίνονται τα αποτελέσματα των ομάδων (αριθμός κόμβων κάθε ομάδας, ID πελατών, ID φορτηγών, κάλυψη χωρητικότητας).

Ομάδες πελατών	Αριθμός πελατών ανά ομάδα	ID πελατών ομάδας	ID Φορτηγού εξυπηρέτησης	Κάλυψη χωρητικότητας (α/A, β/B)
Ομάδα 1	12	42, 86, 89, 91, 96, 97, 102, 103, 111, 143, 160, 161	70	4008/4100 323/350
Ομάδα 2	6	45, 46, 82, 84, 93, 155	55	4053/7050 396/600
Ομάδα 3	12	56, 57, 64, 79, 121, 127, 162, 163, 164, 165, 166, 167	20	1527/23400 890/1300
Ομάδα 4	8	22, 78, 83, 104, 154, 158, 168, 169	28	7350/23400 1193/1300
Ομάδα 5	9	48, 77, 80, 106, 146, 149, 159, 170, 171	27	7756/23400 582/1300
Ομάδα 6	11	36, 39, 40, 60, 67, 75, 85, 87, 88, 137, 147	11	1548/23400 1178/1300
Ομάδα 7	14	11, 13, 41, 44, 55, 81, 122, 130, 142, 145, 148, 150, 156, 157	14	6243/7050 405/600
Ομάδα 8	17	12, 15, 28, 47, 54, 62, 68, 69, 70, 72, 76, 115, 132, 133, 139, 144, 151	53	5384/7050 512/600
Ομάδα 9	15	3, 6, 9, 10, 53, 61, 71, 105, 109, 119, 120, 129, 134, 135, 141	51	4734/7050 392/600
Ομάδα 10	14	4, 27, 29, 30, 49, 50, 51, 73, 98, 100, 101, 107, 123, 126	85	3704/3760 298/300
Ομάδα 11	11	14, 24, 66, 74, 90, 95, 125, 131, 140, 152, 153	26	8116/23400 930/1300
Ομάδα 12	12	23, 31, 32, 33, 34, 35, 37, 38, 65, 110, 128, 138	50	6600/8340 389/600
Ομάδα 13	9	1, 8, 26, 43, 58, 59, 94, 99, 108	49	5523/8340 342/600
Ομάδα 14	9	5, 7, 20, 21, 52, 92, 112, 114, 117	64	4337/4424 304/350
Ομάδα 15	10	2, 16, 17, 18, 19, 113, 116, 118, 124, 136	47	3679/9550 601/700
Ομάδα 16	2	25, 63	58	4225/4424 286/350

Πίνακας 4. 5 Αποτελέσματα ομαδοποίησης από αλγόριθμο Nearest Point.

`

Η λύση αυτή φαίνεται στο σχήμα 4.2. Η κάθε ομάδα που έχει δημιουργηθεί συμβολίζεται με διαφορετικό χρώμα στο σχήμα.



Σχήμα 4. 2 Οι θέσεις των πελατών στη περιοχή της Αττικής σύμφωνα με την ομαδοποίηση

Στη συνέχεια, λύνουμε τα προβλήματα TSP με χρονικά παράθυρα που δημιουργήθηκαν. Η επίλυση γίνεται με βάση το αναλυτικό μοντέλο που περιεγράφηκε στο κεφάλαιο 4.1.2. Τα τελικά αποτελέσματα φαίνονται στον πίνακα 2 του παραρτήματος.

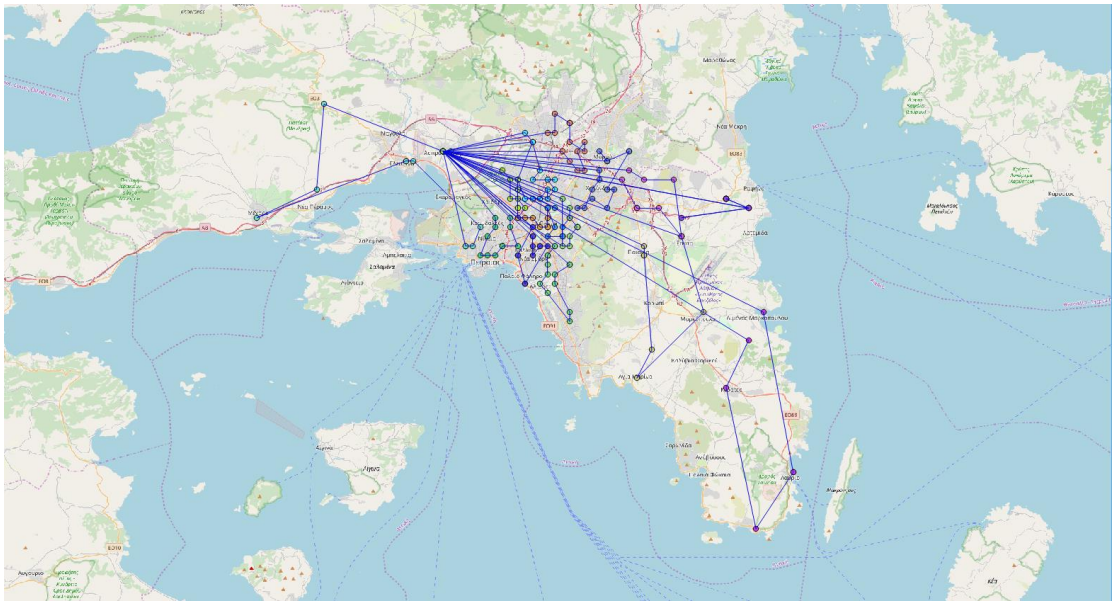
Στον πίνακα 2 του παραρτήματος φαίνονται οι μεταβλητές απόφασης που παίρνουν τιμή διάφορη του 0 καθώς και η απόσταση που θα διανύσει το κάθε όχημα που έχει ανατεθεί στην κάθε ομάδα ξεχωριστά.

- Στην πρώτη στήλη φαίνεται το ID της κάθε ομάδας.
- Στην δεύτερη στήλη έχουμε την μεταβλητή απόφασης Xijk. Όταν παίρνει τιμή ίση με 1 δείχνει ότι το όχημα k πάει από τον πελάτη i στον πελάτη j. Για την κάθε ομάδα έχουμε ταξινομήσει τις μεταβλητές Xijk με την σειρά που θα εξυπηρετηθούν οι πελάτες.
- Στην Τρίτη στήλη φαίνεται η μεταβλητή tik που δείχνει τον χρόνο αρχής εξυπηρέτησης του πελάτη i από το όχημα k. Η τιμή που παίρνει το k είναι ίση με το ID της ομάδας που επιλύεται την κάθε φορά. Τα οχήματα που εξυπηρετούν τις ομάδες φαίνονται στον πίνακα 4.4.
- Στην τέταρτη στήλη φαίνονται οι τιμές της μεταβλητής Yk. Αυτή η μεταβλητή δείχνει την ώρα που επιστρέφει το όχημα που εξυπηρετεί την κάθε ομάδα στο depot.
- Στην τελευταία στήλη φαίνονται οι αποστάσεις που διανύουν τα οχήματά μας σε χιλιόμετρα.

Η λύση αυτή φαίνεται και σχηματικά στο σχήμα 4.3.

Σχήμα 4. 3 Τελική λύση του φυσικού προβλήματος.

`



4.5. Συμπεράσματα του αλγορίθμου Nearest Point.

Το ζητούμενο αυτής της εργασίας ήταν να λύσουμε το συγκεκριμένο πρόβλημα δρομολόγησης οχημάτων μέσα σε μικρό χρονικό διάστημα και να πάρουμε μια ικανοποιητική λύση. Σύμφωνα με τα αποτελέσματα και οι δύο στόχοι έχουν επιτευχθεί αν και προφανώς δεν έχουμε πετύχει τη βελτιστότητα. Από τα αποτελέσματα μπορούμε επίσης να συμπεράνουμε τα παρακάτω για τον αλγόριθμο:

1. Όσο μεγαλύτερο είναι το χωρίο που είναι διεσπαρμένοι οι πελάτες, τόσο περισσότερες ομάδες έχουμε. Αυτό συμβαίνει γιατί οι χρόνοι μεταξύ των πελατών μεγαλώνουν με αποτέλεσμα να πληρώνεται το οκτάωρο της κάθε διαδρομής με λιγότερους πελάτες.
2. Όσο περισσότερη ζήτηση υπάρχει από τους πελάτες, τόσο περισσότερες διαδρομές έχουμε. Η μεγάλη ζήτηση των πελατών πληρώνει σε μεγαλύτερο βαθμό την διαθέσιμη χωρητικότητα και έχει ως επόμενο την δημιουργία μικρότερων διαδρομών.
3. Ο χρόνος επίλυσης του κάθε προβλήματος είναι πολύ μικρός.
4. Όσο μεγαλύτερες διαδρομές δημιουργούνται τόσο περισσότερος είναι ο χρόνος επίλυσης του προβλήματος. Σε περίπτωση που δημιουργηθούν πολύ μεγάλες διαδρομές, μπορεί να προκύψουν μη εφικτά προβλήματα. Για να λυθούν αυτά μπορούμε να συσφίξουμε τους περιορισμούς κατά τη δημιουργία των διαδρομών. Ο υπολογιστικός φόρτος και ο χρόνος επίλυσης των προβλημάτων είναι πολύ μικρός και μας επιτρέπει να κάνουμε αλλαγές ώστε να πάρουμε 100% λυμένα προβλήματα.
5. Με τον αλγόριθμο Nearest point έχουμε πιο πυκνές διαδρομές με λιγότερο χαμένο χρόνο στις διαδρομές μας.

`

Κεφάλαιο 5: Συμπεράσματα

5.1: Γενικά συμπεράσματα.

Στα πλαίσια αυτής της εργασίας έγινε μια προσπάθεια επίλυσης ενός αρκετά δύσκολου προβλήματος επιχειρησιακής έρευνας, το πρόβλημα δρομολόγησης οχημάτων με περιορισμένες χωρητικότητες και χρονικά παράθυρα. Λόγω της δυσκολίας ενός τέτοιου προβλήματος, για την λύση του, γίνονται αρκετές παραδοχές για την προσέγγιση της βελτιστότητας.

Στην δικιά μας περίπτωση, υπάρχουν αρκετοί παράγοντες που αυξάνουν την δυσκολία προβλήματος. Αρχικά, πέρα από τις δύο διαστάσεις που έχουμε για την περιγραφή του χώρου, προστίθεται και η διάσταση του χρόνου μέσα από τα χρονικά παράθυρα εξυπηρέτησης του κάθε πελάτη αλλά και με την μέγιστη ώρα εργασίας των μεταφορέων. Επιπλέον, έχουμε δύο είδη ζήτησης από τον κάθε πελάτη άρα και δύο είδη χωρητικότητας στο κάθε όχημα μεταφοράς. Αυτό μας αναγκάζει να εισάγουμε κάποιες επιπλέον μεταβλητές απόφασης αλλά και περιορισμούς, που αυξάνουν την πολυπλοκότητα του προβλήματος, άρα και τον χρόνο επίλυσής του.

Η πρώτη λύση του φυσικού προβλήματος έγινε με βάση το αναλυτικό μοντέλο επίλυσης που κατασκευάστηκε. Δεν υπήρξε κάποιο συγκεκριμένο είδος διαχωρισμού των δεδομένων μας. Αυτή η λύση θα μπορούσε να χαρακτηριστεί και ως πρώτη προσέγγιση στην επίλυση του προβλήματός. Η τιμή της αντικειμενικής συνάρτησης της αρχικής λύσης είναι 2.594,410 χιλιόμετρα. Από αυτή τη λύση φαίνεται ότι χρειαζόμαστε έναν πιο μεθοδικό τρόπο για ομαδοποίηση των δεδομένων μας, με ευρετικές ή μετευρετικές διαδικασίες.

Η δεύτερη λύση, λοιπόν, αποτελεί βελτίωση της πρώτης αφού έχουν χρησιμοποιηθεί τέτοιες μέθοδοι. Πλέον, η εισαγωγή πελατών σε ομάδες γίνεται με έναν μεθοδευμένο τρόπο. Αυτός ο τρόπος, όχι μόνο μας εξασφαλίζει καλύτερα αποτελέσματα, αλλά μειώνει σε πολύ μεγάλο βαθμό το φόρτο του προβλήματός μας. Έτσι, έχουμε λύση, σε χρόνο μικρότερο του λεπτού, που προσεγγίζει τη βέλτιστη. Η λύση αυτή είναι 1.259,940 χιλιόμετρα, 51,44% μικρότερη της αρχικής.

Με βάση αυτών των αποτελεσμάτων μπορούμε να συμπεράνουμε ότι οι μέθοδοι της επιχειρησιακής έρευνας αποφέρουν μεγάλο όφελος στην επίλυση τέτοιων προβλημάτων. Κατά την διαδικασία επιλογής συγκεκριμένων μεθόδων είναι απαραίτητος ο προσδιορισμός των παραμέτρων του προβλήματος που θέλουμε να ληφθούν υπόψιν. Επίσης, βασικό κριτήριο της επιλογής αυτών είναι ο βαθμός που θέλουμε να προσεγγίσουμε τη βελτιστότητα καθώς και ο χρόνος επίλυσης που στοχεύουμε. Στην δικιά μας περίπτωση, θέλαμε να προσεγγίσουμε την βελτιστότητα σε πολύ μικρό χρονικό διάστημα ώστε να είναι άμεσα διαθέσιμα τα αποτελέσματα σε επιχειρησιακό επίπεδο, πράγμα που πετύχαμε.

5.2: Σκέψεις για μελλοντικές βελτιώσεις.

Με την σημασία του κλάδου της επιχειρησιακής έρευνας να γίνεται καθημερινά όλο και πιο αναγκαίος και σε συνδυασμό με την ραγδαία τεχνολογική πρόοδο στον τομέα της πληροφορικής, τα δεδομένα στα προβλήματα βελτιστοποίησης αλλάζουν συνεχώς. Προβλήματα που απαιτούσαν ώρες για να λυθούν, παίρνουν απάντηση σε μερικά λεπτά μέσα από κορυφαίους solvers που έχουν δημιουργηθεί. Επιπλέον, επιστήμες όπως τα "νευρωνικά δίκτυα" και το "machine learning" δίνουν νέα πνοή στην επίλυση τέτοιων προβλημάτων αφαιρώντας πολλά από τα όρια που είχαν άλλες επιστήμες στο παρελθόν, δείχνοντας ότι στο μέλλον τέτοιες διαδικασίες θα είναι πολύ πιο εύκολες.

`

Βιβλιογραφία

Anily, S. (1996), The vehicle routing problem with delivery and back-haul options, Naval Research Logistics, Vol. 43, pp. 415-434.

Baker, E., Schaffer, J., 1989. Solution improvement heuristics for the vehicle routing and scheduling problem with time windows constraints. American Journal of Mathematics and Management Sciences 6 (3,4), 261–300.

Clarke, G., Wright, W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568– 581.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F. (2002), VRP with time windows, in: Toth, P and Vigo, D. (eds.), The vehicle routing problem, SIAM Monographs on discrete mathematics and applications, Vol. 9, pp. 157-193, Philadelphia, PA .

Crainic, T.G., Laporte, G., (1998), Fleet management and logistics, Kluwer Academic, London, Dordrecht, Boston.

Dantzig, G.B. and Ramser, J.M. (1959), The truck dispatching problem, Management Science, Vol. 6, pp. 81-91.

C.E. Miller, A.W. Tucker, R.A. Zemlin (1960), Integer programming formulations and traveling salesman problems, J. ACM, 7, pp. 326-329.

Imran, A., Slahi, S., Wassan, N.A. (2009), A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem, European Journal of Operational Research, Vol. 197 No. 2, pp. 509-518.

Kolen, A.W.J., Rinnooy Kan, A.H.G., Trienkens, H.W.J.M., (1987), Vehicle routing with time windows, Operations Research, Vol. 35 No. 2, pp. 266-273.

Koskosidis, Y., Powell, W., Solomon, M., 1992. An optimizationbased heuristic for vehicle routing and scheduling with soft time windows constraints. Transportation Science 26, 57–69.

Laporte, G., (1992), The vehicle routing problem: An overview of exact and approximate algorithms, European Journal of Operational Research, Vol. 59 No. 3, pp. 345-358.

Lu, Q., Dessouky, M.M. (2006), A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows, European Journal of Operational Research, Vol. 175, No.2, pp. 672-687.

Savelsbergh, M.W.P. and Sol, M. (1995), The general pickup and delivery problem, Transportation Science, Vol.29, pp. 17-29.

Solomon, M. (1995), Algorithms for the vehicle routing problem with time windows, Transportation Science, Vol. 29 No.2, pp. 156-166.

Yaohuang, G., Binglei, X., Qiang, G. (2002), Overview of stochastic vehicle routing problems, Journal of Southwest Jiaotong university (English Edition), Vol. 10 No.2 , pp. 113-121.

`

Παράρτημα

Πίνακας 1. Αποτελέσματα λύσης των υπο-προβλημάτων που μελετήθηκαν.

Ομάδα πελατών	Μεταβλητή Xijk(i,j,α)	Μεταβλητή tik(i,α)	Μεταβλητή Yk(α)
Ομάδα 1	Xijk(0,2,3)=1 Xijk(2,6,3)=1 Xijk(3,9,3)=1 Xijk(6,3,3)=1 Xijk(9,0,3)=1 Xijk(0,1,5)=1 Xijk(1,4,5)=1 Xijk(4,5,5)=1 Xijk(5,7,5)=1 Xijk(7,8,5)=1 Xijk(8,0,5)=1	tik(0 , 3)=43 tik(2 , 3)=480 tik(3 , 3)=523 tik(6 , 3)=509 tik(9 , 3)=598 tik(0 , 5)=496 tik(1 , 5)=521 tik(4 , 5)=607 tik(5 , 5)=808 tik(7 , 5)=828 tik(8 , 5)=911	Yk(3)=523 Yk(5)=976
Ομάδα 2	Xijk(0,14,6)=1 Xijk(14,18,6)=1 Xijk(15,0,6)=1 Xijk(16,15,6)=1 Xijk(18,16,6)=1 Xijk(0,17,8)=1 Xijk(17,0,8)=1 Xijk(0,10,10)=1 Xijk(10,11,10)=1 Xijk(11,13,10)=1 Xijk(12,0,10)=1	tik(0 , 6)=210 tik(14 , 6)=570 tik(15 , 6)=754 tik(16 , 6)=690 tik(18 , 6)=636 tik(0 , 8)=210 tik(17 , 8)=480 tik(0 , 10)=109 tik(10 , 10)=438 tik(11 , 10)=480 tik(12 , 10)=613 tik(13 , 10)=501	Yk(6)=690 Yk(8)=480 Yk(10)=589

`

	Xijk(13,12,10)=1		
Ομάδα 3	Xijk(0,26,12)=1 Xijk(24,25,12)=1 Xijk(25,0,12)=1 Xijk(26,24,12)=1 Xijk(0,22,14)=1 Xijk(19,21,14)=1 Xijk(20,0,14)=1 Xijk(21,20,14)=1 Xijk(22,27,14)=1 Xijk(23,19,14)=1 Xijk(27,23,14)=1	tik(0 , 12)=0 tik(24 , 12)=33 tik(25 , 12)=74 tik(26 , 12)=20 tik(0 , 14)=606 tik(19 , 14)=852 tik(20 , 14)=918 tik(21 , 14)=887 tik(22 , 14)=667 tik(23 , 14)=822 tik(27 , 14)=781	Yk(12)=33 Yk(14)=887
Ομάδα 4	Xijk(0,32,16)=1 Xijk(28,0,16)=1 Xijk(29,30,16)=1 Xijk(30,28,16)=1 Xijk(31,29,16)=1 Xijk(32,34,16)=1 Xijk(33,36,16)=1 Xijk(34,33,16)=1 Xijk(35,31,16)=1 Xijk(36,35,16)=1	tik(0 , 16)=310 tik(28 , 16)=827 tik(29 , 16)=770 tik(30 , 16)=789 tik(31 , 16)=747 tik(32 , 16)=360 tik(33 , 16)=632 tik(34 , 16)=564 tik(35 , 16)=732 tik(36 , 16)=688	Yk(16)=790
Ομάδα 5	Xijk(0,44,24)=1 Xijk(37,42,24)=1 Xijk(38,37,24)=1 Xijk(39,0,24)=1 Xijk(40,39,24)=1 Xijk(41,45,24)=1 Xijk(42,40,24)=1 Xijk(43,38,24)=1	tik(0 , 24)=363 tik(37 , 24)=750 tik(38 , 24)=723 tik(39 , 24)=924 tik(40 , 24)=834 tik(41 , 24)=501 tik(42 , 24)=807 tik(43 , 24)=662	Yk(24)=843

`

	Xijk(44,41,24)=1 Xijk(45,43,24)=1	tik(44 , 24)=480 tik(45 , 24)=585	
Ομάδα 6	Xijk(0,54,26)=1 Xijk(54,0,26)=1 Xijk(0,47,29)=1 Xijk(46,53,29)=1 Xijk(47,52,29)=1 Xijk(48,46,29)=1 Xijk(49,51,29)=1 Xijk(50,49,29)=1 Xijk(51,48,29)=1 Xijk(52,50,29)=1 Xijk(53,0,29)=1	tik(0 , 26)=282 tik(54 , 26)=300 tik(0 , 29)=461 tik(46 , 29)=812 tik(47 , 29)=480 tik(48 , 29)=707 tik(49 , 29)=635 tik(50 , 29)=585 tik(51 , 29)=660 tik(52 , 29)=516 tik(53 , 29)=981	Yk(26)=250 Yk(29)=941
Ομάδα 7	Xijk(0,62,33)=1 Xijk(55,0,33)=1 Xijk(56,61,33)=1 Xijk(57,56,33)=1 Xijk(58,59,33)=1 Xijk(59,63,33)=1 Xijk(60,58,33)=1 Xijk(61,55,33)=1 Xijk(62,60,33)=1 Xijk(63,57,33)=1	tik(0 , 33)=17 tik(55 , 33)=535 tik(56 , 33)=467 tik(57 , 33)=446 tik(58 , 33)=345 tik(59 , 33)=372 tik(60 , 33)=322 tik(61 , 33)=497 tik(62 , 33)=300 tik(63 , 33)=404	Yk(33)=497
Ομάδα 8	Xijk(0,66,37)=1 Xijk(64,71,37)=1 Xijk(65,72,37)=1 Xijk(66,67,37)=1 Xijk(67,65,37)=1 Xijk(68,0,37)=1 Xijk(69,70,37)=1	tik(0 , 37)=79 tik(64 , 37)=501 tik(65 , 37)=374 tik(66 , 37)=300 tik(67 , 37)=338 tik(68 , 37)=586 tik(69 , 37)=430	Yk(37)=559

`

	Xijk(70,64,37)=1 Xijk(71,68,37)=1 Xijk(72,69,37)=1	tik(70 , 37)=457 tik(71 , 37)=549 tik(72 , 37)=413	
Ομάδα 9	Xijk(0,76,44)=1 Xijk(73,78,44)=1 Xijk(74,75,44)=1 Xijk(75,73,44)=1 Xijk(76,74,44)=1 Xijk(77,81,44)=1 Xijk(78,79,44)=1 Xijk(79,80,44)=1 Xijk(80,77,44)=1 Xijk(81,0,44)=1	tik(0 , 44)=69 tik(73 , 44)=382 tik(74 , 44)=318 tik(75 , 44)=343 tik(76 , 44)=300 tik(77 , 44)=549 tik(78 , 44)=447 tik(79 , 44)=490 tik(80 , 44)=527 tik(81 , 44)=579	Yk(44)=549
Ομάδα 10	Xijk(0,90,46)=1 Xijk(82,0,46)=1 Xijk(83,84,46)=1 Xijk(84,82,46)=1 Xijk(85,83,46)=1 Xijk(86,88,46)=1 Xijk(87,85,46)=1 Xijk(88,87,46)=1 Xijk(89,86,46)=1 Xijk(90,89,46)=1	tik(0 , 46)=274 tik(82 , 46)=883 tik(83 , 46)=687 tik(84 , 46)=754 tik(85 , 46)=621 tik(86 , 46)=540 tik(87 , 46)=595 tik(88 , 46)=571 tik(89 , 46)=519 tik(90 , 46)=480	Yk(46)=754
Ομάδα 11	Xijk(0,93,54)=1 Xijk(91,0,54)=1 Xijk(92,99,54)=1 Xijk(93,98,54)=1 Xijk(94,97,54)=1 Xijk(95,94,54)=1 Xijk(96,91,54)=1 Xijk(97,96,54)=1	tik(0 , 54)=328 tik(91 , 54)=853 tik(92 , 54)=646 tik(93 , 54)=480 tik(94 , 54)=719 tik(95 , 54)=696 tik(96 , 54)=808 tik(97 , 54)=772	Yk(54)=808

`

	Xijk(98,92,54)=1 Xijk(99,95,54)=1 Xijk(0,93,54)=1 Xijk(91,0,54)=1	tik(98 , 54)=600 tik(99 , 54)=674 tik(0 , 54)=328 tik(91 , 54)=853	
Ομάδα 12	Xijk(0,102,56)=1 Xijk(100,101,56)=1 Xijk(101,105,56)=1 Xijk(102,103,56)=1 Xijk(103,108,56)=1 Xijk(104,106,56)=1 Xijk(105,104,56)=1 Xijk(106,0,56)=1 Xijk(107,100,56)=1 Xijk(108,107,56)=1	tik(0 , 56)=271 tik(100 , 56)=646 tik(101 , 56)=665 tik(102 , 56)=480 tik(103 , 56)=529 tik(104 , 56)=751 tik(105 , 56)=712 tik(106 , 56)=781 tik(107 , 56)=624 tik(108 , 56)=587	Yk(56)=751
Ομάδα 13	Xijk(0,111,63)=1 Xijk(109,0,63)=1 Xijk(110,114,63)=1 Xijk(111,112,63)=1 Xijk(112,113,63)=1 Xijk(113,116,63)=1 Xijk(114,115,63)=1 Xijk(115,109,63)=1 Xijk(116,117,63)=1 Xijk(117,110,63)=1	tik(0 , 63)=117 tik(109 , 63)=633 tik(110 , 63)=544 tik(111 , 63)=300 tik(112 , 63)=455 tik(113 , 63)=480 tik(114 , 63)=576 tik(115 , 63)=597 tik(116 , 63)=503 tik(117 , 63)=525	Yk(63)=597
Ομάδα 14	Xijk(0,125,68)=1 Xijk(118,119,68)=1 Xijk(119,121,68)=1 Xijk(120,122,68)=1 Xijk(121,120,68)=1 Xijk(122,0,68)=1	tik(0 , 68)=282 tik(118 , 68)=606 tik(119 , 68)=639 tik(120 , 68)=761 tik(121 , 68)=702 tik(122 , 68)=793	Yk(68)=762

`

	Xijk(123,126,68)=1 Xijk(124,118,68)=1 Xijk(125,123,68)=1 Xijk(126,124,68)=1	tik(123 , 68)=533 tik(124 , 68)=576 tik(125 , 68)=480 tik(126 , 68)=554	
Ομάδα 15	Xijk(0,133,72)=1 Xijk(127,128,72)=1 Xijk(128,135,72)=1 Xijk(129,130,72)=1 Xijk(130,0,72)=1 Xijk(131,127,72)=1 Xijk(132,131,72)=1 Xijk(133,132,72)=1 Xijk(134,129,72)=1 Xijk(135,134,72)=1 Xijk(0,133,72)=1 Xijk(127,128,72)=1	tik(0 , 72)=453 tik(127 , 72)=661 tik(128 , 72)=694 tik(129 , 72)=802 tik(130 , 72)=825 tik(131 , 72)=614 tik(132 , 72)=583 tik(133 , 72)=558 tik(134 , 72)=778 tik(135 , 72)=737 tik(0 , 72)=453 tik(127 , 72)=661	Yk(72)=933
Ομάδα 16	Xijk(0,142,77)=1 Xijk(136,138,77)=1 Xijk(137,143,77)=1 Xijk(138,137,77)=1 Xijk(139,144,77)=1 Xijk(140,139,77)=1 Xijk(141,136,77)=1 Xijk(142,141,77)=1 Xijk(143,140,77)=1 Xijk(144,0,77)=1	tik(0 , 77)=448 tik(136 , 77)=630 tik(137 , 77)=705 tik(138 , 77)=675 tik(139 , 77)=885 tik(140 , 77)=787 tik(141 , 77)=596 tik(142 , 77)=564 tik(143 , 77)=731 tik(144 , 77)=928	Yk(77)=885
Ομάδα 17	Xijk(0,148,84)=1 Xijk(145,147,84)=1 Xijk(146,150,84)=1 Xijk(147,153,84)=1	tik(0 , 84)=307 tik(145 , 84)=617 tik(146 , 84)=567 tik(147 , 84)=657	Yk(84)=787

`

	Xijk(148,149,84)=1 Xijk(149,146,84)=1 Xijk(150,145,84)=1 Xijk(151,0,84)=1 Xijk(152,151,84)=1 Xijk(153,152,84)=1	tik(148 , 84)=480 tik(149 , 84)=531 tik(150 , 84)=599 tik(151 , 84)=813 tik(152 , 84)=783 tik(153 , 84)=703	
Ομάδα 18	Xijk(0,161,86)=1 Xijk(160,0,86)=1 Xijk(161,160,86)=1 Xijk(0,162,89)=1 Xijk(154,158,89)=1 Xijk(155,157,89)=1 Xijk(156,0,89)=1 Xijk(157,156,89)=1 Xijk(158,155,89)=1 Xijk(159,154,89)=1 Xijk(162,159,89)=1	Tik(0 , 86)=0 tik(160 , 86)=542 tik(161 , 86)=457 tik(0 , 89)=397 tik(154 , 89)=514 tik(155 , 89)=729 tik(156 , 89)=896 tik(157 , 89)=877 tik(158 , 89)=681 tik(159 , 89)=480 tik(162 , 89)=448	Yk(86)=480 Yk(89)=877
Ομάδα 19	Xijk(0,168,91)=1 Xijk(163,0,91)=1 Xijk(164,167,91)=1 Xijk(165,163,91)=1 Xijk(166,165,91)=1 Xijk(167,166,91)=1 Xijk(168,164,91)=1 Xijk(0,171,92)=1 Xijk(169,170,92)=1 Xijk(170,0,92)=1 Xijk(171,169,92)=1	tik(0 , 91)=0 tik(163 , 91)=109 tik(164 , 91)=69 tik(165 , 91)=99 tik(166 , 91)=89 tik(167 , 91)=79 tik(168 , 91)=33 tik(0 , 92)=0 tik(169 , 92)=43 tik(170 , 92)=86 tik(171 , 92)=33	Yk(91)=480 Yk(92)=480
Σύνολο	2.594.410		

`

Πίνακας 2. Τελικά αποτελέσματα αλγορίθμου Nearest Point για το φυσικό πρόβλημα.

Ομάδα πελατών	Μεταβλητή Xijk(i,j,α)	Μεταβλητή tik(i,α)	Μεταβλητή Yk(α)	Απόσταση(m)
Ομάδα 1	Xijk(0,160,1)=1 Xijk(160,97,1)=1 Xijk(97,103,1)=1 Xijk(103,91,1)=1 Xijk(91,111,1)=1 Xijk(111,96,1)=1 Xijk(96,89,1)=1 Xijk(89,143,1)=1 Xijk(143,42,1)=1 Xijk(42,86,1)=1 Xijk(86,102,1)=1 Xijk(102,161,1)=1 Xijk(161,0,1)=1	tik(0 , 1)=401 tik(160 , 1)=436 tik(97 , 1)=501 tik(103 , 1)=528 tik(91 , 1)=565 tik(111 , 1)=595 tik(96 , 1)=611 tik(89 , 1)=668 tik(143 , 1)=687 tik(42 , 1)=722 tik(86 , 1)=735 tik(102 , 1)=810 tik(161 , 1)=826	Yk(1)=881	116948
Ομάδα 2	Xijk(0,155,2)=1 Xijk(45,82,2)=1 Xijk(46,84,2)=1 Xijk(82,93,2)=1 Xijk(84,0,2)=1 Xijk(93,46,2)=1 Xijk(155,45,2)=1	tik(0 , 2)=435 tik(45 , 2)=609 tik(46 , 2)=742 tik(82 , 2)=684 tik(84 , 2)=832 tik(93 , 2)=727 tik(155 , 2)=480	Yk(2)=915	160123
Ομάδα 3	Xijk(0,127,3)=1 Xijk(166,163,3)=1 Xijk(127,79,3)=1 Xijk(79,121,3)=1 Xijk(121,64,3)=1 Xijk(64,164,3)=1 Xijk(164,165,3)=1 Xijk(165,167,3)=1 Xijk(167,162,3)=1	tik(0 , 3)=293 tik(166 , 3)=675 tik(127 , 3)=480 tik(79 , 3)=532 tik(121 , 3)=581 tik(64 , 3)=616 tik(164 , 3)=634 tik(165 , 3)=644 tik(167 , 3)=654	Yk(3)=773	128708

`

	Xijk(162,166,3)=1 Xijk(163,56,3)=1 Xijk(56,57,3)=1 Xijk(57,0,3)=1	tik(162 , 3)=664 tik(163 , 3)=685 tik(56 , 3)=714 tik(57 , 3)=727		
Ομάδα 4	Xijk(0,83,4)=1 Xijk(83,104,4)=1 Xijk(104,154,4)=1 Xijk(154,158,4)=1 Xijk(158,22,4)=1 Xijk(22,78,4)=1 Xijk(78,169,4)=1 Xijk(169,0,4)=1	tik(0 , 4)=435 tik(83 , 4)=480 tik(104 , 4)=503 tik(154 , 4)=518 tik(158 , 4)=685 tik(22 , 4)=720 tik(78 , 4)=834 tik(169 , 4)=872	Yk(4)=915	116849
Ομάδα 5	Xijk(0,146,5)=1 Xijk(146,9,5)=1 Xijk(9,77,5)=1 Xijk(77,80,5)=1 Xijk(80,149,5)=1 Xijk(149,168,5)=1 Xijk(168,171,5)=1 Xijk(171,106,5)=1 Xijk(106,159,5)=1 Xijk(159,48,5)=1 Xijk(48,119,5)=1 Xijk(119,105,5)=1 Xijk(105,141,5)=1 Xijk(141,71,5)=1 Xijk(71,0,5)=1	tik(0 , 5)=411 tik(146 , 5)=480 tik(9 , 5)=513 tik(77 , 5)=566 tik(80 , 5)=589 tik(149 , 5)=622 tik(168 , 5)=663 tik(171 , 5)=674 tik(106 , 5)=698 tik(159 , 5)=726 tik(48 , 5)=748 tik(119 , 5)=780 tik(105 , 5)=810 tik(141 , 5)=833 tik(71 , 5)=852	Yk(5)=891	87885
Ομάδα 6	Xijk(0,53,6)=1 Xijk(53,61,6)=1 Xijk(61,10,6)=1 Xijk(10,170,6)=1 Xijk(170,134,6)=1	tik(0 , 6)=93 tik(53 , 6)=300 tik(61 , 6)=368 tik(10 , 6)=387 tik(170 , 6)=440	Yk(6)=573	84576

`

	Xijk(134,120,6)=1 Xijk(120,109,6)=1 Xijk(109,129,6)=1 Xijk(129,0,6)=1	tik(134 , 6)=480 tik(120 , 6)=503 tik(109 , 6)=521 tik(129 , 6)=541		
Ομάδα 7	Xijk(0,60,7)=1 Xijk(60,85,7)=1 Xijk(85,36,7)=1 Xijk(36,40,7)=1 Xijk(40,39,7)=1 Xijk(39,137,7)=1 Xijk(137,147,7)=1 Xijk(147,67,7)=1 Xijk(67,88,7)=1 Xijk(88,87,7)=1 Xijk(87,75,7)=1 Xijk(75,0,7)=1	tik(0 , 7)=368 tik(60 , 7)=389 tik(85 , 7)=450 tik(36 , 7)=496 tik(40 , 7)=539 tik(39 , 7)=629 tik(137 , 7)=687 tik(147 , 7)=708 tik(67 , 7)=730 tik(88 , 7)=755 tik(87 , 7)=779 tik(75 , 7)=804	Yk(7)=848	69695
Ομάδα 8	Xijk(0,122,8)=1 Xijk(122,44,8)=1 Xijk(44,55,8)=1 Xijk(55,148,8)=1 Xijk(148,156,8)=1 Xijk(156,157,8)=1 Xijk(157,41,8)=1 Xijk(41,142,8)=1 Xijk(142,130,8)=1 Xijk(130,81,8)=1 Xijk(81,150,8)=1 Xijk(150,145,8)=1 Xijk(145,11,8)=1 Xijk(11,13,8)=1 Xijk(13,0,8)=1	tik(0 , 8)=405 tik(122 , 8)=480 tik(44 , 8)=497 tik(55 , 8)=515 tik(148 , 8)=537 tik(156 , 8)=556 tik(157 , 8)=582 tik(41 , 8)=609 tik(142 , 8)=645 tik(130 , 8)=663 tik(81 , 8)=683 tik(150 , 8)=706 tik(145 , 8)=724 tik(11 , 8)=745 tik(13 , 8)=766	Yk(8)=885	48043
Ομάδα 9	Xijk(0,68,9)=1	tik(0 , 9)=519	Yk(9)=999	48549

`

	Xijk(68,12,9)=1 Xijk(12,76,9)=1 Xijk(76,62,9)=1 Xijk(62,132,9)=1 Xijk(132,47,9)=1 Xijk(47,133,9)=1 Xijk(133,28,9)=1 Xijk(28,115,9)=1 Xijk(115,72,9)=1 Xijk(72,69,9)=1 Xijk(69,15,9)=1 Xijk(15,70,9)=1 Xijk(70,144,9)=1 Xijk(144,139,9)=1 Xijk(139,54,9)=1 Xijk(54,151,9)=1 Xijk(151,0,9)=1	tik(68 , 9)=533 tik(12 , 9)=552 tik(76 , 9)=585 tik(62 , 9)=603 tik(132 , 9)=619 tik(47 , 9)=646 tik(133 , 9)=666 tik(28 , 9)=689 tik(115 , 9)=711 tik(72 , 9)=729 tik(69 , 9)=746 tik(15 , 9)=769 tik(70 , 9)=820 tik(144 , 9)=837 tik(139 , 9)=870 tik(54 , 9)=889 tik(151 , 9)=932		
Ομάδα 10	Xijk(0,126,10)=1 Xijk(126,123,10)=1 Xijk(123,107,10)=1 Xijk(107,101,10)=1 Xijk(101,27,10)=1 Xijk(27,98,10)=1 Xijk(98,51,10)=1 Xijk(51,100,10)=1 Xijk(100,49,10)=1 Xijk(49,50,10)=1 Xijk(50,73,10)=1 Xijk(73,29,10)=1 Xijk(29,30,10)=1 Xijk(30,4,10)=1 Xijk(4,0,10)=1	tik(0 , 10)=361 tik(126 , 10)=480 tik(123 , 10)=497 tik(107 , 10)=525 tik(101 , 10)=544 tik(27 , 10)=571 tik(98 , 10)=590 tik(51 , 10)=625 tik(100 , 10)=643 tik(49 , 10)=660 tik(50 , 10)=685 tik(73 , 10)=734 tik(29 , 10)=751 tik(30 , 10)=770 tik(4 , 10)=789	Yk(10)=841	87543

`

Ομάδα 11	Xijk(0,152,11)=1 Xijk(152,90,11)=1 Xijk(90,74,11)=1 Xijk(74,131,11)=1 Xijk(131,140,11)=1 Xijk(140,24,11)=1 Xijk(24,14,11)=1 Xijk(14,95,11)=1 Xijk(95,66,11)=1 Xijk(66,125,11)=1 Xijk(125,153,11)=1 Xijk(153,0,11)=1	tik(0 , 11)=429 tik(152 , 11)=480 tik(90 , 11)=497 tik(74 , 11)=521 tik(131 , 11)=537 tik(140 , 11)=571 tik(24 , 11)=662 tik(14 , 11)=706 tik(95 , 11)=746 tik(66 , 11)=768 tik(125 , 11)=784 tik(153 , 11)=808	Yk(11)=909	45863
Ομάδα 12	Xijk(0,32,12)=1 Xijk(32,33,12)=1 Xijk(33,35,12)=1 Xijk(35,128,12)=1 Xijk(128,31,12)=1 Xijk(31,110,12)=1 Xijk(110,23,12)=1 Xijk(23,138,12)=1 Xijk(138,65,12)=1 Xijk(65,37,12)=1 Xijk(37,38,12)=1 Xijk(38,34,12)=1 Xijk(34,0,12)=1	tik(0 , 12)=378 tik(32 , 12)=407 tik(33 , 12)=480 tik(35 , 12)=532 tik(128 , 12)=546 tik(31 , 12)=563 tik(110 , 12)=586 tik(23 , 12)=604 tik(138 , 12)=636 tik(65 , 12)=655 tik(37 , 12)=686 tik(38 , 12)=725 tik(34 , 12)=765	Yk(12)=858	72076
Ομάδα 13	Xijk(0,26,13)=1 Xijk(26,58,13)=1 Xijk(58,99,13)=1 Xijk(59,108,13)=1 Xijk(99,1,13)=1 Xijk(1,43,13)=1 Xijk(43,8,13)=1	tik(0 , 13)=259 tik(26 , 13)=279 tik(58 , 13)=300 tik(59 , 13)=667 tik(99 , 13)=480 tik(1 , 13)=497 tik(43 , 13)=573	Yk(13)=739	55750

`

	Xijk(8,59,13)=1 Xijk(108,94,13)=1 Xijk(94,0,13)=1	tik(8 , 13)=622 tik(108 , 13)=686 tik(94 , 13)=702		
Ομάδα 14	Xijk(0,112,14)=1 Xijk(112,52,14)=1 Xijk(52,5,14)=1 Xijk(5,20,14)=1 Xijk(20,21,14)=1 Xijk(21,92,14)=1 Xijk(92,117,14)=1 Xijk(117,7,14)=1 Xijk(7,114,14)=1 Xijk(114,0,14)=1	tik(0 , 14)=217 tik(112 , 14)=300 tik(52 , 14)=321 tik(5 , 14)=480 tik(20 , 14)=499 tik(21 , 14)=547 tik(92 , 14)=581 tik(117 , 14)=603 tik(7 , 14)=632 tik(114 , 14)=660	Yk(14)=697	52208
Ομάδα 15	Xijk(0,135,15)=1 Xijk(135,3,15)=1 Xijk(3,6,15)=1 Xijk(6,18,15)=1 Xijk(18,17,15)=1 Xijk(17,113,15)=1 Xijk(113,19,15)=1 Xijk(19,118,15)=1 Xijk(118,136,15)=1 Xijk(136,2,15)=1 Xijk(2,16,15)=1 Xijk(16,0,15)=1	tik(0 , 15)=317 tik(135 , 15)=495 tik(3 , 15)=518 tik(6 , 15)=542 tik(18 , 15)=562 tik(17 , 15)=602 tik(113 , 15)=618 tik(19 , 15)=640 tik(118 , 15)=681 tik(136 , 15)=702 tik(2 , 15)=718 tik(16 , 15)=750	Yk(15)=797	59417
Ομάδα 16	Xijk(0,25,16)=1 Xijk(25,124,16)=1 Xijk(124,116,16)=1 Xijk(116,63,16)=1 Xijk(63,0,16)=1	tik(0 , 16)=89 tik(25 , 16)=149 tik(124 , 16)=480 tik(116 , 16)=500 tik(63 , 16)=514	Yk(16)=569	59569
Σύνολο:				1290800

`

Κώδικας για αρχικές λύσεις

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


int i, j, k, h, b, p;
int c = 0; //pelates pou exeis eksipiretisei
int d = 0; //fortiga pou exeis xrisimopoiisei
const int imax = 10;
const int jmax = imax;
const int hmax = imax;
const int kmax = 5;
const int pel = 172;
const int fort = 105;
const int M = 1000000;
float totald = 0; //total distance
float totalf = 0; //total fortigwn
float maxd = 0; //max distance
float mind = 1000000; //min distance
float dap = 0;
float dbp = 0;
float xorap = 0; //pososto kalipsis a
float xorbp = 0; //pososto kalipsis b
float xorapg = 0; //pososto kalipsis a geniko
float xorbpg = 0; //pososto kalipsis b geniko
float minp = 500; //elaxistoi pelates se diadromi
float maxp = 0; //megistoi pelates se diadromi

float D1[imax];
float D2[imax];
float MD[imax];
```

`

```
float TWS[imax];
float TWF[imax];
float Q1[kmax];
float Q2[kmax];
float Dij[imax][imax];
float Tij[imax][jmax];
float T2ij[imax][jmax];


int main() {
	ifstream inFile;
	ofstream outFile;
	outFile.open("lisi.txt");

	p = 0;

	do {
		/*Stoixeia Demand- Delay- Time Windows */
		inFile.open("Pelates.txt");
		if (inFile.fail())
		{
			cout << "input file could not be opened" << endl;
			system("pause");
			exit(1);
		}

		for (i = 0; i < imax; i++) {
			D1[i] = 0;
			D2[i] = 0;
			MD[i] = 0;
			TWS[i] = 0;
			TWF[i] = 0;
```

```
`
                }


                for (i = 0; i < pel; i++) {
                        if (i == 0) {
                                inFile >> D1[i] >> D2[i] >> MD[i] >> TWS[i] >> TWF[i];
                        }
                        else if (i > 0 + c && i < imax + c) {
                                inFile >> D1[i - c] >> D2[i - c] >> MD[i - c] >> TWS[i - c]
>> TWF[i - c];
                                //cout << D1[i] << "     " << D2[i] << "     " << MD[i] << "     "
<< TWS[i] << "     " << TWF[i] << "     " << endl;
                        }
                        else {
                                inFile >> b >> b >> b >> b >> b;
                        }
                }


                for (i = 0; i < imax; i++) {
                        cout << D1[i] << "      " << D2[i] << "      " << MD[i] << "      " <<
TWS[i] << "     " << TWF[i] << "     " << endl;
                }


                inFile.close();


                /*Stoixeia Xwritikotitas */
                inFile.open("Fortiga.txt");
                if (inFile.fail())
                {
                        cout << "input file could not be opened" << endl;
                        system("pause");
                        exit(1);
                }
```

`

```
for (k = 0; k < kmax; k++) {
	Q1[k] = 0;
	Q2[k] = 0;
}

for (k = 0; k < fort; k++) {
	if (k < kmax + d && k >= 0 + d) {
		inFile >> Q1[k - d] >> Q2[k - d];
	}
	else {
		inFile >> b >> b;
	}
}

for (k = 0; k < kmax; k++) {
	cout << Q1[k] << "     " << Q2[k] << endl;
}

inFile.close();

/*Stoixeia apostasis pelatwn*/
inFile.open("Dij.txt");
if (inFile.fail())
{
	cout << "input file could not be opened" << endl;
	system("pause");
	exit(1);
}

for (i = 0; i < imax; i++) {
	for (j = 0; j < jmax; j++) {
```

`

```
                Dij[i][j] = 0;
        }
}

for (i = 0; i < pel; i++) {
        for (j = 0; j < pel; j++) {
                if (j < jmax + c && j > 0 + c && i < imax + c && i > 0 + c) {
                        inFile >> Dij[i - c][j - c];
                }
                else if (j == 0 && i < imax + c && i > 0 + c) {
                        inFile >> Dij[i - c][j];
                }
                else if (i == 0 && j < jmax + c && j > 0 + c) {
                        inFile >> Dij[i][j - c];
                }
                else {
                        inFile >> b;
                }
        }
}

for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                cout << Dij[i][j] << "    ";
        }
        cout << endl;
}

inFile.close();

/*Stoixeia xronou metaksy pelatwn*/
inFile.open("Tij.txt");
```

`

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

for (i = 0; i < imax; i++) {
	for (j = 1; j < jmax; j++) {
		Tij[i][j] = 0;
	}
}

for (i = 0; i < pel; i++) {
	for (j = 0; j < pel; j++) {
		if (j < jmax + c && j > 0 + c && i < imax + c && i > 0 + c) {
			inFile >> Tij[i - c][j - c];
			Tij[i - c][j - c] = Tij[i - c][j - c] / 60;
		}
		else if (j == 0 && i < imax + c && i > 0 + c) {
			inFile >> Tij[i - c][j];
			Tij[i - c][j] = Tij[i - c][j] / 60;
		}
		else if (i == 0 && j < jmax + c && j > 0 + c) {
			inFile >> Tij[i][j - c];
			Tij[i][j - c] = Tij[i][j - c] / 60;
		}
		else {
			inFile >> b;
		}
	}
}
```

`

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


inFile.close();

/*telos data*/

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);
```

`

```
//Metavlites

//Xijk
IloNumVarMatrix3x3 Xijk(env, 0);
for (i = 0; i < imax; i++) {
	IloNumVarMatrix2x2 Xjk(env, 0);
	for (j = 0; j < jmax; j++) {
		IloNumVarArray Xk(env, 0);
		for (k = 0; k < kmax; k++) {
			char metavlitiX[70];
			sprintf(metavlitiX,  "Xijk(i%d,j%d,k%d)",  i,
j, k);
			IloNumVar X(env, 0, 1, ILOINT,
metavlitiX);
			Xk.add(X);
		}
		Xjk.add(Xk);
	}
	Xijk.add(Xjk);
}

//tik
IloNumVarMatrix2x2 tik(env, 0);
for (i = 0; i < imax; i++) {
	IloNumVarArray tk(env, 0);
	for (k = 0; k < kmax; k++) {
		char metavlitit[70];
		sprintf(metavlitit, "tik(i%d,k%d)", i, k);
		IloNumVar t(env, 0, 1440, ILOFLOAT, metavlitit);
		tk.add(t);
	}
	tik.add(tk);
```

`

```
}

//Uj
IloNumVarArray Uj(env, 0);
for (j = 0; j < jmax; j++) {
	char metavlitiU[70];
	sprintf(metavlitiU, "Ui(j%d)", j);
	IloNumVar U(env, 0, IloInfinity, ILOFLOAT, metavlitiU);
	Uj.add(U);

}

//Yk
IloNumVarArray Yk(env, 0);
for (k = 0; k < kmax; k++) {
	char metavlitiY[70];
	sprintf(metavlitiY, "Yk(k%d)", i, k);
	IloNumVar Y(env, 0, 1440, ILOFLOAT, metavlitiY);
	Yk.add(Y);
}



//Periorismoi

//Single visit 1
IloRangeArray SumXi(env, 0);
for (j = 1; j < jmax; j++) {
	IloExpr expr(env, 0);
	for (i = 0; i < imax; i++) {
		for (k = 0; k < kmax; k++) {
			expr += Xijk[i][j][k];
```

`

```
                }
        }
        char SumXijk[60];
        sprintf(SumXijk, "SumXijk(j%d)", j);
        float LB = 1, UB = 1;
        IloRange SumX1(env, LB, expr, UB, SumXijk);
        model.add(SumX1);
        SumXi.add(SumX1);
        expr.end();
}

//Single visit 2
IloRangeArray SumXj(env, 0);
for (i = 1; i < imax; i++) {
        IloExpr expr(env, 0);
        for (j = 0; j < jmax; j++) {
                for (k = 0; k < kmax; k++) {
                        expr += Xijk[i][j][k];
                }
        }
        char SumX2ijk[60];
        sprintf(SumX2ijk, "SumX2ijk(i%d)", i);
        float LB = 1, UB = 1;
        IloRange SumX2(env, LB, expr, UB, SumX2ijk);
        model.add(SumX2);
        SumXj.add(SumX2);
        expr.end();
}

//Starting point
IloRangeArray SumX0jk(env, 0);
for (k = 0; k < kmax; k++) {
```

`

```
		IloExpr expr(env, 0);
		for (j = 0; j < jmax; j++) {
			expr += Xijk[0][j][k];
		}
		char SumXj[60];
		sprintf(SumXj, "SumX0jk(k%d)", k);
		float LB = 0, UB = 1;
		IloRange SumX(env, LB, expr, UB, SumXj);
		model.add(SumX);
		SumX0jk.add(SumX);
		expr.end();
	}

	//Final point
	IloRangeArray SumXi0k(env, 0);
	for (k = 0; k < kmax; k++) {
		IloExpr expr(env, 0);
		for (i = 0; i < imax; i++) {
			expr += Xijk[i][0][k];
		}
		char SumXi[60];
		sprintf(SumXi, "SumX0jk(k%d)", k);
		float LB = 0, UB = 1;
		IloRange SumX(env, LB, expr, UB, SumXi);
		model.add(SumX);
		SumXi0k.add(SumX);
		expr.end();
	}

	//Force na xekinaei apo to 0
	IloRangeArray XEKINAAPOTO0RE(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 = 1; j < jmax; j++) {
        expr1 += Xijk[i][j][k];
    }
}
for (h = 1; h < hmax; h++) {
    expr2 -= Xijk[0][h][k];
}
expr = expr1 + expr2 * jmax;
char XEKINA[60];
sprintf(XEKINA,  "XEKINAAPOTO0(i%d,j%d,k%d)",  i,  j,
k);
float LB = -IloInfinity, UB = 0;
IloRange CNGXj(env, LB, expr, UB, XEKINA);
model.add(CNGXj);
XEKINAAPOTO0RE.add(CNGXj);
expr.end();
}


//CNG
IloRangeMatrix2x2 CNGXijk(env, 0);
for (k = 0; k < kmax; k++) {
    IloRangeArray CNGXjk(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 CNGXijk[60];
                sprintf(CNGXijk, "CNGXijk(i%d,j%d,k%d)", i, j, k);
                float LB = 0, UB = 0;
                IloRange CNGXj(env, LB, expr, UB, CNGXijk);
                model.add(CNGXj);
                CNGXjk.add(CNGXj);
                expr.end();
        }
        CNGXijk.add(CNGXjk);
}

//Kalipsi Zitisis 1

IloRangeArray D1ijk(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 += D1[i] * Xijk[i][j][k];


                }
        }
        char Zitisi1[60];
        sprintf(Zitisi1, "D1ijk(k%d)", k);
        float LB = 0, UB = Q1[k];
        IloRange SumX(env, LB, expr, UB, Zitisi1);
        model.add(SumX);
        D1ijk.add(SumX);
        expr.end();
}
```

`

```
//Kalipsi Zitisis 2

IloRangeArray D2ijk(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 += D2[i] * Xijk[i][j][k];


		}
	}
	char Zitisi2[60];
	sprintf(Zitisi2, "D2ijk(k%d)", k);
	float LB = 0, UB = Q2[k];
	IloRange SumX(env, LB, expr, UB, Zitisi2);
	model.add(SumX);
	D2ijk.add(SumX);
	expr.end();
}

//Miller–Tucker–Zemlin
IloRangeMatrix3x3 cng1ijk2(env, 0);
for (i = 1; i < imax; i++) {
	IloRangeMatrix2x2 cng1jk2(env, 0);
	for (j = 1; j < jmax; j++) {
		IloRangeArray cng1k2(env, 0);
		for (k = 0; k < kmax; k++) {
			IloExpr expr(env, 0);
			expr += Uj[i];
			expr -= Uj[j];
			expr += (imax - 1) * Xijk[i][j][k];
```

`

```
                                        char comego2[60];
                                        sprintf(comego2,  "cng12(i%d,j%d,k%d)",  i,
j, k);
                                        float LB = -IloInfinity, UB = imax - 2;
                                        IloRange  cng12(env,  LB,  expr,  UB,
comego2);
                                        expr.end();
                                        model.add(cng12);
                                        cng1k2.add(cng12);
                                }
                                cng1jk2.add(cng1k2);
                        }
                        cng1ijk2.add(cng1jk2);
                }

                ////Time Windows

                //Time Windows1
                IloRangeMatrix2x2 TWSik(env, 0);
                for (k = 0; k < kmax; k++) {
                        IloRangeArray TWSk(env, 0);
                        for (i = 0; i < imax; i++) {
                                IloExpr expr(env, 0);
                                IloExpr expr1(env, 0);
                                expr -= tik[i][k];
                                for (j = 0; j < jmax; j++) {
                                        expr1 += Xijk[j][i][k];
                                }
                                expr += expr1 * TWS[i];
                                char TimeWindowsS[60];
                                sprintf(TimeWindowsS, "TWSik(i%d,k%d)", i, k);
                                float LB = -IloInfinity, UB = 0;
                                IloRange TWS(env, LB, expr, UB, TimeWindowsS);
```

`

```
                model.add(TWS);
                TWSk.add(TWS);
                expr.end();
        }
        TWSik.add(TWSk);
}

//Time Windows2
IloRangeMatrix2x2 TWFik(env, 0);
for (k = 0; k < kmax; k++) {
        IloRangeArray TWFk(env, 0);
        for (i = 0; i < imax; i++) {
                IloExpr expr(env, 0);
                IloExpr expr1(env, 0);
                expr += tik[i][k];
                for (j = 0; j < jmax; j++) {
                        expr1 -= Xijk[j][i][k];
                }
                expr += expr1 * (TWF[i] + MD[i]);
                char TimeWindowsF[60];
                sprintf(TimeWindowsF, "TWik(i%d,k%d)", i, k);
                float LB = -IloInfinity, UB = 0;
                IloRange TWF(env, LB, expr, UB, TimeWindowsF);
                model.add(TWF);
                TWFk.add(TWF);
                expr.end();
        }
        TWFik.add(TWFk);
}

//Time Window Relatability
IloNumVarMatrix3x3 TWRijk(env, 0);
```

`

```
for (i = 0; i < imax; i++) {
    IloNumVarMatrix2x2 TWRjk(env, 0);
    for (j = 1; j < jmax; j++) {
        IloNumVarArray TWRk(env, 0);
        for (k = 0; k < kmax; k++) {
            IloExpr expr(env, 0);
            IloExpr expr1(env, 0);
            expr -= tik[j][k];
            expr += tik[i][k];
            expr += M * Xijk[i][j][k];
            char TimeWindowsRel[60];
            sprintf(TimeWindowsRel,
"TWRjk(i%d,j%d,k%d)", i, j, k);
            float LB = -IloInfinity, UB = M - Tij[i][j] -
MD[i];
            IloRange SumX(env, LB, expr, UB,
TimeWindowsRel);
            model.add(SumX);
            TWRk.add(SumX);
            expr.end();
            expr1.end();
        }
        TWRjk.add(TWRk);
    }
    TWRijk.add(TWRjk);
}

//Telos
IloRangeMatrix2x2 Tik(env, 0);
for (k = 0; k < kmax; k++) {
    IloRangeArray Tk(env, 0);
    for (i = 1; i < imax; i++) {
        IloExpr expr(env, 0);
```

`

```
                expr += Yk[k];
                expr -= tik[i][k];
                expr += Xijk[i][0][k] * (Tij[i][0] + MD[i]);
                char Telos[60];
                sprintf(Telos, "Telos(i%d,k%d)", i, k);
                float LB = 0, UB = IloInfinity;
                IloRange T(env, LB, expr, UB, Telos);
                model.add(T);
                Tk.add(T);
                expr.end();
        }
        Tik.add(Tk);
}

//Midenismos Yk pou den kanoun dromologia
IloRangeArray MYk(env, 0);
for (k = 0; k < kmax; k++) {
        IloExpr expr(env, 0);
        IloExpr expr1(env, 0);
        expr += Yk[k];
        for (i = 1; i < imax; i++) {
                expr1 -= Xijk[i][0][k];
        }
        expr += expr1 * 1440;
        char Telos1[60];
        sprintf(Telos1, "Telos1(k%d)", k);
        float LB = -IloInfinity, UB = 0;
        IloRange MY(env, LB, expr, UB, Telos1);
        model.add(MY);
        MYk.add(MY);
        expr.end();
}
```

`

```
//Kalipsi 8wrou
IloRangeArray oktawro(env, 0);
for (k = 0; k < kmax; k++) {
        IloExpr expr(env, 0);
        expr += Yk[k];
        expr -= tik[0][k];
        char oktawro1[60];
        sprintf(oktawro1, "oktawro(k%d)", k);
        float LB = -IloInfinity, UB = 480;
        IloRange SumX(env, LB, expr, UB, oktawro1);
        model.add(SumX);
        oktawro.add(SumX);
        expr.end();
}


//Antikeimeniki
IloExpr expr5(env, 0);

for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                for (k = 0; k < kmax; k++) {
                        expr5 += Xijk[i][j][k] * Dij[i][j];
                }
        }
}



model.add(IloMinimize(env, expr5));
expr5.end();
```

`

```
cplex.extract(model);
cplex.exportModel("Montelo.lp");
//cplex.setParam(IloCplex::EpGap, 0.50);
//cplex.setParam(IloCplex::TiLim, 10);   //(time limit)/2 gia kathe komati toy provlimatos
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;
if (cplex.getObjValue() < mind) mind = cplex.getObjValue();
if (cplex.getObjValue() > maxd) maxd = cplex.getObjValue();
totald = totald + cplex.getObjValue();



//outFile.open("lisi.txt");

for (k = 0; k < kmax; k++) {
        for (i = 0; i < imax; i++) {
                for (j = 0; j < jmax; j++) {
                        float g = cplex.getValue(Xijk[i][j][k]);
                        if (i == 0 && j == 0) {
                                if (g != 0) cout << "Xijk" << "(" << i << "," << j << "," << k + d + 1 << ")" << "=" << g << endl;
                                if (g != 0) outFile << "Xijk" << "(" << i << "," << j << "," << k + d + 1 << ")" << "=" << g << endl;
                        }
                        else if (i == 0) {
```

`

```
                                        if (g != 0) cout << "Xijk" << "(" << i << "," << j + c << "," << k + d + 1 << ")" << "=" << g << endl;

                                        if (g != 0) outFile << "Xijk" << "(" << i << "," << j + c << "," << k + d + 1 << ")" << "=" << g << endl;

                                    }

                                    else if (j == 0) {

                                        if (g != 0) cout << "Xijk" << "(" << i + c << "," << j << "," << k + d + 1 << ")" << "=" << g << endl;

                                        if (g != 0) outFile << "Xijk" << "(" << i + c << "," << j << "," << k + d + 1 << ")" << "=" << g << endl;

                                    }

                                    else {

                                        if (g != 0) cout << "Xijk" << "(" << i + c << "," << j + c << "," << k + d + 1 << ")" << "=" << g << endl;

                                        if (g != 0) outFile << "Xijk" << "(" << i + c << "," << j + c << "," << k + d + 1 << ")" << "=" << g << endl;

                                    }

                                }

                            }

                        }


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

                            for (i = 0; i < imax; i++) {

                                float g = cplex.getValue(tik[i][k]);

                                if (i == 0) {

                                    if (g != 0) cout << "tik" << "(" << i << " , " << k + d + 1 << ")" << "=" << g << endl;

                                    if (g != 0) outFile << "tik" << "(" << i << " , " << k + d + 1 << ")" << "=" << g << endl;

                                }

                                else {

                                    if (g != 0) {

                                        cout << "tik" << "(" << i + c << " , " << k + d + 1 << ")" << "=" << g << endl;

                                        dap = dap + D1[i];
```

`

```
                                        dbp = dbp + D2[i];
                                        p += 1;
                                }
                                if (g != 0) outFile << "tik" << "(" << i + c <<
" , " << k + d + 1 << ")" << "=" << g << endl;
                        }
                }
                xorap = dap / Q1[k];
                xorbp = dbp / Q2[k];
                xorapg += dap / Q1[k];
                xorbpg += dbp / Q2[k];
                if (xorap > 0) cout << "fortigo " << k + d + 1 << ": " << xorap
<< "      " << xorbp << endl;
                if (minp > p && p>0) minp = p;
                if (maxp < p) maxp = p;
                p = 0;
                xorap = 0;
                xorbp = 0;
                dap = 0;
                dbp = 0;
        }

        for (k = 0; k < kmax; k++) {
                float g = cplex.getValue(Yk[k]);
                if (g != 0) {
                        cout << "Yk" << "(" << k + d + 1 << ")" << "=" << g
<< endl;
                        totalf = totalf + 1;
                }
                if (g != 0) outFile << "Yk" << "(" << k + d + 1 << ")" << "="
<< g << endl;
        }

    }
```

`

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



            env.end();
            c = c + imax - 1;
            d = d + kmax;

        } while (c<pel - 1);

        cout << totald << endl;
        cout << totalf << endl;
        cout << totald / totalf << endl;
        cout << maxd << "        " << mind << endl;
        cout << xorapg / totalf << "        " << xorbpg / totalf << endl;
        cout << (pel - 2) / totalf << endl;
        cout << maxp << "        " << minp << endl;
        outFile.close();
        system("pause");
        return 0;
}
```

`

Κώδικας Nearest Point

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



#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>

using namespace std;

ifstream infile;
ifstream inFile;
ofstream outfile;
ofstream outFile;
ofstream out;

int i, j, c, d, e, f, k, z, h, ax, bx, q, o;
int gm = 0;
int p = 0;
int uv = 0;
double g, pos1, pos2;
double x, s, ev; //mesi timi kai tipiki apoklisi
int AD = 0; // AD = arithmos diadromis
int countDiad, countDiad1;
int countProsp = 0;
int count = 0;
int meso;
double oktawro = 0;
double okt1; //apostaseis xronika
```

`

```
double okt2; //kathisteriseis
double okt3;
double okt4;
int xora, xorb;
double xorag = 0;
double xorbg = 0;
double ta = 0;
double tb = 0;
const int pel = 171;
const int tr = 105;
double kentrox, kentroy, a, b;
double kentrox1, kentroy1; //kentro anaptisomenis diadromis
int adrom = pel - 1;
float totald = 0; //total distance
float totalf = 0; //total fortigwn
float maxd = 0; //max distance
float mind = 1000000; //min distance
float dap = 0;
float dbp = 0;
float xorap = 0; //pososto kalipsis a
float xorbp = 0; //pososto kalipsis b
float xorapg = 0; //pososto kalipsis a geniko
float xorbpg = 0; //pososto kalipsis b geniko
float minp = 500; //elaxistoi pelates se diadromi
float maxp = 0; //megistoi pelates se diadromi

double pelates[pel][8]; //teleftea stili: se poia diadromi einai aftos o pelatis
double D[pel][pel];
double T[pel][pel];
double DDij[pel][pel]; //neos pinakas Dij
double TTij[pel][pel]; //neos pinakas Tij
double TW[49]; // 48 misawra i kathe mera + 1 telos / posoi pelates mporoun na
eksipiretithoun to sigkekrimeno 30lepto
```

```
`

double trucks[tr][3]; //xora,xorb,diadromi
double fort[2]; //synthiki fortigwn
double diadromi[pel - 1][4]; // i tetarti stili einai gia to psaksimo stous 3 kontinous
int id[pel];

//orismata gia to tsp
const int meg = 30; //megistos arithmos konvwn se kathe diadromi
int NumDiad[50]; //arithmos komvwn sti diadromi
int D1[meg];
int D2[meg];
int MD[meg];
int TWS[meg];
int TWF[meg];
double Dij[meg][meg];
double Tij[meg][meg];
int kmax = 1;
double pelatesTSP[meg][5];
const int M = 1000000;
int hmax, zmax, jmax, imax;
float total;

//synartiseis
double insert_pelates();
double insert_fortiga();
double calculateDistance(double x, double y, double x1, double y1);

void main() {

	insert_pelates(); //eisagwgi stoixeiwn pelatwn, eisagwgi stoixeiwn TW[i], i o kathe pelatis
	outfile.open("statistika.txt");

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

`

```
		ax += pelates[i][0];
		bx += pelates[i][1];
		for (k = 0; k < pel; k++) {
			x += DDij[i][k];
		}
}

x = x / ((pel - 1)*(pel - 1));
cout << "xmeso= " << x << endl;
outfile << "xmeso= " << x << endl;

for (i = 0; i < pel; i++) {
		for (k = 0; k < pel; k++) {
			s += (DDij[i][k] - x)*(DDij[i][k] - x);
		}
}
s = s / ((pel - 1)*(pel - 1));
cout << "s= " << sqrt(s) / x << endl;
outfile << "s= " << sqrt(s) / x << endl;

ev = 4 * (1 + (sqrt(s) / x));
cout << "cv= " << ev << endl;
outfile << "cv= " << ev << endl;
cout << "ax= " << ax << endl;
outfile << "ax= " << ax << endl;
cout << "bx= " << bx << endl;
outfile << "bx= " << bx << endl;

outfile.close();

insert_fortiga(); //eisagwgi stoixeiwn fortigwn, kata afksousa seira se eidos a
fort[0] = trucks[0][0];
```

`

```
	fort[1] = trucks[0][1];


	cout << "forta: " << fort[0] << " fortb: " << fort[1] << endl;


	//EPILOGI KOMVWN ME VASI NEAREST POINT (TEMPORAL AND
GEOGRAFICAL DEPENDANCE)


	//arxikopoiisi pinaka arithmou komvwn se diadromi
	for (i = 0; i < 50; i++) {
		NumDiad[i] = 0;
	}


	//arxikopoiisi pinaka dromologisis
	for (i = 0; i < pel; i++) {
		diadromi[i][0] = 0; //an exei mpei se diadromi
		diadromi[i][1] = i; //poios einai
		diadromi[i][2] = 0; //
		diadromi[i][3] = 0; //an exei xrisimopoiithei se afti ti diadromi
	}


	//ypologismos kentrou varous komvwn
	for (i = 1; i < pel; i++) {
		kentrox += pelates[i][5];
		kentroy += pelates[i][6];
	}
	kentrox = kentrox / (pel - 1);
	kentroy = kentroy / (pel - 1);


	cout << "geniko kentro varous: " << kentrox << "		" << kentroy << endl;


	out.open("synt1.txt");


	do {
```

`

```
		//vriskoume pio makrino komvo apo kentro varos pou den exei dromologithei gia na ksekinisoume epilogi
		a = calculateDistance(pelates[1][5], pelates[1][6], kentrox, kentroy);
		for (i = 2; i < pel; i++) {
			if (pelates[i][7] == 0) {
				b = calculateDistance(pelates[i][5], pelates[i][6], kentrox, kentroy);
				if (b > a) {
					a = b;
					d = i; //poios pelatis einai
				}
				cout << i << " : " << a << endl;
			}
		}
		cout << "pio makrinos pelatis: " << d << endl;

		AD += 1;
		pelates[d][7] = AD;
		diadromi[d][0] = 1;
		countDiad = 0;
		gm += 1;

		if (gm > 2) {
			pelates[d][7] = AD;
			diadromi[d][0] = 1;
			adrom -= 1;
			gm = 0;
		}

		//dimiourgia diadromis
		do {
			//ypologismos kentrou varous komvwn
			kentrox1 = 0;
```

`

```
kentroy1 = 0;
for (i = 1; i < pel; i++) {
	if (pelates[i][7] == AD) {
		kentrox1 += pelates[i][5];
		kentroy1 += pelates[i][6];
		countDiad += 1;
		cout << kentrox1 << endl;
	}
}
cout << countDiad << endl;
kentrox1 = kentrox1 / countDiad;
kentroy1 = kentroy1 / countDiad;

cout << "kentro varous komvwn diadromis: " << kentrox1 << " , " << kentroy1 << endl;

//kontinoteros pelatis
meso = 0;
a = calculateDistance(0, 0, kentrox1, kentroy1);
for (i = 1; i < pel; i++) {
	if (diadromi[i][0] == 0) {
		b = calculateDistance(pelates[i][5], pelates[i][6], kentrox1, kentroy1);
		if (b < a) {
			a = b;
			meso += a;
			e = i; //poios pelatis einai
		}
		//cout << i << " : " << a << endl;
	}
}

meso = meso / countDiad;
```

`

```
                cout << meso << endl;

                //exit loop gia ton telefteo pelati
                if (adrom < 2) {
                        pelates[e][7] = AD - 1;
                        goto stop;
                }

                cout << "kontinoteros pelatis stin diadromi: " << e << endl;

                //elegxos gia 3 kontinoterous
                countProsp = 0;
                do {
                        //kontinoteros pelatis ston kontinotero pelati
                        a = calculateDistance(0, 0, kentrox1, kentroy1);
                        for (i = 1; i < pel; i++) {
                                if (pelates[i][7] == AD && diadromi[i][3] == 0) {
                                        b = calculateDistance(pelates[i][5],
pelates[i][6], pelates[e][5], pelates[e][6]);
                                        if (b < a) {
                                                a = b;
                                                f = i; //poios pelatis einai
                                        }
                                        //cout << i << " : " << a << endl;
                                }
                        }

                        xora = 0;
                        xorb = 0;
                        for (i = 1; i < pel; i++) {
                                if (pelates[i][7] == AD) {
                                        xora += pelates[i][0];
                                        xorb += pelates[i][1];
```

`

```
                    }
                }

                //elegxos pelati gia eisodo sti diadromi
                if (pelates[e][3] - pelates[f][4] <= TTij[e][f] || pelates[f][3] -
pelates[e][4] <= TTij[f][e]) {

                    if (xora >= trucks[0][0] && xorb >= trucks[0][1]) {
                        pelates[e][7] = 0;
                        diadromi[e][0] = 0;
                        adrom += 1;
                        diadromi[e][2] = 0;
                    }
                    else {
                        for (i = 0; i < pel; i++) {
                            diadromi[i][3] = 0;
                        }
                        countProsp = 3;
                        pelates[e][7] = AD;
                        diadromi[e][0] = 1;
                        adrom -= 1;
                        diadromi[e][2] = countDiad;

                        xora = 0;
                        xorb = 0;
                        for (i = 1; i < pel; i++) {
                            if (pelates[i][7] == AD) {
                                xora += pelates[i][0];
                                xorb += pelates[i][1];
                            }
                        }
```

`

```
                        }
                }
                else {
                        countProsp += 1;
                        diadromi[i][3] = 1;
                }

        } while (countProsp < 3);

        //synthiki xoritikotitas a kai b
        xora = 0;
        xorb = 0;
        for (i = 1; i < pel; i++) {
                if (pelates[i][7] == AD) {
                        xora += pelates[i][0];
                        xorb += pelates[i][1];
                }
        }
        if (xora > trucks[0][0] || xorb > trucks[0][1]) {
                pelates[e][7] = 0;
                diadromi[e][0] = 0;
                adrom += 1;
                diadromi[e][2] = 0;
        }
        cout << "xora: " << xora << "     xorb: " << xorb << endl;
        cout << pelates[e][7] << "          " << diadromi[e][0] << "            "
<< diadromi[d][1] << endl;

        //synthiki oktawrou
        okt1 = 0;
        okt2 = 0;
        okt3 = 0;
        oktawro = 0;
```

`

```
                        for (i = 1; i < pel; i++) {
                                if (pelates[i][7] == AD) {
                                        okt2 += pelates[i][2];
                                }
                                for (j = 1; j < pel; j++) {
                                        if (pelates[i][7] == AD && pelates[j][7] == AD) {
                                                okt1 += TTij[i][j];
                                                okt3 += TTij[0][i];
                                                okt3 += TTij[i][0];
                                        }
                                }
                        }
                        okt1 = okt1 / (2 * countDiad);
                        okt3 = okt3 / (ev * countDiad);
                        oktawro = okt1 + okt2 + okt3;

                        countDiad = 0;

                } while (xora < fort[0] && xorb < fort[1] && oktawro<480);

        } while (adrom > 0);

        //print diadromwn
stop:
        countDiad = 0;
        for (j = 1; j <= AD; j++) {
                countDiad1 = 0;
                xora = 0;
                xorb = 0;
                for (i = 0; i <= pel; i++) {
                        if (pelates[i][7] == j) {
                                cout << i << "    :          " << pelates[i][7] << "    //          "
<< diadromi[i][2] << endl;
```

`

```
                        out << i << ", ";
                        countDiad += 1;
                        countDiad1 += 1;
                        xora += pelates[i][0];
                        xorb += pelates[i][1];
                        xorag += pelates[i][0];
                        xorbg += pelates[i][1];
                }
        }
        out << endl << endl;
        NumDiad[j - 1] = countDiad1;
        for (i = tr - 1; i > -1; i -= 1) {
                if (xora < trucks[i][0] && xorb < trucks[i][1] && trucks[i][2] == 0) {
                        trucks[i][2] = j;
                        goto end;
                }
        }
end:
        cout << "komvoi diadromis: " << countDiad1 << endl;
        cout << "fortigo pou eksipiretei ti diadromi: " << i + 1 << endl;
        cout << "me kalipsi a fortigou: " << xora << " apo " << trucks[i][0] << endl;
        cout << "me kalipsi b fortigou: " << xorb << " apo " << trucks[i][1] << endl;
        ta = ta + trucks[i][0];
        tb = tb + trucks[i][1];
        cout << xorag << "        " << xorbg << endl;
        cout << ta << "  " << tb << endl;
        pos1 = xorag / ta;
        pos2 = xorbg / tb;
        cout << "kalipsi xwritikotitas: " << pos1 << "      " << pos2 << endl;
}
cout << countDiad << endl;
```

```
`

	for (i = 0; i < tr; i++) {
		if (!(trucks[i][2] == 0)) cout << "fortigo: " << i + 1 << " diadromi: " << trucks[i][2] << endl;
	}

	//neoi pinakes
	for (i = 0; i < pel; i++) {
		id[i] = 0;
	}

	outfile.open("pelatesnew.txt");

	if (outfile.fail())
	{
		cout << "output file pelatesnew could not be opened" << endl;
		system("pause");
		exit(1);
	}

	outfile << "0	0	0	0	1440" << endl;
	c = 1;
	for (i = 1; i <= AD; i++) {
		for (j = 1; j < pel; j++) {
			if (pelates[j][7] == i) {
				for (d = 0; d < 5; d++) {
					outfile << pelates[j][d] << "	";
				}
				outfile << endl;
				id[c] = j;
				c += 1;
			}
		}
	}
```

`

```
outfile.close();

outfile.open("Dijnew.txt");

for (i = 0; i < pel; i++) {
        for (j = 0; j < pel; j++) {
                D[i][j] = DDij[id[i]][id[j]];
                outfile << D[i][j] << "    ";
        }
        outfile << endl;
}

outfile.close();

outfile.open("Tijnew.txt");

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

outfile.close();
out.close();

infile.close();

//----------------------
//TSP gia kathe diadromi
```

`

```
//----------------------

//c gia to tsp tha einai to counter twn pelatwn pou exoun dromologithei
c = 0;
//

outFile.open("lisi.txt");
outfile.open("Synt.txt");

for (d = 0; d < 50; d++) {

        imax = NumDiad[d] + 1;
        jmax = imax;
        hmax = imax;
        zmax = imax;

        if (NumDiad[d] != 0) {

                //=========================================
                //eisagwgi dedomenwn stous pinakes gia to TSP
                //=========================================

                //arxikopoiisi stoixeiwn pelatwn
                for (i = 0; i < meg; i++) {
                        D1[i] = 0;
                        D2[i] = 0;
                        MD[i] = 0;
                        TWS[i] = 0;
                        TWF[i] = 0;
                }

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

`

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

                        //eisagwgi stoixeiwn pelatwn diadromis
                        for (i = 0; i < pel; i++) {
                                if (i == 0)
                                {
                                        infile >> D1[0] >> D2[0] >> MD[0] >> TWS[0] >>
TWF[0];
                                }
                                else if (i > c && i <= c + NumDiad[d])
                                {
                                        infile >> D1[i - c] >> D2[i - c] >> MD[i - c] >>
TWS[i - c] >> TWF[i - c];
                                }
                                else
                                {
                                        for (j = 0; j < 5; j++) {
                                                infile >> e;
                                        }
                                }
                        }

                        infile.close();

                        cout << "Diadromi: " << d + 1 << endl << endl;
                        /*for (i = 0; i < NumDiad[d] + 1; i++) {
                        cout << D1[i] << "         " << D2[i] << " " << MD[i] << "            "
<< TWS[i] << "" << TWF[i] << endl;
```

`

```
                }
                cout << endl;*/


                //arxikopoiisi stoixeiwn apostasis pelatwn diadromis
                for (i = 0; i < meg; i++) {
                        for (j = 0; j < meg; j++) {
                                Dij[i][j] = 0;
                        }
                }

                //eisagwgi stoixeiwn apostasis pelatwn diadromis
                infile.open("Dijnew.txt");
                if (infile.fail())
                {
                        cout << "Dijnew file could not be opened" << endl;
                        system("pause");
                        exit(1);
                }

                for (i = 0; i < pel; i++) {
                        for (j = 0; j < pel; j++) {
                                if (j <= NumDiad[d] + c && j > c && i <=
NumDiad[d] + c && i > c) {
                                        infile >> Dij[i - c][j - c];
                                }
                                else if (j == 0 && i <= NumDiad[d] + c && i > c) {
                                        infile >> Dij[i - c][j];
                                }
                                else if (i == 0 && j <= NumDiad[d] + c && j > c) {
                                        infile >> Dij[i][j - c];
                                }
                                else {
```

`

```
                                infile >> b;
                        }
                }
        }

        /*for (i = 0; i < NumDiad[d] + 1; i++) {
        for (j = 0; j < NumDiad[d] + 1; j++) {
        cout << Dij[i][j] << "    ";
        }
        cout << endl;
        }*/

        infile.close();

        //arxikopoiisi stoixeiwn xronou pelatwn diadromis
        for (i = 0; i < meg; i++) {
                for (j = 0; j < meg; j++) {
                        Tij[i][j] = 0;
                }
        }

        //eisagwgi stoixeiwn apostasis pelatwn diadromis
        infile.open("Tijnew.txt");
        if (infile.fail())
        {
                cout << "Tijnew file could not be opened" << endl;
                system("pause");
                exit(1);
        }

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

`

```
                                        if (j <= NumDiad[d] + c && j > c && i <=
NumDiad[d] + c && i > c) {
                                                infile >> Tij[i - c][j - c];
                                        }
                                        else if (j == 0 && i <= NumDiad[d] + c && i > c) {
                                                infile >> Tij[i - c][j];
                                        }
                                        else if (i == 0 && j <= NumDiad[d] + c && j > c) {
                                                infile >> Tij[i][j - c];
                                        }
                                        else {
                                                infile >> b;
                                        }
                                }
                        }

                        /*      for (i = 0; i < NumDiad[d] + 1; i++) {
                        for (j = 0; j < NumDiad[d] + 1; j++) {
                        cout << Tij[i][j] << "    ";
                        }
                        cout << endl;
                        }*/

                        infile.close();

                        //============================================
                        //Montelo TSP
                        //============================================

                        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);

//Metavlites

//Xijk
IloNumVarMatrix3x3 Xijk(env, 0);
for (i = 0; i < imax; i++) {
        IloNumVarMatrix2x2 Xjk(env, 0);
        for (j = 0; j < jmax; j++) {
                IloNumVarArray Xk(env, 0);
                for (k = 0; k < kmax; k++) {
                        char metavlitiX[70];
                        sprintf(metavlitiX,
"Xijk(i%d,j%d,k%d)", i, j, k);
                        IloNumVar X(env, 0, 1, ILOINT,
metavlitiX);
                        Xk.add(X);
                }
```

`

```
                    Xjk.add(Xk);
                }
                Xijk.add(Xjk);
            }

            //tik
            IloNumVarMatrix2x2 tik(env, 0);
            for (i = 0; i < imax; i++) {
                IloNumVarArray tk(env, 0);
                for (k = 0; k < kmax; k++) {
                    char metavlitit[70];
                    sprintf(metavlitit, "tik(i%d,k%d)", i, k);
                    IloNumVar t(env, 0, 1440, ILOFLOAT,
metavlitit);
                    tk.add(t);
                }
                tik.add(tk);
            }

            //Uj
            IloNumVarArray Uj(env, 0);
            for (j = 0; j < jmax; j++) {
                char metavlitiU[70];
                sprintf(metavlitiU, "Ui(j%d)", j);
                IloNumVar U(env, 0, IloInfinity, ILOFLOAT,
metavlitiU);
                Uj.add(U);

            }

            //Yk
            IloNumVarArray Yk(env, 0);
            for (k = 0; k < kmax; k++) {
```

`

```
                char metavlitiY[70];
                sprintf(metavlitiY, "Yk(k%d)", i, k);
                IloNumVar Y(env, 0, 1440, ILOFLOAT,
metavlitiY);
                Yk.add(Y);
        }



        //Periorismoi

        //Single visit 1
        IloRangeArray SumXi(env, 0);
        for (j = 1; j < jmax; j++) {
                IloExpr expr(env, 0);
                for (i = 0; i < imax; i++) {
                        for (k = 0; k < kmax; k++) {
                                expr += Xijk[i][j][k];
                        }
                }
                char SumXijk[60];
                sprintf(SumXijk, "SumXijk(j%d)", j);
                float LB = 1, UB = 1;
                IloRange SumX1(env, LB, expr, UB, SumXijk);
                model.add(SumX1);
                SumXi.add(SumX1);
                expr.end();
        }

        //Single visit 2
        IloRangeArray SumXj(env, 0);
        for (i = 1; i < imax; i++) {
                IloExpr expr(env, 0);
```

`

```
            for (j = 0; j < jmax; j++) {
                  for (k = 0; k < kmax; k++) {
                        expr += Xijk[i][j][k];
                  }
            }
            char SumX2ijk[60];
            sprintf(SumX2ijk, "SumX2ijk(j%d)", j);
            float LB = 1, UB = 1;
            IloRange SumX2(env, LB, expr, UB, SumX2ijk);
            model.add(SumX2);
            SumXj.add(SumX2);
            expr.end();
      }

      //Starting point
      IloRangeArray SumX0jk(env, 0);
      for (k = 0; k < kmax; k++) {
            IloExpr expr(env, 0);
            for (j = 0; j < jmax; j++) {
                  expr += Xijk[0][j][k];
            }
            char SumXj[60];
            sprintf(SumXj, "SumX0jk(k%d)", k);
            float LB = 0, UB = 1;
            IloRange SumX(env, LB, expr, UB, SumXj);
            model.add(SumX);
            SumX0jk.add(SumX);
            expr.end();
      }

      //Final point
      IloRangeArray SumXi0k(env, 0);
```

`

```
for (k = 0; k < kmax; k++) {
        IloExpr expr(env, 0);
        for (i = 0; i < imax; i++) {
                expr += Xijk[i][0][k];
        }
        char SumXi[60];
        sprintf(SumXi, "SumX0jk(k%d)", k);
        float LB = 0, UB = 1;
        IloRange SumX(env, LB, expr, UB, SumXi);
        model.add(SumX);
        SumXi0k.add(SumX);
        expr.end();
}

//Force na xekinaei apo to 0
IloRangeArray XEKINAAPOTO0RE(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 = 1; j < jmax; j++) {
                        expr1 += Xijk[i][j][k];
                }
        }
        for (z = 1; z < zmax; z++) {
                expr2 -= Xijk[0][z][k];
        }
        expr = expr1 + expr2 * jmax;
        char XEKINA[60];
        sprintf(XEKINA,
"XEKINAAPOTO0(i%d,j%d,k%d)", i, j, k);
        float LB = -IloInfinity, UB = 0;
```

`

```
                IloRange CNGXj(env, LB, expr, UB, XEKINA);
                model.add(CNGXj);
                XEKINAAPOTO0RE.add(CNGXj);
                expr.end();
        }


        //CNG
        IloRangeMatrix2x2 CNGXijk(env, 0);
        for (k = 0; k < kmax; k++) {
                IloRangeArray CNGXjk(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 CNGXijk[60];
                        sprintf(CNGXijk,
"CNGXijk(i%d,j%d,k%d)", i, j, k);
                        float LB = 0, UB = 0;
                        IloRange CNGXj(env, LB, expr, UB,
CNGXijk);
                        model.add(CNGXj);
                        CNGXjk.add(CNGXj);
                        expr.end();
                }
                CNGXijk.add(CNGXjk);
        }

        //Miller–Tucker–Zemlin
```

`

```
IloRangeMatrix3x3 cng1ijk2(env, 0);
for (i = 1; i < imax; i++) {
	IloRangeMatrix2x2 cng1jk2(env, 0);
	for (j = 1; j < jmax; j++) {
		IloRangeArray cng1k2(env, 0);
		for (k = 0; k < kmax; k++) {
			IloExpr expr(env, 0);
			expr += Uj[i];
			expr -= Uj[j];
			expr += (imax - 1) * Xijk[i][j][k];
			char comego2[60];
			sprintf(comego2,
"cng12(i%d,j%d,k%d)", i, j, k);
			float LB = -IloInfinity, UB = imax -
2;
			IloRange cng12(env, LB, expr, UB,
comego2);
			expr.end();
			model.add(cng12);
			cng1k2.add(cng12);
		}
		cng1jk2.add(cng1k2);
	}
	cng1ijk2.add(cng1jk2);
}

////Time Windows

//Time Windows1
IloRangeMatrix2x2 TWSik(env, 0);
for (k = 0; k < kmax; k++) {
	IloRangeArray TWSk(env, 0);
	for (i = 0; i < imax; i++) {
```

`

```
                                        IloExpr expr(env, 0);
                                        IloExpr expr1(env, 0);
                                        expr -= tik[i][k];
                                        for (j = 0; j < jmax; j++) {
                                                expr1 += Xijk[j][i][k];
                                        }
                                        expr += expr1 * TWS[i];
                                        char TimeWindowsS[60];
                                        sprintf(TimeWindowsS,
"TWSik(i%d,k%d)", i, k);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange TWS(env, LB, expr, UB,
TimeWindowsS);
                                        model.add(TWS);
                                        TWSk.add(TWS);
                                        expr.end();
                                }
                                TWSik.add(TWSk);
                        }

                        //Time Windows2
                        IloRangeMatrix2x2 TWFik(env, 0);
                        for (k = 0; k < kmax; k++) {
                                IloRangeArray TWFk(env, 0);
                                for (i = 0; i < imax; i++) {
                                        IloExpr expr(env, 0);
                                        IloExpr expr1(env, 0);
                                        expr += tik[i][k];
                                        for (j = 0; j < jmax; j++) {
                                                expr1 -= Xijk[j][i][k];
                                        }
                                        expr += expr1 * (TWF[i] + MD[i]);
                                        char TimeWindowsF[60];
```

`

```
                                        sprintf(TimeWindowsF, "TWik(i%d,k%d)",
i, k);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange TWF(env, LB, expr, UB,
TimeWindowsF);
                                        model.add(TWF);
                                        TWFk.add(TWF);
                                        expr.end();
                                }
                                TWFik.add(TWFk);
                        }

                        //Time Window Relatability
                        IloNumVarMatrix3x3 TWRijk(env, 0);
                        for (i = 0; i < imax; i++) {
                                IloNumVarMatrix2x2 TWRjk(env, 0);
                                for (j = 1; j < jmax; j++) {
                                        IloNumVarArray TWRk(env, 0);
                                        for (k = 0; k < kmax; k++) {
                                                IloExpr expr(env, 0);
                                                IloExpr expr1(env, 0);
                                                expr -= tik[j][k];
                                                expr += tik[i][k];
                                                expr += M * Xijk[i][j][k];
                                                char TimeWindowsRel[60];
                                                sprintf(TimeWindowsRel,
"TWRjk(i%d,j%d,k%d)", i, j, k);
                                                float LB = -IloInfinity, UB = M -
Tij[i][j] - MD[i];
                                                IloRange SumX(env, LB, expr, UB,
TimeWindowsRel);
                                                model.add(SumX);
                                                TWRk.add(SumX);
                                                expr.end();
```

`

```
                    expr1.end();
                }
                TWRjk.add(TWRk);
            }
            TWRijk.add(TWRjk);
        }

        //Telos
        IloRangeMatrix2x2 Tik(env, 0);
        for (k = 0; k < kmax; k++) {
            IloRangeArray Tk(env, 0);
            for (i = 1; i < imax; i++) {
                IloExpr expr(env, 0);
                expr += Yk[k];
                expr -= tik[i][k];
                expr -= Xijk[i][0][k] * (Tij[i][0] + MD[i]);
                char Telos[60];
                sprintf(Telos, "Telos(i%d,k%d)", i, k);
                float LB = 0, UB = IloInfinity;
                IloRange T(env, LB, expr, UB, Telos);
                model.add(T);
                Tk.add(T);
                expr.end();
            }
            Tik.add(Tk);
        }

        //Midenismos Yk pou den kanoun dromologia
        IloRangeArray MYk(env, 0);
        for (k = 0; k < kmax; k++) {
            IloExpr expr(env, 0);
            IloExpr expr1(env, 0);
```

`

```
		expr += Yk[k];
		for (i = 1; i < imax; i++) {
			expr1 -= Xijk[i][0][k];
		}
		expr += expr1 * 1440;
		char Telos1[60];
		sprintf(Telos1, "Telos1(k%d)", k);
		float LB = -IloInfinity, UB = 0;
		IloRange MY(env, LB, expr, UB, Telos1);
		model.add(MY);
		MYk.add(MY);
		expr.end();
	}

	//Kalipsi 8wrou
	IloRangeArray oktawro(env, 0);
	for (k = 0; k < kmax; k++) {
		IloExpr expr(env, 0);
		expr += Yk[k];
		expr -= tik[0][k];
		char oktawro1[60];
		sprintf(oktawro1, "oktawro(k%d)", k);
		float LB = -IloInfinity, UB = 480;
		IloRange SumX(env, LB, expr, UB, oktawro1);
		model.add(SumX);
		oktawro.add(SumX);
		expr.end();
	}



	//Antikeimeniki
	IloExpr expr5(env, 0);
```

`

```
for (i = 0; i < imax; i++) {
        for (j = 0; j < jmax; j++) {
                for (k = 0; k < kmax; k++) {
                        expr5 += Xijk[i][j][k] * Dij[i][j];
                }
        }
}



model.add(IloMinimize(env, expr5));
expr5.end();

cplex.extract(model);
cplex.exportModel("Montelo.lp");
//cplex.setParam(IloCplex::EpGap, 0.38);
cplex.setParam(IloCplex::TiLim, 20);
cplex.solve();

if (!cplex.solve()) {
        env.error() << "Faild to optimize LP." << endl;
        uv += 1;
        throw(-1);
}

env.out() << "Solution status = " << cplex.getStatus() <<
endl;
env.out() << "Solution value = " << cplex.getObjValue() <<
endl;
total = total + cplex.getObjValue();
if (cplex.getObjValue() < mind) mind =
cplex.getObjValue();
```

`

```
                    if (cplex.getObjValue() > maxd) maxd =
cplex.getObjValue();
                    totald = totald + cplex.getObjValue();



                    //outFile.open("lisi.txt");


                    for (k = 0; k < kmax; k++) {
                        for (i = 0; i < imax; i++) {
                            for (j = 0; j < jmax; j++) {
                                g = cplex.getValue(Xijk[i][j][0]);
                                if (i == 0 && j == 0) {
                                    if (g != 0) {
                                        cout << "Xijk" <<
"(" << i << "," << j << "," << d + 1 << ")" << "=" << g << endl;
                                        outFile << "Xijk"
<< "(" << i << "," << j << "," << d + 1 << ")" << "=" << g << endl;
                                    }
                                }
                                else if (i == 0) {
                                    if (g != 0) {
                                        cout << "Xijk" <<
"(" << i << "," << id[j + c] << "," << d + 1 << ")" << "=" << g << endl;
                                        outFile << "Xijk"
<< "(" << i << "," << id[j + c] << "," << d + 1 << ")" << "=" << g << endl;
                                        q += 1;
                                    }
                                }
                                else if (j == 0) {
                                    if (g != 0) {
                                        cout << "Xijk" <<
"(" << id[i + c] << "," << j << "," << d + 1 << ")" << "=" << g << endl;
                                        outFile << "Xijk"
<< "(" << id[i + c] << "," << j << "," << d + 1 << ")" << "=" << g << endl;
                                    }
```

`

```
                                                }
                                                else {
                                                        if (g != 0) {
                                                                cout << "Xijk" <<
"(" << id[i + c] << "," << id[j + c] << "," << d + 1 << ")" << "=" << g << endl;
                                                                outFile << "Xijk"
<< "(" << id[i + c] << "," << id[j + c] << "," << d + 1 << ")" << "=" << g << endl;
                                                                q += 1;
                                                        }
                                                }
                                        }
                                }
                        }

                        a = 0;
                        g = 0;
                        for (k = 0; k < imax; k++) {
                        patata:
                                for (i = 0; i < imax; i++) {
                                        for (j = 0; j < jmax; j++) {
                                                g = cplex.getValue(Xijk[i][j][0]);
                                                if (g != 0 && i == a) {
                                                        outfile << id[i] << " " <<
pelates[id[i]][5] << " " << pelates[id[i]][6] << endl;
                                                        a = j;
                                                        k += 1;
                                                        if (k >= imax) goto patata1;
                                                        goto patata;
                                                }
                                        }
                                }
                        }
                patata1:
```

`

```
                outfile << endl;


                for (k = 0; k < kmax; k++) {
                        for (i = 0; i < imax; i++) {
                                float g = cplex.getValue(tik[i][k]);
                                if (i == 0) {
                                        if (g != 0) cout << "tik" << "(" << i
<< " , " << k + d + 1 << ")" << "=" << g << endl;
                                        if (g != 0) outFile << "tik" << "(" <<
i << " , " << k + d + 1 << ")" << "=" << g << endl;
                                }
                                else {
                                        if (g != 0) {
                                                cout << "tik" << "(" << id[i
+ c] << " , " << k + d + 1 << ")" << "=" << g << endl;
                                                p += 1;
                                        }
                                        if (g != 0) outFile << "tik" << "(" <<
id[i + c] << " , " << k + d + 1 << ")" << "=" << g << endl;
                                }
                        }
                        if (minp > p && p>0) minp = p;
                        if (maxp < p) maxp = p;
                        p = 0;
                }


                for (k = 0; k < kmax; k++) {
                        float g = cplex.getValue(Yk[k]);
                        if (g != 0) {
                                cout << "Yk" << "(" << k + d + 1 << ")" <<
"=" << g << endl;
                                totalf = totalf + 1;
                        }
```

`

```
                                        if (g != 0) outFile << "Yk" << "(" << k + d + 1 <<
")" << "=" << g << endl;
                                }
                        }
                        catch (IloException& e) {
                                cerr << "concert exception caught:" << e << endl;
                        }
                        catch (...) {
                                cerr << "Unknown exception caught" << endl;
                        }


                        env.end();

                }

                c += NumDiad[d];

        }

        outFile << endl << "total:          " << total;
        cout << totald << endl;
        cout << totalf << endl;
        cout << totald / totalf << endl;
        cout << maxd << "        " << mind << endl;
        cout << pos1 << "        " << pos2 << endl;
        cout << (pel - 2) / totalf << endl;
        cout << maxp << "        " << minp << endl;
        cout << "Failed: " << uv << endl;
        cout << q << endl;
        outFile.close();

        cin >> b;
```

```
`

}

double insert_pelates() {

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

	for (int i = 0; i < pel; i++) {
		for (int j = 0; j < 8; j++) {
			pelates[i][j] = 0;
		}
	}

	for (int i = 0; i < pel; i++) {
		infile >> pelates[i][0] >> pelates[i][1] >> pelates[i][2] >> pelates[i][3] >>
pelates[i][4] >> pelates[i][5] >> pelates[i][6];
	}

	//0: xoritikotita se a
	//1: xoritikotita se b
	//2: delay time se pelati
	//3: time window start
	//4: time window end
	//5: sintetagmeni x
	//6: sintetagmeni y

	for (int i = 0; i < pel; i++) {
		pelates[i][4] -= pelates[i][2];
```

`

```
}

infile.close();

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

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

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

infile.close();

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++) {
		TTij[i][j] = 0;
	}
}

for (i = 0; i < pel; i++) {
	for (j = 0; j < pel; j++) {
		infile >> TTij[i][j];
	}
}

for (int i = 0; i < pel; i++) {
	for (int j = 0; j < pel; j++) {
		TTij[i][j] = TTij[i][j] / 60;
	}
}

infile.close();

for (i = 0; i < 49; i++) {
	TW[i] = 0;
}

b = 0;
for (i = 0; i < 49; i++) {
	b = i * 30;
	for (j = 1; j < pel; j++) {
		if (b <= pelates[j][4] && b >= pelates[j][3]) TW[i]++;
	}
```

```
`
		}

		for (i = 0; i < 49; i++) {
			cout << i * 30 << " : " << TW[i] << endl;
		}
}

double insert_fortiga() {

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

		for (i = 0; i < tr; i++) {
			for (j = 0; j < 3; j++) {
				trucks[i][j] = 0;
			}
		}

		for (i = 0; i < tr; i++) {
			infile >> trucks[i][0] >> trucks[i][1];
		}

		infile.close();

		for (i = 0; i < tr; i++) {
			for (j = 0; j < 3; j++) {
				cout << trucks[i][j] << " ";
```

```
`
                }
                cout << endl;
        }

}

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;
}
```