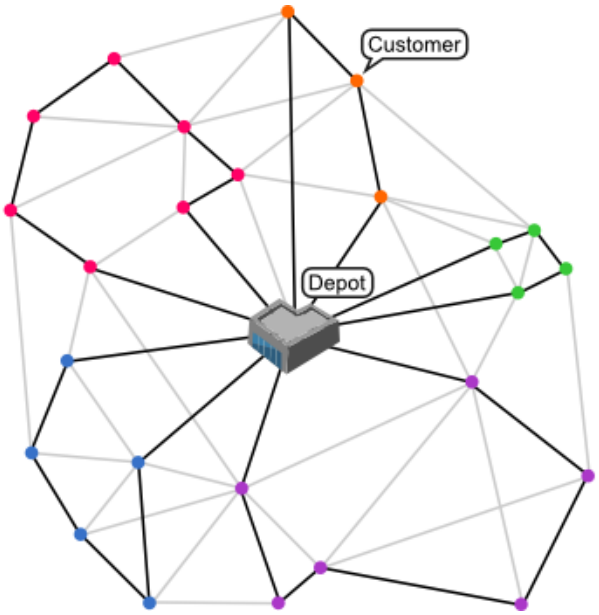
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΑΛΓΟΡΙΘΜΟΣ ΠΕΡΙΟΡΙΣΜΕΝΗΣ ΑΝΑΖΗΤΗΣΗΣ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΧΡΟΝΙΚΑ ΠΑΡΑΘΥΡΑ

(TABU SEARCH ALGORITHM FOR VEHICLE ROUTING PROBLEM WITH TIME WINDOWS)



Συγγραφέας:
ΣΤΟΥΚΑΣ ΓΙΑΝΝΑΚΗΣ

Επιβλέπων Καθηγητής:
ΔΡ. ΜΑΡΙΝΑΚΗΣ ΙΩΑΝΝΗΣ

Χανιά 2015

Περιεχόμενα

Ευχαριστίες

Αρχικά θέλω να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Ιωάννη Μαρινάκη για όλη την υποστήριξη, την καθοδήγηση και τις γνώσεις που μου προσέφερε στη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Σε αυτό το σημείο νιώθω την ανάγκη να ευχαριστήσω προσωπικά κάποια άτομα που με βοήθησαν. Τον Τζίμα Γ. και τον Μοσχονά Γ. γιατί με στήριξαν και με φιλοξένησαν σε μία πραγματικά δύσκολη περίοδο για μένα. Τον Ζόγκα για την βοήθεια του και τη στήριξη του σε πάρα πολλά ζητήματα. Τον Γιάννη Θ. για την παρέα του και τις στιγμές γέλιου όλα αυτά τα χρόνια. Τον Λευτέρη για την βοήθεια στα μαθήματα και όπου τον χρειάστηκα.

Πολύ μεγάλα ευχαριστώ θέλω να δώσω σε κάποιους ξεχωριστούς ανθρώπους, τους συντρόφους μου από την ενωτική πρωτοβουλία που η γνωριμία μου μαζί τους άλλαξε πολλά πράγματα στη ζωή μου. Όλα αυτά τα χρόνια μου δίδαξαν πως το εμείς υπερέχει του εγώ, πως η αλληλεγγύη υπερέχει του ατομικού δρόμου, πως η αισιοδοξία υπερέχει της ηττοπάθειας, πως το 'θα νικήσουμε' υπερέχει του 'θα χάσουμε'.

Τέλος τα μεγαλύτερα ευχαριστώ πάνε στην οικογένεια μου, για όλη τη στήριξη τους σε οποιαδήποτε μου απόφαση, και τις θυσίες που έκαναν για να καταφέρω να βγάλω εις πέρας τις σπουδές μου.

" Μη βαδίζεις μπροστά μου γιατί μπορεί να μην σε ακολουθήσω. Μη βαδίζεις πίσω μου γιατί μπορεί να μη σε οδηγήσω. Βάδιζε πλάι μου και γίνε ο σύντροφός μου" Αλμπέρτ Καμύ

Η διπλωματική αυτή εργασία αφιερώνεται στην οικογένεια μου

Στην αδερφή μου την Μαρία, στην μητέρα μου Γιαννούλα
και στον πατέρα μου Θανάση

Περίληψη

Λόγω της παγκοσμιοποίησης και των διαρκώς αυξανόμενων απαιτήσεων των πελατών η ανάπτυξη της Εφοδιαστικής αλυσίδας αποτελεί καθοριστικό παράγοντα για τη δημιουργία πλεονεκτημάτων στις επιχειρήσεις. Πρωταρχικές δραστηριότητες της εφοδιαστικής αλυσίδας είναι οι μεταφορές και τα αποθέματα οι οποίες απορροφούν μεγάλο μερίδιο του κόστους. Έτσι, ένας από τους στόχους των επιχειρήσεων είναι η μείωση αυτού του κόστους. Στη συγκεκριμένη διπλωματική εργασία θα μας απασχολήσει το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα (Vehicle routing problem with time windows), που σχετίζεται με τη διανομή των προϊόντων μίας εταιρίας σε ένα συγκεκριμένο αριθμό πελατών με συγκεκριμένη ζήτηση για τον κάθε πελάτη, από μία αποθήκη, μέσω συγκεκριμένου αριθμού οχημάτων περιορισμένης χωρητικότητας. Η εξυπηρέτηση του κάθε πελάτη πρέπει να γίνει μία φορά, από ένα όχημα, και το πρόβλημα έχει την ιδιομορφία ότι ο κάθε πελάτης μπορεί να εξυπηρετηθεί μέσα σε ένα συγκεκριμένο χρονικό περιθώριο και όχι πριν ή μετά από αυτό. Στόχος του προβλήματος είναι η ελαχιστοποίηση της ευκλείδειας απόστασης, τηρουμένων των περιορισμών που αναφέρθηκαν παραπάνω. Στο πρώτο μέρος της εργασίας αυτής, χρησιμοποιείται ένας απλός αλγόριθμος για την εύρεση μίας εφικτής υποβέλτιστης λύσης που ικανοποιεί τους περιορισμούς. Επειδή όμως αυτή η λύση δεν είναι η καλύτερη δυνατή, στο δεύτερο μέρος γίνεται προσπάθεια να βελτιστοποιηθεί η λύση αυτή. Για την βελτιστοποίηση της λύσης γίνεται χρήση του μεθευρετικού αλγορίθμου Περιορισμένης Αναζήτησης (Tabu Search Algorithm). Ο αλγόριθμος αυτός χρησιμοποιεί έναν ευρετικό αλγόριθμο για την αναζήτηση καλύτερων αποτελεσμάτων. Επειδή όμως ο ευρετικός αλγόριθμος μπορεί να παγιδευτεί σε τοπικό ελάχιστο, ο μεθευρετικός αλγόριθμος χρησιμοποιεί ορισμένες στρατηγικές για να αποφευχθούν οι επαναλαμβανόμενοι κύκλοι γύρω από μία λύση. Οι στρατηγικές αυτές απαιτούν μνήμη (μία μικρής διάρκειας, μία μεσαίας, και μία μεγάλης διάρκειας). Ουσιαστικά αυτός ο αλγόριθμος αποθηκεύει τις προηγούμενες κινήσεις και απαγορεύει να επαναληφθούν για ένα συγκεκριμένο αριθμό επαναλήψεων, εκτός και αν αποφέρουν καλύτερη λύση οπότε ενεργοποιείται ένα κριτήριο (το κριτήριο της φιλοδοξίας). Ο αλγόριθμος αυτός βελτιώνει την αρχική εφικτή λύση. Τέλος μέσω ενός ακόμα αλγόριθμου ελαχιστοποιείται ο αριθμός οχημάτων. Στην εργασία παρουσιάζεται το πρόβλημα και τα αποτελέσματα από τη χρήση των αλγορίθμων. Για την επίλυση των αλγορίθμων χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον της Matlab.

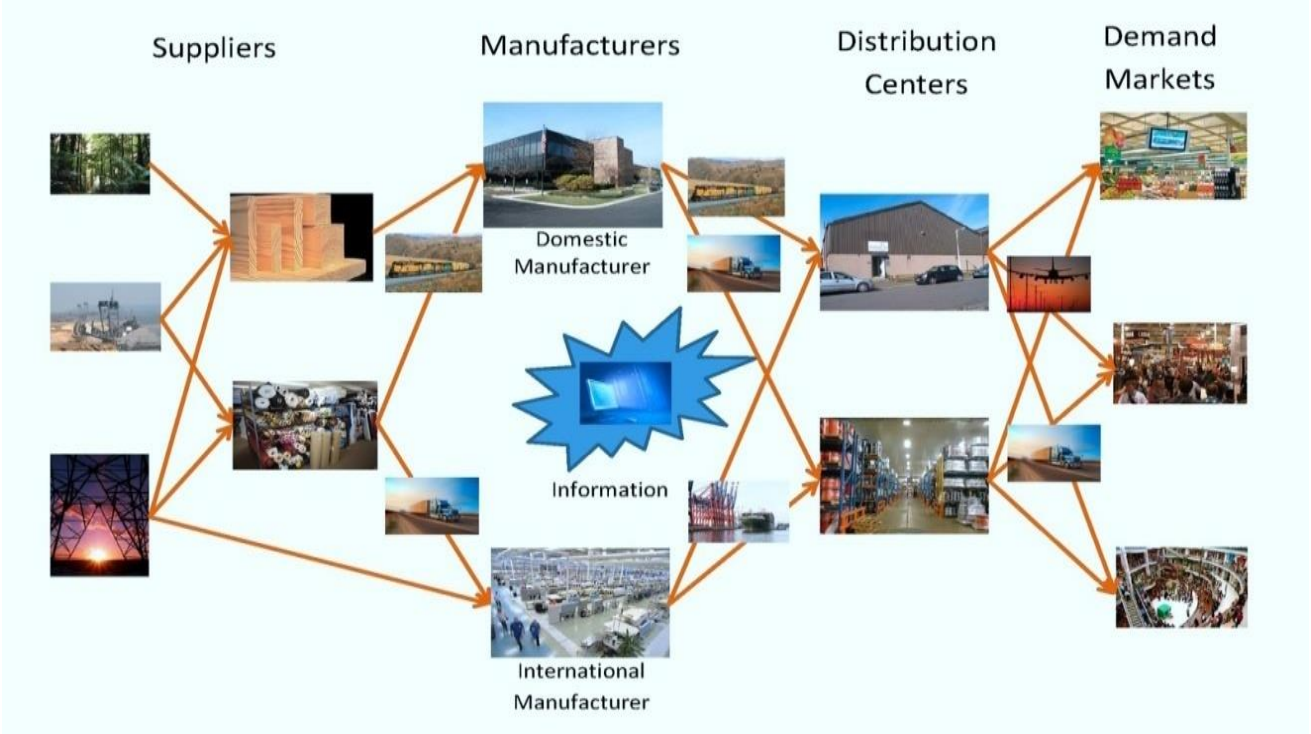
Κεφάλαιο 1.
Εισαγωγή

1.1 Η εφοδιαστική αλυσίδα

Την εποχή της παγκοσμιοποίησης που ζούμε, με την τεχνολογία να έχει φτάσει σε ένα πολύ υψηλό επίπεδο εξέλιξης και τις απαιτήσεις και τις ανάγκες των πελατών να επεκτείνονται διαρκώς, ο ανταγωνισμός μεταξύ των επιχειρήσεων εντείνεται. Το ανταγωνιστικό περιβάλλον που έχει δημιουργηθεί, έχει οδηγήσει τις επιχειρήσεις στην ανακάλυψη νέων πηγών κερδοφορίας όπου μπορούν να αποκτήσουν πλεονέκτημα έναντι άλλων επιχειρήσεων. Τέτοιες πηγές που θα προσδώσουν κερδοφορία, είναι το management, η έρευνα, και η ανάπτυξη μιας πλήρους ανταποκρινόμενης εφοδιαστικής αλυσίδας.

Με τον όρο Εφοδιαστική Αλυσίδα εννοούμε τη ροή υλικών και υπηρεσιών από τον προμηθευτή πρώτων υλών ή κατασκευαστή μέχρι τον καταναλωτή, και παράλληλα την ροή πληροφοριών μεταξύ των μελών της ίδιας αλυσίδας και σύμφωνα με τις ανάγκες των πελατών. Αποτελείται από όλα τα στάδια και όλους τους εμπλεκόμενους από τη στιγμή που προμηθεύονται οι πρώτες ύλες, μέχρι τη στιγμή που το τελικό προϊόν θα φτάσει στον πελάτη η καταναλωτή. Δηλαδή, η εφοδιαστική αλυσίδα αποτελείται από τους κατασκευαστές, και προμηθευτές υλικών, τους χώρους αποθήκευσης, τα κέντρα διανομών, τους μεταφορείς, τους πωλητές, τους πελάτες, τις πρώτες ύλες, αλλά και έτοιμα προϊόντα που μπορεί να υπάρχουν ανάμεσα. Η εφοδιαστική αλυσίδα όμως δεν είναι στατική, αλλά δυναμική και μπορεί να περιλαμβάνει και ενδοεπιχειρησιακές δραστηριότητες ή και μία αλυσίδα επιχειρήσεων, τους προμηθευτές των προμηθευτών και εξωτερικούς συνεργάτες.

Βασική επιδίωξη της εφοδιαστικής αλυσίδας είναι η ικανοποίηση του πελάτη. Όλα τα στάδια και οι ροές προϊόντων υλικών ή πληροφοριών που περιλαμβάνει η εφοδιαστική αλυσίδα δημιουργούν δαπάνες οι οποίες προσαυξάνονται και στο τελικό προϊόν. Μοναδική πηγή εσόδων είναι ο πελάτης, οπότε είναι καθοριστικής σημασίας να μείνει ικανοποιημένος από το τελικό προϊόν που θα φτάσει στα χέρια του. Για αυτό και είναι απαραίτητη η βέλτιστη διαχείριση της εφοδιαστικής αλυσίδας για να παραλάβει το τελικό προϊόν ταχύτερα ο πελάτης, στην καλύτερη τιμή, παρέχοντας του την καλύτερη εξυπηρέτηση και με εμπεριεχόμενες τις νέες τεχνολογίες.



Σχήμα 1.1 Μία εφοδιαστική αλυσίδα (Πηγή: google.com)

Κύριος σκοπός είναι η ελαχιστοποίηση του συνολικού κόστους που προκύπτει εντός της αλυσίδας και σε όλα τα στάδια της, όπως τα έξοδα των μεταφορών, των διανομών κ.λπ. με αποτέλεσμα την τελική αύξηση των κερδών της επιχείρησης.

1.2 Εφοδιαστική (Logistics)

Οι πρώτες ιστορικές αναφορές για τον όρο εφοδιαστική (logistics) όπως τον ξέρουμε σήμερα, ξεκινάνε από την εποχή [4] που οι Αιγύπτιοι κατασκεύασαν τις πυραμίδες και ήταν απαραίτητος ο σωστός σχεδιασμός για την αποθήκευση και μεταφορά της μεγάλης ποσότητας των υλικών που χρησιμοποίησαν. Έκτοτε υπάρχουν πολλές ιστορικές αναφορές, και φτάνουμε στο σήμερα που η εφοδιαστική παίζει καταλυτικό ρόλο για την απόκτηση πλεονεκτήματος των επιχειρήσεων έναντι των ανταγωνιστών τους για την μείωση των δαπανών τους.

Με τον όρο εφοδιαστική εννοούμε [5] τη διαχείριση της ροής αγαθών ή υλικών από την πηγή μέχρι το σημείο κατανάλωσης. Είναι δηλαδή το τμήμα που σχεδιάζει, υλοποιεί και ελέγχει την αποδοτική και αποτελεσματική, κανονική και αντίστροφη, ροή και αποθήκευση των προϊόντων, υπηρεσιών και των πληροφοριών από το σημείο προέλευσης τους έως το σημείο κατανάλωσης τους, με στόχο την ικανοποίηση των απαιτήσεων των πελατών.

Η εφοδιαστική δηλαδή περιέχει τον σχεδιασμό και την υλοποίηση της μεθοδολογίας για την ικανοποίηση του πελάτη. Συνοπτικά, είναι το τμήμα εκείνο που έχει σαν στόχο να φτάσει το επιθυμητό τελικό προϊόν, στην επιθυμητή ποσότητα και ποιότητα, την χρονική στιγμή και σημείο που επιθυμεί ο πελάτης και με μία σωστή τιμή.

1.3 Μεταφορές και αποθέματα

Οι μεταφορές και τα αποθέματα είναι κομβικής σημασίας για την εφοδιαστική αλυσίδα, καθώς αποτελούν τις πιο δαπανηρές δραστηριότητες της. Οι μεταφορές μόνο, συνήθως καταλαμβάνουν το μισό από το συνολικό κόστος της εφοδιαστικής αλυσίδας, ενώ αποτελούν το σύνδεσμο μεταξύ της παραγωγής, της αποθήκευσης και της κατανάλωσης. Εξαιτίας της σημαντικότητας τους, οι επιχειρήσεις δίνουν μεγάλη σημασία στα προβλήματα που σχετίζονται με μεταφορές καθώς δεν χωράει λάθος για την ομαλή της λειτουργία και τον περιορισμό κόστους.

Οι μεταφορές κατηγοριοποιούνται σε αυτές που περιλαμβάνουν τη διακίνηση των πρώτων υλών στους χώρους επεξεργασίας και στα εργοστάσια, και σε αυτές που περιλαμβάνουν την μεταφορά από τα εργοστάσια ή την αποθήκη του τελικού προϊόντος προς τον καταναλωτή. Τα προβλήματα που συναντώνται σε αυτές τις κατηγορίες περιλαμβάνουν το στόλο των οχημάτων, τη δρομολόγηση , το σχεδιασμό δικτύου διανομής, τον χρονοπρογραμματισμό των δρομολογίων, την επιλογή του προσωπικού που θα εκτελέσει τα δρομολόγια, καθώς και υποκατηγορίες αυτών.

Για την επίλυση των προβλημάτων που περιγράφηκαν παραπάνω οι επιχειρήσεις έχουν αναπτύξει διάφορες μεθοδολογίες και αλγορίθμους βελτιστοποίησης. Παρακάτω θα δούμε τις κύριες κατηγορίες προβλημάτων δρομολόγησης καθώς και τρόπους επίλυσης τους.

Κεφάλαιο 2.
Δρομολόγηση οχημάτων

2.1 Το πρόβλημα δρομολόγησης οχημάτων

Όπως αναφέρθηκε και παραπάνω, η παγκοσμιοποίηση της οικονομίας έχει οδηγήσει στην συνεχιζόμενη ανάπτυξη του εμπορίου και των μεταφορών, σε παγκόσμιο επίπεδο. Το υψηλό κόστος των μεταφορών, η πολυπλοκότητα των προβλημάτων οι περιορισμοί που υπάρχουν, καθώς και οι υψηλές απαιτήσεις των πελατών έχουν οδηγήσει τις επιχειρήσεις, στην ανάπτυξη μεθόδων και στρατηγικών επίλυσης, για την βελτίωση του κόστους, των προβλημάτων δρομολόγησης οχημάτων

Τα προβλήματα που έχουν σαν αντικείμενο την δρομολόγηση οχημάτων εντός των επιχειρησιακών δραστηριοτήτων και περιλαμβάνουν τόσο την παράδοση, όσο και την παραλαβή προϊόντων από τους πελάτες, είναι γνωστά ως Προβλήματα Δρομολόγησης οχημάτων (Vehicle Routing Problems). Τα προβλήματα αυτά σχετίζονται με τη διανομή των προϊόντων σε ένα συγκεκριμένο αριθμό πελατών που είναι διασκορπισμένοι γεωγραφικά στο χώρο, από μία ή και περισσότερες αποθήκες, με ένα συγκεκριμένο αριθμό διαθέσιμων οχημάτων και οδηγών για μία δεδομένη χρονική στιγμή και μέσα από ένα συγκεκριμένο οδικό δίκτυο. Σκοπός της επίλυσης του προβλήματος δρομολόγησης οχημάτων, είναι η εύρεση των διαδρομών εκείνων η εκτέλεση των οποίων ελαχιστοποιεί το συνολικό κόστος, που στη συγκεκριμένη περίπτωση είναι η συνολική απόσταση που θα διανύσουν όλα τα οχήματα περνώντας από όλους τους πελάτες. Ταυτόχρονα στόχος είναι η ελαχιστοποίηση του αριθμού των οδηγών και οχημάτων που θα χρησιμοποιηθούν. Η λύση του προβλήματος θα πρέπει να ικανοποιεί όλους τους πιθανούς περιορισμούς που ενδέχεται να υπάρχουν και αξιοποιώντας όλες τις δυνατές πληροφορίες που είναι προς αξιοποίηση. Πιο αναλυτικά οι περιορισμοί από τα χαρακτηριστικά των οχημάτων, των πελατών, των οδηγών και των διαδρομών.

- Μερικά χαρακτηριστικά των οχημάτων
  1)Από ποιά αποθήκη προέρχονται και αν υπάρχει πιθανότητα να τερματίσουν σε μια άλλη αποθήκη
  2)Η χωρητικότητα των οχημάτων
  3)Πόσα τμήματα έχουν τα οχήματα
  4)Αν υπάρχουν οχήματα για εξειδικευμένα προϊόντα
  5)Αν υπάρχουν διαθέσιμα μηχανήματα για την φόρτωση-εκφόρτωση
  6)Δρόμοι που μπορούν να προσπελαστούν από τα οχήματα
  7) Κόστος του κάθε οχήματος

- Μερικά χαρακτηριστικά των πελατών
  1)Το σημείο που βρίσκεται ο πελάτης
  2)Η ποσότητα των αγαθών που πρέπει να παραλάβει ή να παραδώσει ο πελάτης
  3)Τα χρονικά διαστήματα που μπορεί να εξυπηρετηθεί ο πελάτης
  4)Ο χρόνος που απαιτείται για την παράδοση η παραλαβή του προϊόντος από τον πελάτη
  5)Το είδος του οχήματος που απαιτεί η εξυπηρέτηση ενός συγκεκριμένου πελάτη.

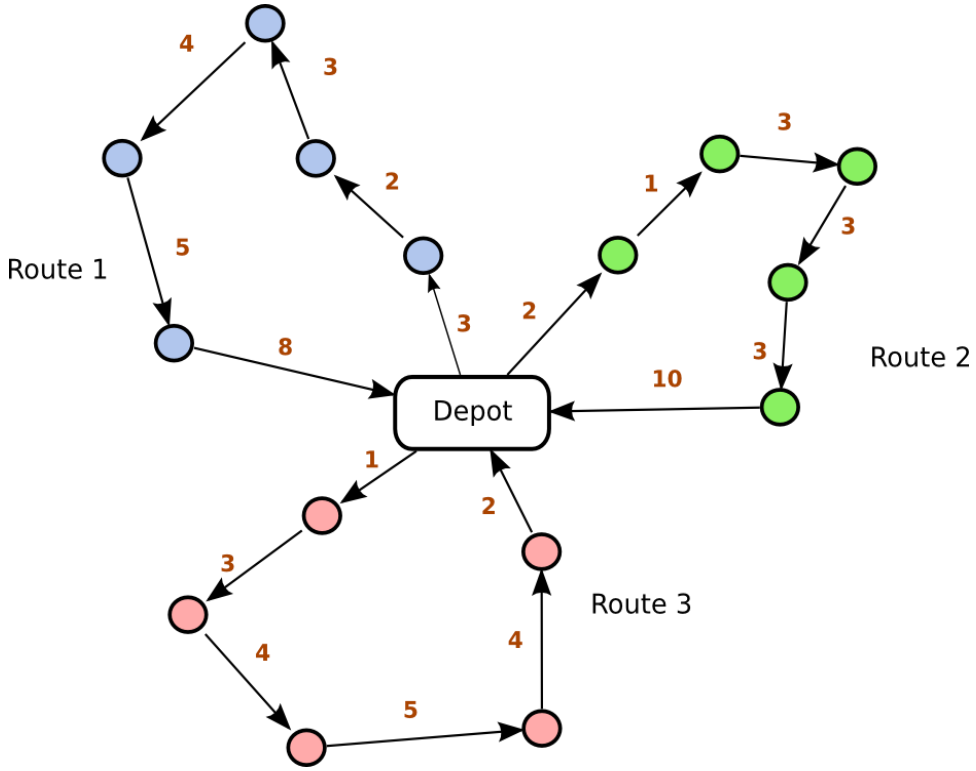
- Μερικά χαρακτηριστικά των οδηγών, κάποια από τα οποία μπορεί να είναι επιβεβλημένα για την καλή υγεία τους και πρέπει να τηρηθούν.
  1)Οκτάωρο ύπνου
  2)Όχι παραπάνω από 10 ώρες συνεχής οδήγησης
  3)Όχι παραπάνω από 6 μέρες
  4)Όχι παραπάνω από 15 ώρες την ημέρα οδήγηση

- Χαρακτηριστικά που αφορούν τις διαδρομές, και μπορεί να σχετίζονται με το προϊόν που διανέμεται ή να εξαρτώνται από τα χαρακτηριστικά των προηγούμενων κατηγοριών.
  1)Η συνολική ποσότητα που μεταφέρει το όχημα δεν πρέπει να ξεπερνάει τη χωρητικότητα του.
  2)Υπάρχουν πελάτες που ζητούν παράδοση ή παραλαβή ή και τα δύο
  3)Τα χρονικά παράθυρα εξυπηρέτησης που θέλουν οι πελάτες
  4)Οι οδηγοί μπορεί να δουλεύουν κάποια συγκεκριμένη χρονική περίοδο.
  5)Τα οχήματα μπορεί να μεταφέρουν παραπάνω από ένα προϊόντα.

Από τα παραπάνω χαρακτηριστικά προκύπτουν και συγκεκριμένοι περιορισμοί που πρέπει να ληφθούν υπ' όψιν κατά την υλοποίηση και την αναζήτηση της βέλτιστης λύσης, ώστε η τελική λύση να είναι εφικτή και πραγματοποιήσιμη. Οι στόχοι ορίζονται από την εκάστοτε επιχείρηση ανάλογα με το τι θέλει να επιτύχει, τις απαιτήσεις των πελατών, σε ποιό αγοραστικό κοινό θέλει να απευθυνθεί και τις προτεραιότητες που η ίδια βάζει. Οι στόχοι λοιπόν μπορεί να είναι:

- Η ελαχιστοποίηση του συνολικού κόστους μεταφοράς
- Η ελαχιστοποίηση των οχημάτων ή και των οδηγών
- Η ισορροπία μεταξύ των διαδρομών που δημιουργούνται, και μπορεί να ζητείται να είναι ίδιος ο αριθμός των πελατών σε κάθε διαδρομή ή ίδιος ο χρόνος της διαδρομής
- Η ελαχιστοποίηση των ποινών μη ικανοποιημένων πελατών
- Η ελαχιστοποίηση της ενέργειας, όπως η βενζίνη
- Η ελαχιστοποίηση του συνολικού χρόνου στις διαδρομές.

Τα προβλήματα δρομολόγησης οχημάτων αναπαρίστανται γραφικά σε ένα δίκτυο G=(N,A), όπου C: {1,…., n} ο αριθμός των πελατών διασκορπισμένους σε διαφορετικές τοποθεσίες N={0,1,...., n} με i=0 να είναι η αποθήκη και για i=1,2…., n οι τοποθεσίες που βρίσκονται οι πελάτες. A είναι το σύνολο των τόξων (i , j) για να μετακινηθούμε από τον πελάτη i στον πελάτη j, ενώ περιέχουν και ένα κόστος c_{ij} που συμβολίζει το κόστος του τόξου για να μετακινηθούμε από τον πελάτη i στον πελάτη j. Υπάρχει ο χρόνος t_{ij} που συμβολίζει τον χρόνο μετάβασης από τον πελάτη i στον πελάτη j. Οι πελάτες αναπαριστώνται στο γράφημα με κόμβους και οι διαδρομές μεταξύ των πελατών, αναπαριστώνται με τόξα. Μία γραφική αναπαράσταση διαδρομών σε ένα πρόβλημα δρομολόγησης οχημάτων φαίνεται στο παρακάτω σχήμα.



Σχήμα 2.1 Πρόβλημα δρομολόγησης οχημάτων (Πηγή: http://neo.lcc.uma.es/dynamic/vrp.html)

Τα προβλήματα δρομολόγησης οχημάτων ανήκουν στην κατηγορία των NP Hard προβλημάτων της συνδυαστικής βελτιστοποίησης της επιχειρησιακής έρευνας [7]. Ανάλογα με τους περιορισμούς που περιγράφηκαν παραπάνω, υπάρχουν διάφορες παραλλαγές και θα γίνει μία αναφορά στα επόμενα κεφάλαια. Στην παρούσα διπλωματική εργασία θα μας απασχολήσει το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα. Για την επίλυση των VRP προβλημάτων έχουν αναπτυχθεί διάφοροι αλγόριθμοι που έχουν σκοπό να βελτιστοποιήσουν το τελικό αποτέλεσμα και θα παρουσιαστούν σε επόμενο κεφάλαιο.

2.2 Πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα (Vehicle Routing Problem with Time Windows)

Το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα (Vehicle Routing Problem with Time Windows) είναι μία παραλλαγή του γενικότερου προβλήματος δρομολόγησης οχημάτων.

Στο συγκεκριμένο πρόβλημα οι κόμβοι (ή πελάτες) έχουν περισσότερα χαρακτηριστικά και η λύση θα πρέπει να ικανοποιήσει περισσότερους περιορισμούς. Σε αυτή την κατηγορία προβλημάτων έχουμε ένα σύνολο πελατών γεωγραφικά διασκορπισμένο σε μία συγκεκριμένη περιοχή. Ο κάθε ένας από αυτούς έχει συγκεκριμένη ποσότητα φορτίου που πρέπει να παραλάβει, και η ιδιαιτερότητα του συγκεκριμένου προβλήματος από τα υπόλοιπα, είναι ότι ο πελάτης μπορεί να εξυπηρετηθεί εντός ενός συγκεκριμένου χρονικού διαστήματος, το οποίο ονομάζεται χρονικό παράθυρο. Πριν ή μετά το χρονικό παράθυρο δεν μπορεί να γίνει η εξυπηρέτηση του πελάτη. Η εξυπηρέτηση των πελατών γίνεται από οχήματα περιορισμένης χωρητικότητας, τα οποία τη χρονική στιγμή t=0 βρίσκονται στην αποθήκη, όπου βρίσκεται το συνολικό φορτίο που πρέπει να διανεμηθεί. Η εξυπηρέτηση του πελάτη πρέπει να γίνει από ένα μόνο φορτηγό μία μόνο φορά και θα πρέπει να παραλάβει όλο το φορτίο. Η αποθήκη έχει επίσης ένα χρονικό παράθυρο, με ενωρίτερο χρόνο τη χρονική στιγμή t=0 και

βραδύτερο, μία χρονική στιγμή που θεωρητικά κλείνει και τα φορτηγά θα πρέπει στο τέλος της διαδρομής να επιστρέψουν στην αποθήκη. Σκοπός του προβλήματος είναι η εύρεση των διαδρομών εξυπηρέτησης πελατών, που ξεκινάνε και τελειώνουν στην αποθήκη χωρίς να παραβιάζονται οι περιορισμοί χρόνου, και χωρητικότητας. Σε κάθε παράδοση το φορτηγό αδειάζει το φορτίο, που σημαίνει πως η ζήτηση του πελάτη δεν πρέπει να υπερβαίνει το συνολικό εναπομείναν διαθέσιμο φορτίο. Σε αντίθετη περίπτωση το φορτηγό δεν θα πραγματοποιήσει την εξυπηρέτηση.

Στο συγκεκριμένο πρόβλημα έχουμε αυστηρά χρονικά παράθυρα που σημαίνει πως ο πελάτης σε καμία περίπτωση δεν μπορεί να εξυπηρετηθεί μετά τον βραδύτερο χρόνο και πριν τον ενωρίτερο χρόνο. Στην περίπτωση που το όχημα φτάσει πριν τον ενωρίτερο χρόνο εξυπηρέτησης, ο οδηγός του φορτηγού θα πρέπει να περιμένει στην τοποθεσία του πελάτη μέχρι τη χρονική στιγμή που μπορεί να ξεκινήσει η εξυπηρέτηση του πελάτη. Στην περίπτωση που το όχημα φτάσει σε έναν πελάτη μετά τον βραδύτερο χρόνο εξυπηρέτησης, τότε δεν μπορεί να πραγματοποιήσει την παράδοση του προϊόντος και θα συνεχίσει την πορεία του. Σε αυτήν την περίπτωση ο πελάτης θα παραλάβει από κάποιο άλλο όχημα. Στην περίπτωση που είχαμε χαλαρά χρονικά παράθυρα η εξυπηρέτηση θα μπορούσε να γίνει ακόμα και αν το φορτηγό έφτανε κάποια χρονική στιγμή εκτός των χρονικών παραθύρων.

Όσον αφορά τις μεταβλητές που προστίθενται στα προβλήματα δρομολόγησης οχημάτων είναι το χρονικό παράθυρο του κάθε πελάτη $[a_i, b_i]$, που όπως αναφέρθηκε η έναρξη εξυπηρέτησης δεν πρέπει να υπερβαίνει το συγκεκριμένο χρονικό διάστημα. Από την στιγμή που το φορτηγό φτάσει στον πελάτη, απαιτείται ένας συγκεκριμένος χρόνος εξυπηρέτησης s_i, που θεωρητικά είναι ο χρόνος που απαιτείται μέχρι το φορτηγό ξεφορτώσει. Για να μεταβεί το φορτηγό από έναν κόμβο i σε έναν κόμβο j απαιτείται χρόνος t_{ij}.

Το πρόβλημα δρομολόγησης με χρονικά παράθυρα μπορεί να αναπαρασταθεί σε ένα γράφημα $G = (V, A)$, όπου η αποθήκη συμβολίζεται με τον κόμβο 0 όταν είναι σημείο εκκίνησης και n+1 όταν είναι σημείο τερματισμού. Δεδομένου ότι η αποθήκη έχει χρονικά παράθυρα, θα ισχύει ότι $\{a_0, b_0\} = \{a_{n+1}, b_{n+1}\} = \{E, L\}$, με τα E και L να είναι η ελάχιστη πιθανή αναχώρηση από την αποθήκη, και η αργότερη δυνατή άφιξη αντίστοιχα. Επίσης η αποθήκη όπως είναι φυσικό έχει μηδενικό χρόνο εξυπηρέτησης και μηδενική ζήτηση δηλαδή $d_0 = d_{n+1} = s_0 = s_{n+1} = 0$. Επίσης έχουμε την χρονική μεταβλητή w_{ik} που δείχνει τη χρονική στιγμή που ξεκινάει η εξυπηρέτηση στον πελάτη i από το όχημα k και την μεταβλητή

$$x_{ijk} = \begin{cases} 0, \text{ εάν το όχημα k επισκέπτεται τον πελάτη j αμέσως μετά τον πελάτη k} \\ 1, \text{ αλλιώς} \end{cases}$$

Η αντικειμενική συνάρτηση γίνεται

$$Z = \min \sum_{i,j} c_{ij} \sum_{k} x_{ijk}$$

Υπό

$$\sum_{k\in K}\sum_{j\in V} x_{ijk} = 1 \qquad \forall i \in N \tag{1}$$

$$\sum_{i\in V-\{0\}} x_{0jk} = 1 \qquad \forall j \in N, k \in K \tag{2}$$

$$\sum_{i\in V-\{j\}} x_{ijk} - \sum_{i\in V-\{j\}} x_{ijk} = 0 \qquad \forall j \in N, k \in K \tag{3}$$

$$\sum_{i\in V-\{n+1\}} x_{in+1k} = 1 \qquad \forall k \in K \tag{4}$$

$$x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \le 0 \qquad \forall k \in K(i,j) \in A \tag{5}$$

$$a_i \sum_{j\in V} x_{ijk} \le w_{ik} \le b_i \sum_{j\in V} x_{ijk} \qquad \forall i \in N, k \in K \tag{6}$$

$$E \le w_{ik} \le L \qquad \forall i \in (0, n+1), k \in K \tag{7}$$

$$\sum_{i\in N} d_i \sum_{j\in V} x_{ijk} \le C \qquad \forall k \in K \tag{8}$$

$$x_{ijk} \ge 0 \qquad \forall k \in K, (i,j) \in A \tag{9}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in A \tag{10}$$

Ας εξηγήσουμε μία τις μεταβλητές, την αντικειμενική και τους περιορισμούς. Οι κόμβοι των πελατών είναι για i=1,…,n με i=1 τον κόμβο αφετηρίας. Το κόστος c_{ij} της διαδρομής από το I στο j και ισχύει ότι $c_{ij}=c_{ji}$ για κάθε $i \neq j$ και $(i,j) \in 1,\ldots.,n$ ισχύει δηλαδή αρχή της συμμετρίας και η τριγωνική ανισότητα $c_{ij} \le c_{ik} + c_{kj}, i \neq j \neq k, (i,j,k) = 1,....,n$.Επίσης όπου d_i η ζήτηση του κάθε πελάτη και C η συνολική ποσότητα προϊόντος. Η αντικειμενική εκφράζει το συνολικό κόστος που θέλουμε να μειώσουμε. Ο (1) περιορίζει τον πελάτη σε ένα μόνο όχημα, οι (2), (3), (4) εκφράζουν τη ροή στο μονοπάτι από το όχημα k. Οι περιορισμοί (5), (7), (8) είναι οι περιορισμοί που αφορούν τα χρονικά περιθώρια και τη χωρητικότητα και ο (6) δίνει την τιμή 0 στη χρονική μεταβλητή w_{ik}, εάν το όχημα δεν επισκέπτεται τους πελάτες i, j σε αυτή την διαδρομή.

Τα προβλήματα δρομολόγησης οχημάτων με χρονικά παράθυρα βρίσκουν εφαρμογή σε πολλές επιχειρήσεις. Για παράδειγμα τα supermarket είναι ανοιχτά ένα συγκεκριμένο ωράριο και μπορούν να εφοδιαστούν από την αποθήκη. Οι μεταφορικές εταιρίες λειτουργούν σε ένα συγκεκριμένο χρονικό περιθώριο που το ορίζουν οι πελάτες. Ακόμα και στα delivery ενός μεγάλου ταχυφαγείου συναντάται, και για να μείνουν ικανοποιημένοι οι πελάτες, θα πρέπει να ληφθεί υπ' όψιν πολύ σοβαρά και το χρονικό παράθυρο που βάζει ο κάθε πελάτης μαζί με τη συνολική απόσταση που θα διανυθεί.

2.3 Άλλα προβλήματα δρομολόγησης οχημάτων

- **Το περιορισμένης χωρητικότητας πρόβλημα δρομολόγησης οχημάτων (Capacited Vehicle Routing Problem)**

Στο πρόβλημα δρομολόγησης οχημάτων με περιορισμένη χωρητικότητα έχουμε μία αποθήκη, ένα στόλο οχημάτων με το κάθε όχημα να έχει συγκεκριμένη χωρητικότητα. Ένας αριθμός πελατών πρέπει να εξυπηρετηθεί με τον κάθε πελάτη να έχει μία συγκεκριμένη ζήτηση. Εδώ υπάρχει ο περιορισμός χωρητικότητας, με την συνολική ζήτηση των πελατών που θα εξυπηρετηθούν από ένα όχημα, να απαγορεύεται να υπερβεί τη συνολική χωρητικότητα του οχήματος. Η εξυπηρέτηση για τον κάθε πελάτη διαρκεί μία συγκεκριμένη χρονική διάρκεια, και το φορτηγό μετά από κάθε εξυπηρέτηση θα πρέπει να έχει ένα χρονικό περιθώριο ικανό για να μπορέσει να επιστρέψει στην αποθήκη. Μετά την εξυπηρέτηση όλων των πελατών που ανήκουν στον κάθε κύκλο, το όχημα επιστρέφει στην αποθήκη. Εδώ στόχος είναι η ελαχιστοποίηση της ευκλείδειας απόστασης που θα διανυθεί συνολικά. Τα χαρακτηριστικά είναι ότι ο κάθε κύκλος περνάει από την αποθήκη, κάθε πελάτης θα δεχθεί επίσκεψη από έναν μόνο κύκλο και το άθροισμα της ζήτησης των κόμβων που επισκέπτονται από έναν κύκλο δεν πρέπει να ξεπερνάει τη συνολική χωρητικότητα του οχήματος.

- **Πρόβλημα Δρομολόγησης οχημάτων με την ύπαρξη πολλών αποθηκών (Multidepot Vehicle Routing).**

Στο συγκεκριμένο πρόβλημα η παραλλαγή είναι ότι διατίθενται παραπάνω από μία αποθήκες για την εξυπηρέτηση των πελατών. Στην περίπτωση που οι πελάτες είναι ομαδοποιημένοι γύρω από κάθε αποθήκη η επίλυση του προβλήματος γίνεται εύκολη, καθώς μπορεί να αντιμετωπιστεί σαν ξεχωριστό VRP . Σε αντίθετη περίπτωση επιλύεται με διαφορετική μεθοδολογία. Σύμφωνα με αυτή το όχημα ξεκινάει από μία αποθήκη και μπορεί είτε να σταματήσει ενδιάμεσα σε άλλη πλησιέστερη αποθήκη για ανεφοδιασμό, είτε να τερματίσει σε μια άλλη. Γίνεται δηλαδή πρόβλημα ομαδοποίησης, αφού σε πρώτη φάση βρίσκονται οι πελάτες που ανήκουν σε κάθε αποθήκη και σε δεύτερη βρίσκονται οι διαδρομές που θα εκτελεστούν. Στόχος του προβλήματος είναι η ελαχιστοποίηση της συνολικής απόστασης που θα διανύσουν τα οχήματα, επιστρέφοντας στο τέλος κάθε διαδρομής σε κάποια αποθήκη.

- **Πρόβλημα δρομολόγησης οχημάτων χωρίς την επιστροφή στην αποθήκη (Open Vehicle Routing Problem)**

Σε όλα τα προηγούμενα προβλήματα μετά την εκτέλεση των δρομολογίων το όχημα έπρεπε να επιστρέψει στην αποθήκη εντός ενός χρονικού περιθωρίου. Σε αυτή την κατηγορία προβλημάτων, το φορτηγό δεν επιστρέφει στην αποθήκη, αλλά συνεχίζει να εκτελεί δρομολόγια μέχρι να αδειάσει το φορτίο του. Αυτή η κατηγορία προβλημάτων συναντάται στην περίπτωση που ο στόλος των φορτηγών δεν ανήκουν στην ιδιοκτησία της επιχείρησης, αλλά συνεργάζεται με μεσάζοντες για την εκτέλεση των δρομολογίων. Έτσι, αντί το φορτηγό να ξοδεύει χρόνο για να γυρίσει στην αποθήκη, τον αξιοποιεί για να εξυπηρετήσει περισσότερους πελάτες. Στόχος είναι η μείωση της συνολικής απόστασης που θα διανυθεί, εξυπηρετώντας όλους τους πελάτες.

- **Στοχαστικά προβλήματα δρομολόγησης οχημάτων (Stochastic or Probabilistic Vehicle Routing Problem)**

Σε αυτά τα προβλήματα ένα όχημα πεπερασμένης χωρητικότητας φεύγει από την αποθήκη και πρέπει να παραδώσει ένα μέρος του φορτίου του σε ένα συγκεκριμένο αριθμό πελατών, των οποίων η ζήτηση γίνεται γνωστή τη χρονική στιγμή που θα φτάσει στον πελάτη. Το όχημα εκτελεί μία διαδρομή και μπορεί να ανεφοδιαστεί όταν χρειάζεται, με τα σημεία στα οποία πραγματοποιούνται οι επιστροφές να είναι στοχαστικά. Το όχημα πρέπει εξυπηρετήσει όλους τους πελάτες διανύοντας την όσο το δυνατόν μικρότερη απόσταση, Η ζήτηση είναι στοχαστική μεταβλητή, με την κατανομή της όμως, να είναι γνωστή.

- **Προβλήματα δρομολόγησης με δυναμική ζήτηση (Dynamic Vehicle Routing Problem)**

Η διαφορά αυτής της κατηγορίας προβλημάτων έγκειται στο γεγονός ότι στα προηγούμενα ήταν γνωστή η γεωγραφική θέση των πελατών από την αρχή, πριν ξεκινήσει η εκτέλεση κάποιου δρομολογίου με τα δεδομένα να παραμένουν ως έχουν γνωστά. Σε αντίθεση, στα προβλήματα δυναμικής ζήτησης μπορεί να προκύψει κάποιος καινούριος πελάτης κατά τη διάρκεια εκτέλεσης ενός δρομολογίου. Αυτή η κατηγορία προβλημάτων συναντάται κυρίως σε εργασίες που σχετίζονται με την παροχή υπηρεσιών, σε δουλειές που ο οδηγός λαμβάνει εντολές σε πραγματικό χρόνο, όπως αυτή του ταξιτζή ή του οδηγού του ασθενοφόρου.

Κεφάλαιο 3.
Αλγόριθμοι εύρεσης βέλτιστης λύσης

3.1 Αλγόριθμοι απληστίας (Greedy Algorithms)

Η ακριβής έννοια του άπληστου αλγόριθμου είναι δύσκολα ορίσιμη. Η λογική τους είναι σχετικά πολύ απλή. Η λύση του αλγορίθμου κατασκευάζεται βηματικά. Αρχικά ξεκινάει από μία μερική λύση, και ελέγχεται ποιά μπορεί να είναι η επόμενη κίνηση που θα πραγματοποιηθεί χωρίς να παραβιαστεί κανένας από τους περιορισμούς που έχουν τεθεί. Η επόμενη κίνηση που θα επιλεχθεί, είναι αυτή που ικανοποιεί τους περιορισμούς και ταυτόχρονα οδηγεί στην βελτίωση του αποτελέσματος συγκριτικά με τις υπόλοιπες πιθανές κινήσεις. Η διαδικασία θα επαναληφθεί μέχρι να δημιουργηθεί η τελική λύση, ή μέχρι να ενεργοποιηθεί κάποιος περιορισμός τερματισμού του προγράμματος.

Στο τέλος δημιουργείται μία τελική εφικτή λύση με ένα σχετικά καλό αποτέλεσμα. Ο αλγόριθμος όμως έχει το μειονέκτημα ότι η κατασκευή της λύσης γίνεται τμηματικά βρίσκοντας την καλύτερη λύση από το σημείο που βρισκόμαστε μία δεδομένη στιγμή και δεν ελέγχονται πολλές άλλες πιθανές λύσεις, οι οποίες μπορεί τμηματικά να μην είναι οι καλύτερες αλλά δίνουν ένα καλύτερο τελικό αποτέλεσμα. Είναι μυωπικοί αλγόριθμοι, δηλαδή βλέπουν μόνο μπροστά και δεν δίνουν τη συνολική εικόνα.

Οι αλγόριθμοι αυτοί μπορεί να δώσουν βέλτιστη λύση μόνο υπό συγκεκριμένες προϋποθέσεις, αλλά στις περισσότερες περιπτώσεις δίνουν ικανοποιητικές αρχικές εφικτές λύσεις. Στα περισσότερα προβλήματα χρησιμοποιούνται για την εύρεση μίας αρχικής λύσης η οποία θα βελτιστοποιηθεί στη συνέχεια με κάποιον άλλον αλγόριθμο. Αυτή η διαδικασία θα ακολουθηθεί και στην παρούσα διπλωματική εργασία, όπως θα δούμε και παρακάτω, όπου θα χρησιμοποιήσουμε τον αλγόριθμο απληστίας για να βρούμε μία αρχική λύση, και στη συνέχεια θα τη βελτιστοποιήσουμε με τη χρήση του αλγορίθμου περιορισμένης αναζήτησης (Tabu Search Algorithm).

3.2 Αλγόριθμος Περιορισμένης Αναζήτησης (Tabu Search Algorithm)

Ο αλγόριθμος της περιορισμένης αναζήτησης είναι μία μέθοδος επίλυσης που συνδυάζει τη διαδικασία τοπικής αναζήτησης και στρατηγική υψηλότερου επιπέδου για την αποφυγή παγίδευσης σε τοπικό ελάχιστο, δηλαδή ανήκει στην κατηγορία των μεθευρετικών αλγορίθμων. Η μέθοδος αρχικά παρουσιάστηκε από τον Glover [13], [14].

Ο αλγόριθμος περιορισμένης αναζήτησης χρησιμοποιεί ένα ευρετικό αλγόριθμο για να μετακινηθεί από την μία λύση στην άλλη. Υπάρχει όμως ο κίνδυνος να εγκλωβιστεί σε κάποιο τοπικό ελάχιστο και να μην είναι δυνατή η εξερεύνηση άλλων περιοχών για τη βελτίωση της λύσης καθώς είναι πιθανό το ενδεχόμενο να πραγματοποιούνται οι ίδιες εναλλαγές. Η φιλοσοφία της στρατηγικής αυτής είναι η χρησιμοποίηση μνήμης για τις προηγούμενες πραγματοποιηθέντες κινήσεις ώστε να ξεφύγει από το τοπικό ελάχιστο. Για να αποφευχθεί η επανάληψη των προηγούμενων λύσεων, οι κινήσεις που

πραγματοποιούνται καταγράφονται σε μία λίστα, η οποία ονομάζεται λίστα περιορισμένων κινήσεων. Οι κινήσεις που θα εισαχθούν στην λίστα απαγορεύεται να επαναληφθούν και έτσι αποφεύγεται η επανάληψη των ίδιων λύσεων. Οι συγκεκριμένες κινήσεις παραμένουν στη λίστα για ένα συγκεκριμένο αριθμό επαναλήψεων που το ορίζει ο χρήστης ή προσαρμόζεται δυναμικά ανάλογα με τις απαιτήσεις του προβλήματος. Με την είσοδο των απαγορευμένων κινήσεων, το αποτέλεσμα της αντικειμενικής συνάρτησης ενδέχεται να χειροτερέψει αρχικά, αλλά έτσι θα αποφευχθεί η κυκλικότητα πράγμα που μπορεί να βελτιώσει αργότερα το αποτέλεσμα. Η ανανέωση της λίστας απαγορευμένων κινήσεων γίνεται δυναμικά και λειτουργεί σύμφωνα με το σύστημα FIFO (First In-First Out). Η διαδικασία αυτή ονομάζεται μνήμη μικρής περιόδου (short term memory). Μία άλλη στρατηγική που μπορεί να προστεθεί, είναι η εισαγωγή περιορισμών που αφορούν τη συχνότητα. Σύμφωνα με αυτή, μετράται ο αριθμός των φορών που μια συγκεκριμένη κίνηση εμφανίζεται κατά τη διάρκεια της αναζήτησης και με βάση τον περιορισμό συχνότητας εμφάνισης μίας κίνησης, γίνεται προσπάθεια να εμποδιστεί η επανάληψη αυτής της κίνησης. Αυτή η διαδικασία ονομάζεται μνήμη μακράς διάρκειας (long term memory), και βοηθάει να εξερευνηθούν κάποιοι αναξιοποίητοι χώροι. Η στρατηγική αυτή, ονομάζεται στρατηγική διάχυσης (diversification). Υπάρχουν όμως και άλλες στρατηγικές. Μία από αυτές έχει στόχο, να παραμείνουν οι μεταβλητές που εμφανίζονται πολύ συχνά, εντός της τελικής λύσης. Η λογική αυτής της στρατηγικής είναι ότι η βέλτιστη λύση θα βρίσκεται κοντά στο υποσχόμενο σημείο, και μπορεί να βρεθεί αν γίνουν κάποιες μικροαλλαγές με αναζήτηση στη γειτονιά των μεταβλητών που κρατάμε. Η στρατηγική αυτή ονομάζεται στρατηγική εντατικοποίησης (intensification strategies) και η μνήμη ονομάζεται μνήμη μεσαίας περιόδου (medium term memory). Μερικές φορές είναι ευνοϊκότερο να αγνοήσουμε τους περιορισμούς. Αυτό μπορεί να γίνει, όταν μία κίνηση που θα πραγματοποιηθεί να επιφέρει καλύτερο αποτέλεσμα στη συνάρτηση κόστους. Σε αυτή την περίπτωση θα αγνοήσουμε τον περιορισμό και θα πραγματοποιήσουμε την κίνηση αφού θα μας επιφέρει καλύτερο αποτέλεσμα. Η αγνόηση αυτή των περιορισμών ονομάζεται κριτήριο απενεργοποίησης των περιορισμών (aspiration cretirion) και λέμε ότι ενεργοποιήθηκε το κριτήριο φιλοδοξίας. Στην περίπτωση που το αποτέλεσμα όλων των κινήσεων επιφέρει μία κατώτερη λύση, τότε θα επιλέξουμε την πρώτη μη απαγορευμένη κίνηση, όπως περιγράφηκε και παραπάνω.

Η περιορισμένη αναζήτηση δεν συγκλίνει με φυσικό τρόπο και το κριτήριο σταματήματος το καθορίζουμε εμείς. Αυτό μπορεί να είναι είτε ένας συγκεκριμένος, μεγάλος αριθμός επαναλήψεων, είτε όταν η διαδικασία επαναληφθεί αρκετές φορές χωρίς να βελτιώνεται η λύση.

Αλγορίθμους περιορισμένης αναζήτησης για τα Vrp προβλήματα, έχουν κατασκευάσει οι Gendreau, Laporte και Potvin (2002), Christofides, Mingozzi και toth (1979), Laporte και Semet (2002), Taillard (1992), Hertz και de Werra (1991), Glover και Laguna (1993,1997), και Gendreau (2003), Osman (1991,1993), Xu και Kelly (1996), Rego και Roucairol (1996), Toth και Vigo (2003). Οι εφαρμογές τους έχουν διαφορετικές στρατηγικές προσεγγίσεις και η κάθε κατασκευή παρουσίασε μία νέα καινοτομία.

Ο ψευδοκώδικας του αλγορίθμου είναι ο ακόλουθος.

Αλγόριθμος περιορισμένης αναζήτησης

Κατασκευή μίας αρχικής λύσης s_0

$s^* = s_0$! Αρχικοποίηση της βέλτιστης λύσης

$k = 0$

$f(s^*) = f(s_0)$

Κύρια Φάση
επανάληψη

Υπολογισμός μιας γειτονικής λύσης s'
Εάν $f(s') < f(s^*)$ **τότε**
$s^* = s'$
$f^* = f(s')$

Τέλος Αν
Αποθήκευσε την τελευταία κίνηση στη λίστα περιορισμένων κινήσεων και ταυτόχρονα αν έχει συμπληρωθεί το μέγεθος της λίστας διέγραψε την παλαιότερη εγγραφή

Κάλεσε κάθε k_1 επαναλήψεις τη στρατηγική εντατικοποίησης
Εάν $f(s_{intensification}) < f(s^*)$ **τότε**
$s^* = s_{intensification}$
$f^* = f(s_{intensification})$
Τέλος Αν

Κάλεσε κάθε k_2 επαναλήψεις τη στρατηγική διαφοροποίησης
Εάν $f(s_{diversification}) < f(s^*)$ **τότε**
$s^* = s_{diversification}$
$f^* = f(s_{diversification})$
Τέλος Αν

Μέχρι κάποιο κριτήριο σταματήματος

Επέστρεψε τη βέλτιστη λύση s*.

3.3 Άλλοι Αλγόριθμοι Βελτιστοποίησης

- **Προσομοιωμένη ανόπτηση (Simulated Annealing)**

Το όνομα του αλγορίθμου προέρχεται από την αναλογία ανάμεσα στην προσομοίωση της ανόπτησης των υλικών και την στρατηγική επίλυσης στα προβλήματα συνδυαστικής βελτιστοποίησης. Η αρχική δημοσίευση έγινε από τον Kirkpatrick S. το 1983 [15]. Η μεθοδολογία του αλγορίθμου είναι η εξής. Ξεκινάμε από μία αρχική λύση, και σε κάθε επανάληψη επιλέγεται μία τυχαία κίνηση. Εάν αυτή η κίνηση βελτιώνει το αποτέλεσμα τότε είναι πάντοτε αποδεκτή. Σε διαφορετική περίπτωση η καινούρια λύση είναι αποδεκτή με κάποια πιθανότητα που προέρχεται από τους νόμους της θερμοδυναμικής και πιο συγκεκριμένα από τον τύπο $p(\delta) = e^{-\frac{\delta}{kt}}$, με k να είναι η σταθερά Boltzmann, t η θερμοκρασία και

δ η ποσότητα της ενέργειας. Αυτό σημαίνει ότι η προσομοιωμένη ανόπτηση αποδέχεται μία μικρή αύξηση στην αντικειμενική συνάρτηση ελέγχοντας την πιθανότητα αποδοχής p(δ), διά μέσου της θερμοκρασίας. Η θερμοκρασία ξεκινάει από μία υψηλή τιμή και μειώνεται σταδιακά. Ο αλγόριθμος περατώνεται στις περιπτώσεις που, η θερμοκρασία έχει πέσει σε ένα αρκετά χαμηλό επίπεδο, έχει περάσει σημαντικός αριθμός επαναλήψεων με μείωση της θερμοκρασία χωρίς τη βελτίωση της λύσης, ένας αριθμός από προκαθορισμένες επαναλήψεις έχει ολοκληρωθεί.

- **Αλγόριθμος αποδοχής κατωφλίου (Threshold Accepting Method)**

Ο αλγόριθμος αποδοχής κατωφλίου είναι ουσιαστικά μία παραλλαγή της μεθόδου προσομοιωμένης ανόπτησης. Η μέθοδος αυτή είναι μία από τις μεθόδους καθορισμένης ανόπτησης. Στην παραλλαγή αυτή δεν χρησιμοποιείται ένα στοχαστικό στοιχείο για την αποδοχή μίας χειρότερης λύσης, αλλά ένα καθορισμένο κατώφλι $T \geq 0$. Μία χειρότερη λύση γίνεται αποδεκτή όταν $c(s') - c(s) \leq T$,όταν δηλαδή ικανοποιεί το κριτήριο αποδοχής της λύσης. Η μέθοδος αυτή εφαρμόστηκε στα προβλήματα δρομολόγησης οχημάτων πρώτη φορά από τους Dueck και Scheurer [16].

- **Γενετικοί Αλγόριθμοι (Genetic Algorithms)**

Οι Γενετικοί Αλγόριθμοι αποτελούν μία μέθοδο αναζήτησης βέλτιστων λύσεων, και είναι χρήσιμοι σε προβλήματα που περιέχουν πολλές παραμέτρους ή διαστάσεις και δεν υπάρχει αναλυτική μέθοδος που να μπορεί να βρει το βέλτιστο συνδυασμό. Οι γενετικοί αλγόριθμοι βασίζονται σε μια μίμηση της βιολογικής διαδικασίας στην οποία αναπτύσσονται νέοι πληθυσμοί μεταξύ διαφορετικών ειδών κατά τη διάρκεια της εξέλιξης, μέσω της διασταύρωσης, της γενετικής μετάλλαξης και της φυσικής επιλογής . Ένας γενετικός αλγόριθμος είναι μία στοχαστική επαναληπτική διαδικασία κατά την διάρκεια των οποίων διατηρεί το μέγεθος του πληθυσμού σταθερό, και ονομάζεται γενιά (generation). Κατά τη διάρκεια της αναζήτησης βέλτιστης λύσης χρησιμοποιούν πληροφορίες από ένα αριθμό λύσεων που ονομάζονται άτομα ή χρωμοσώματα (individuals). Για να δημιουργηθεί η νέα λύση εφαρμόζονται, ένας δυαδικός τελεστής που ονομάζεται διασταύρωση (crossover) και ένας μοναδιαίος που ονομάζεται μετάλλαξη (mutation). Η διασταύρωση παίρνει δύο άτομα που ονομάζονται γονείς (parents) και παράγει δύο νέα που ονομάζονται απόγονοι (offspring). Οι λύσεις που βρίσκονται κοντά στο πιο επιθυμητό αποτέλεσμα, αναπαράγονται σε επόμενη γενιά λύσεων όπου θα δεχθούν μία τυχαία μετάλλαξη. Μετά από έναν αριθμό επαναλήψεων οι τυχαίες μεταλλάξεις σε συνδυασμό με την επιβίωση ή αναπαραγωγή των λύσεων που είναι κοντά στο επιθυμητό αποτέλεσμα, θα παράξουν το τελικό βέλτιστο αποτέλεσμα. Ο γενετικός κώδικας συνήθως αναπαρίσταται με δυαδικά στοιχεία.

- **Αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization)**

Ο αλγόριθμος αυτός είναι εμπνευσμένος από τη φύση και πιο συγκεκριμένα από τον τρόπο με τον οποίο τα μυρμήγκια αναζητούν την τροφή τους, και εκδόθηκε αρχικά από τον Dorigo το 1992 [17]. Αποτελεί αλγόριθμο εύρεσης βέλτιστων μονοπατιών σε ένα γράφημα με σχετικά απλή λογική. Τα μυρμήγκια αναζητούν τυχαία τη τροφή τους γύρω από την πηγή και αφήνουν πίσω μία ποσότητα μιας ουσίας που ονομάζεται φερομόνη για να μαρκάρουν το μονοπάτι που έχουν διανύσει. Η ποσότητα της φερομόνης που αφήνουν εξαρτάται από την ποιότητα και την ποσότητα της τροφής που βρέθηκε καθώς και από την απόσταση. Το επόμενο μυρμήγκι πιθανότατα θα επιλέξει να ακολουθήσει το μονοπάτι με την περισσότερη φερομόνη, και στο ίδιο σημείο θα αφήσει και αυτό ποσότητα φερομόνης. Όσο η φερομόνη αυξάνεται, όλο και περισσότερα μυρμήγκια θα ακολουθήσουν αυτό το

μονοπάτι. Κατά τη διάρκεια του χρόνου όμως η ποσότητα φερομόνης μειώνεται μέχρι που υπάρχει φερομόνη μόνο σε ένα μονοπάτι το οποίο αποτελεί και την τελική λύση. Σε αντιστοιχία, κάθε μυρμήγκι αποτελεί μία λύση για το πρόβλημα. Η διαδικασία είναι επαναληπτική και το επόμενο μυρμήγκι-λύση ξεκινάει να κατασκευάσει μία νέα καλύτερη λύση, λαμβάνοντας υπ' όψιν την εμπειρία που υπάρχει από τα προηγούμενα μυρμήγκια-λύσεις. Ο αλγόριθμος περατώνεται μετά από ένα προκαθορισμένο αριθμό επαναλήψεων.

- **Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization)**

Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων είναι εμπνευσμένος από τη φυσική κίνηση των ατόμων ενός πληθυσμού μέσα σε ένα σμήνος και προσαρμόζεται στις ικανότητες εξερεύνησης των ατόμων εντός ενός σμήνους. Προτάθηκε αρχικά από τους Kennedy και Eberhart [20], και έχει αποδειχτεί πολύ χρήσιμη λόγω της ευκολίας της μεθόδου και της ποιότητας των αποτελεσμάτων. Στην μέθοδο, τα μέλη εντός του σμήνους συνεργάζονται για να βρεθεί η βέλτιστη λύση, και η γνώση από προηγούμενα καλά αποτελέσματα, κληρονομείται και στις επόμενες γενιές. Στον αλγόριθμο, όπου σμήνος είναι ένα σύνολο από πιθανές λύσεις και το κάθε σωματίδιο εντός, αποτελεί μία υποψήφια λύση. Κάθε σωματίδιο κινείται σε ένα χώρο αναζήτησης, όπου η θέση του κάθε στιγμή καθορίζεται από την θέση που βρίσκονται τα υπόλοιπα σωματίδια του σμήνους και από την εμπειρία των προηγούμενων κινήσεων και επαναλήψεων που έχει εκτελέσει το σμήνος. Ο αλγόριθμος περατώνεται μετά από ένα προκαθορισμένο αριθμό επαναλήψεων.

Κεφάλαιο 4.
Περιγραφή και επίλυση του προβλήματος δρομολόγησης οχημάτων

4.1 Περιγραφή και μοντελοποίηση του προβλήματος

Το πρόβλημα που καλούμαστε να επιλύσουμε, είναι το πρόβλημα δρομολόγησης οχημάτων, με χρονικά παράθυρα για τους πελάτες. Στο συγκεκριμένο πρόβλημα διαθέτουμε μία αποθήκη, ένα συγκεκριμένο αριθμό πελατών με συγκεκριμένη ζήτηση, ένα στόλο οχημάτων ίδιας και πεπερασμένης χωρητικότητας το καθένα. Η ιδιαιτερότητα του προβλήματος, είναι ότι όλοι οι πελάτες έχουν ένα συγκεκριμένο χρονικό παράθυρο που μπορούν να δεχθούν εξυπηρέτηση και δεν μπορούν οποιαδήποτε άλλη χρονική στιγμή. Χρονικό παράθυρο έχει και η αποθήκη, με ενωρίτερο χρόνο τη χρονική στιγμή t=0, όπου θεωρητικά είναι η χρονική στιγμή εκκίνησης της λειτουργία της, και καθορισμένο βραδύτερο χρόνο, που είναι η χρονική στιγμή του κλεισίματος της αποθήκης, και θα πρέπει τα φορτηγά να έχουν επιστρέψει εγκαίρως. Στόχος είναι η εξυπηρέτηση όλων των πελατών, ικανοποιώντας τους περιορισμούς και ταυτόχρονα η αναζήτηση της βέλτιστης λύσης, ώστε τα φορτηγά να διανύσουν την όσο το δυνατόν μικρότερη ευκλείδεια απόσταση. Σε πρώτη φάση, θα βρεθεί μία αρχική υποβέλτιστη εφικτή λύση που ικανοποιεί όλους τους περιορισμούς και στη συνέχεια έχοντας ως δεδομένη αυτή την αρχική λύση, καλούμαστε να τη βελτιστοποιήσουμε, με τη χρήση του μεθευρετικού αλγορίθμου περιορισμένης αναζήτησης (Tabu Search Algorithm). Οι αλγόριθμοι επιλύθηκαν στο προγραμματιστικό περιβάλλον της Matlab έχοντας γνωστά δεδομένα εισόδου.

Αρχικά, πρέπει να εξετάσουμε το πρόβλημα και να δούμε ποιές μεταβλητές πρέπει να αρχικοποιήσουμε και ποιές χρειάζεται να κατασκευάσουμε στη συνέχεια. Ένα από τα δεδομένα που έχουμε, είναι η αρίθμηση των πελατών, με τον πελάτη 0 να είναι η αποθήκη. Επειδή χρησιμοποιούμε το προγραμματιστικό περιβάλλον της Matlab, η αποθήκη θα αναπαριστάται με τον αριθμό 1, τον πελάτη 1 να αναπαριστάται με το 2, κ.ο.κ. Τα υπόλοιπα δεδομένα είναι, οι συντεταγμένες x και y του κάθε πελάτη, η ζήτηση του κάθε πελάτη, τα χρονικά παράθυρα ενωρίτερων και βραδύτερων χρόνων, η χωρητικότητα των οχημάτων, ο χρόνος εξυπηρέτησης για τον κάθε πελάτη, ο μέγιστος αριθμός διαθέσιμων οχημάτων. Τα δεδομένα αυτά αποθηκεύτηκαν σε φύλλα excel, ως .csv αρχεία, και διαβάστηκαν από την matlab με την εντολή csvread. Για παράδειγμα η ζήτηση αποθηκεύτηκε στο μονοδιάστατο πίνακα ‘dem’ και η εντολή ήταν dem=csvread(‘zitisi.csv’), με το zitisi.csv να είναι το όνομα του αρχείου excel. Αντίστοιχα αποθηκεύτηκαν και τα άλλα δεδομένα σε αντίστοιχες μεταβλητές. Αναλυτικά τα δεδομένα παρουσιάζονται παρακάτω:

customers: ο αριθμός των πελατών

dem: μονοδιάστατος πίνακας ζήτησης του κάθε πελάτη

Rt: μονοδιάστατος πίνακας ελάχιστου χρονικού παραθύρου για τον κάθε πελάτη

Dt: μονοδιάστατος πίνακας μέγιστου χρονικού παραθύρου για τον κάθε πελάτη

service_time: μονοδιάστατος πίνακας χρόνου εξυπηρέτησης για τον κάθε πελάτη

X: μονοδιάστατος πίνακας τετμημένης του κάθε πελάτη

Y: μονοδιάστατος πίνακας τεταγμένης του κάθε πελάτη

max_vehicle_number: ο μέγιστος διαθέσιμος αριθμός οχημάτων

capacity: Η συνολική διαθέσιμη χωρητικότητα του κάθε οχήματος

Όπως αναφέρθηκε παραπάνω, ο κόμβος 1 είναι η αποθήκη, οπότε από τα δεδομένα θα έχουμε dem(1)=0= Rt(1)=service_time(1). Συνεπώς ο μέγιστος χρόνος που μπορεί να διαρκέσει μία διαδρομή υποδηλώνεται από το βραδύτερο χρόνο της αποθήκης, συνεπώς

max_service_time= Dt(1)

Στη συνέχεια θα πρέπει να κατασκευάσουμε κάποιους επιπλέον πίνακες και μεταβλητές που μας είναι απαραίτητοι σε όλο τη διαδικασία επίλυσης. Ο πρώτος πίνακας που χρειαζόμαστε, είναι αυτός που θα υπολογίσει τις αποστάσεις μεταξύ όλων των κόμβων. Ο πίνακας αυτός θα συμβολίζεται ως A(i,j), όπου i ο προηγούμενος πελάτης και j ο επόμενος. Η απόσταση που υπολογίζεται είναι η ευκλείδεια, ο πίνακας είναι συμμετρικός που σημαίνει ότι η απόσταση που διανύουμε για να πάμε από έναν κόμβο i σε έναν επόμενο κόμβο j, ισούται με την απόσταση για να πάμε από τον κόμβο j στον κόμβο i. Ένα άλλο στοιχείο που θα πρέπει να λάβει ο πίνακας είναι ότι απαγορεύεται η παραμονή σε έναν κόμβο, και θα πρέπει να μετακινηθούμε από έναν κόμβο σε έναν άλλον. Για αυτό το λόγο η απόσταση A(i,i) απειρίζεται για να μη ληφθεί η συγκεκριμένη απόσταση υπ' όψιν στην επίλυση του προβλήματος. Δεδομένου ότι ο πίνακας A εμφανίζει όλες τις αποστάσεις μεταξύ των κόμβων, θα είναι διαστάσεων A(customers+1, customers+1), δηλαδή όσοι είναι οι πελάτες και μαζί η αποθήκη. Ο πίνακας κατασκευάστηκε ως εξής.

$$A(i,j) = \sqrt{((X(j) - X(i))^2 + (Y(j) - Y(i))^2)}$$

Τώρα έχουμε όλες τις αποστάσεις που χρειάζεται να διανύσουμε για να μεταβούμε από τον προηγούμενο στον επόμενο κόμβο. Σε αυτό το σημείο να αναφέρουμε ότι στα προβλήματα που θα συναντήσουμε, ο χρόνος για να μεταβούμε από τον πελάτη i στον πελάτη j, ισούται με την απόσταση που θα διανύσουμε. Άρα t(i,j) = A(i,j). Στην επίλυση θα χρησιμοποιούμε τα δεδομένα του πίνακα A, όταν θα θέλουμε να αναφερθούμε σε χρόνο.

Τώρα έχουμε όλους πίνακες και όλες τις μεταβλητές βάσει των δεδομένων που μας δόθηκαν. Στη συνέχεια του προβλήματος θα χρειαστούν και άλλοι πίνακες και μεταβλητές που θα περιγραφούν στη συνέχεια.

4.2 Εύρεση αρχικής εφικτής λύσης με τον αλγόριθμο απληστίας

Έχοντας αποθηκεύσει τα δεδομένα του προβλήματος, το επόμενο βήμα είναι να ξεκινήσουμε την επίλυση του. Στην πρώτη φάση της επίλυσης, χρησιμοποιούμε και προσαρμόζουμε αναλόγως τον αλγόριθμο απληστίας, για την εύρεση μίας αρχικής εφικτής λύσης. Το πρώτο βήμα στον αλγόριθμο, είναι η αρχικοποίηση των μεταβλητών που θα χρειαστούμε για την επίλυση του προβλήματος. Αρχικά, δηλώνονται οι μεταβλητές που δηλώνουν την αρχική κατάσταση που επικρατεί πριν ξεκινήσει η διαδικασία επίλυσης του προβλήματος. Τη χρονική στιγμή t=0 γνωρίζουμε ότι τα φορτηγά βρίσκονται στην αποθήκη, ετοιμάζεται να ξεκινήσει το πρώτο όχημα, δεν έχει εξυπηρετηθεί κανένας πελάτης και το φορτηγό που θα εκτελέσει το πρώτο δρομολόγιο έχει πλήρη διαθέσιμη χωρητικότητα. Επομένως ορίζουμε τις μεταβλητές

V=1: ο αριθμός των χρησιμοποιούμενων οχημάτων, ορίζεται ως 1 γιατί αναφερόμαστε στη χρονική στιγμή που θα ξεκινήσει το πρώτο όχημα

At=0: ο χρόνος άφιξης του φορτηγού στον πελάτη, που ορίζεται ως 0 επειδή ξεκινάμε από τη χρονική στιγμή 0

cap=capacity: το συνολικό εναπομείναν φορτίο που έχει το φορτηγό. Τη χρονική στιγμή 0 είναι γεμάτο εφόσον δεν έχει εξυπηρετήσει κάποιον πελάτη και θα μειώνεται κατά την εξυπηρέτηση των πελατών.

N=customers: η μεταβλητή αυτή μας δείχνει πόσοι πελάτες περιμένουν να εξυπηρετηθούν. Τη χρονική στιγμή 0, δεν έχει εξυπηρετηθεί κάποιος πελάτης. Ο αλγόριθμος θα τερματιστεί όταν εξυπηρετηθούν όλοι οι πελάτες, δηλαδή N=0. Κάθε φορά που βρίσκεται ο επόμενος εφικτός πελάτης η μεταβλητή θα μειώνεται κατά 1 μονάδα.

pnode=1: μεταβλητή που δείχνει σε ποιόν πελάτη βρισκόμαστε. Τη χρονική στιγμή 0 βρισκόμαστε στην αποθήκη

cost=0: το ολικό κόστος, η μεταβλητή που θέλουμε να ελαχιστοποιήσουμε

Επίσης θα ορίσουμε ένα διάνυσμα y το οποίο θα αρχικοποιηθεί ως μηδενικό διάνυσμα και θα αποτυπώνει ποιοί πελάτες έχουν εξυπηρετηθεί. Όταν εξυπηρετηθεί κάποιος πελάτης θα πάρει την τιμή 1 στην αντίστοιχη θέση, για παράδειγμα όταν εξυπηρετηθεί ο $55^{ος}$ πελάτης το y(55) = 1. Όταν τελειώσει ο αλγόριθμος το διάνυσμα αυτό θα έχει μόνο την τιμή 1. Θα μας βοηθήσει για την αποφυγή επανεξυπηρέτησης κάποιου πελάτη. Έτσι το διάνυσμα τη χρονική στιγμή 0 έχει μηδενικά, εκτός από το y(1)=1, επειδή δεν θέλουμε να ληφθεί προς εξέταση η αποθήκη.

Στη συνέχεια, θα πρέπει να κατασκευαστούν δύο επιπλέον πίνακες, εντός της αλγοριθμικής επαναληπτικής διαδικασίας. Θα χρειαστεί ένας πίνακας διαδρομών και ένας ακόμη πίνακας χρονικών αφίξεων. Στις διαστάσεις των δύο πινάκων θα ισούνται, οι γραμμές με τον αριθμό των διαδρομών, και οι στήλες με την ποσότητα των πελατών που εξυπηρετεί η κάθε διαδρομή. Το πρώτο και το τελευταίο στοιχείο αντιστοιχούν στην αποθήκη. Ο πίνακας διαδρομών θα δηλωθεί με το όνομα routes. Το πρώτο στοιχείο κάθε γραμμής παίρνει την τιμή 1, αφού το σημείο εκκίνησης κάθε διαδρομής είναι η αποθήκη. Όταν εξυπηρετήσει ένα αριθμό πελατών, θα πρέπει να επιστρέψει στην αποθήκη, που σημαίνει ότι, το τελευταίο στοιχείο της κάθε γραμμής θα παίρνει τιμή ίση με τη μονάδα. Όσον αφορά

τον πίνακα αφίξεων, το πρώτο στοιχείο της κάθε γραμμής θα πάρει την τιμή μηδέν καθώς εκείνη τη χρονική στιγμή το όχημα βρίσκεται στην αποθήκη. Τα επόμενα στοιχεία παίρνουν την τιμή που αντιστοιχεί στη χρονική στιγμή στην οποία το φορτηγό φτάνει στον πελάτη. Το τελευταίο στοιχείο, που αντιστοιχεί στην τιμή 1 στον πίνακα των διαδρομών, παίρνει την τιμή της χρονικής στιγμής κατά την οποία το όχημα επιστρέφει στην αποθήκη. Για να ικανοποιηθεί ο περιορισμός, η τιμή πρέπει να είναι μικρότερη από την τιμή της μεταβλητής max_service_time. Συνοπτικά οι πίνακες που θα κατασκευαστούν.

routes: πίνακας διαδρομών, δείχνει την πορεία που εκτελεί το κάθε φορτηγό

Arriving_time: πίνακας αφίξεων των φορτηγών στους πελάτες

Αυτοί οι πίνακες θα μας χρειαστούν και στο δεύτερο μέρος του προβλήματος, και η βελτιστοποίηση του αποτελέσματος θα γίνει με αλλαγές πάνω σε αυτούς.

Ξεκινώντας το κύριο μέρος του προγράμματος, το σημείο εκκίνησης τη χρονική στιγμή t=0 είναι η αποθήκη. Δεδομένου, ότι χρησιμοποιούμε τον αλγόριθμο απληστίας αναζητούμε την βέλτιστη λύση τοπικά, σε κάθε βήμα. Επομένως το επόμενο βήμα μας είναι να αναζητήσουμε τον επόμενο ‘καλύτερο’ πελάτη που θέλουμε να επισκεφθούμε. Για να το κάνουμε αυτό, θα κατασκευάσουμε δύο βοηθητικούς πίνακες, τον J και τον NN1. Ο J πίνακας είναι μονοδιάστατος και θα περιέχει το άθροισμα της απόστασης και του μέσου χρονικού παραθύρου από τον πελάτη που βρισκόμαστε πολλαπλασιασμένα με βάρη. Για την εύρεση των βαρών εκτελέστηκαν μία σειρά από επαναλήψεις του τελικού προγράμματος, και επιλέχτηκαν αυτά που έδιναν το καλύτερο τελικό αποτέλεσμα. Ο πίνακας NN1, είναι και αυτό διάνυσμα και περιέχει την αρίθμηση των πελατών κατά αύξουσα σειρά από τον 1 έως το customers+1, και θα μας χρειαστεί στη συνέχεια.

Ο πίνακας J επομένως, κατασκευάζεται ως εξής:

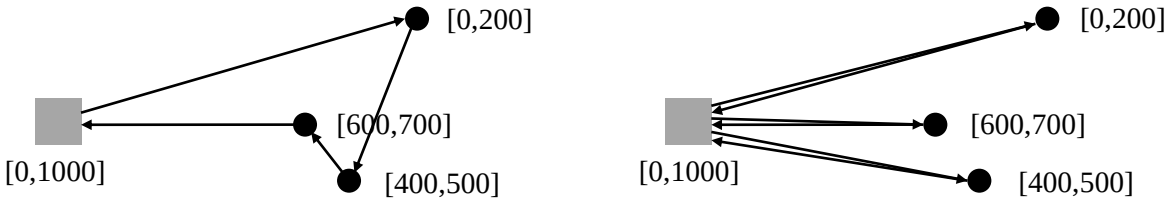
$$J(i) = K * A(pnode, i) + \frac{L*(Dt(i) + Rt(i))}{2}$$

Για τον πελάτη που έχει ήδη εξυπηρετηθεί το αντίστοιχο J θα απειριστεί, δηλαδή για το i στο οποίο y(i) = 1, J(i) = inf. Οι τιμές K, L είναι τα βάρη που όπως αναφέρθηκε επιλέγονται είτε αυθαίρετα είτε με δοκιμές και επαναλήψεις.

Στη συνέχεια εφόσον έχει κατασκευαστεί το διάνυσμα J, θα ταξινομηθεί κατά αύξουσα σειρά, δηλαδή τα στοιχεία που έχουν το μικρότερο άθροισμα θα είναι στις πρώτες θέσεις και αυτά που έχουν το μεγαλύτερο άθροισμα, θα είναι στις τελευταίες θέσεις του διανύσματος. Ταυτόχρονα όμως με τον πίνακα J, θα ταξινομηθεί και ο πίνακας NN1 που κατασκευάστηκε προηγουμένως, και έτσι θα γνωρίζουμε κατά αύξουσα σειρά τους πιο ωφέλιμους πελάτες, για να πάμε. Έτσι ο πελάτης NN1(1), είναι θεωρητικά ο καλύτερος που μπορούμε να επισκεφτούμε βάσει του συνυπολογισμού απόστασης και χρονικού παράθυρου, ο NN(2) ο δεύτερος κ.ο.κ.

Με αυτή τη τεχνική μπορεί να μην επιλεγεί ο πελάτης που είναι πιο κοντά και φαινομενικά μπορεί να δώσει άμεσα μικρότερο κόστος, αλλά αυτός που σε έναν πιο μακρινό ορίζοντα μπορεί να μας βοηθήσει να βρούμε καλύτερο τελικό κόστος και να χρησιμοποιήσουμε λιγότερα οχήματα. Στο

παρακάτω σχήμα φαίνεται σε τι μας ωφελεί η μέθοδος. Στην πρώτη περίπτωση που συνυπολογίζεται η απόσταση χρησιμοποιούμε ένα όχημα, ενώ στη δεύτερη που δεν συνυπολογίζεται χρησιμοποιούμε τρία.



Σχήμα 4.1 Αριστερά η περίπτωση που συνυπολογίζεται η απόσταση με το χρονικό παράθυρο για την εύρεση του πιο ωφέλιμου πελάτη, ενώ στη δεύτερη λαμβάνεται υπ' όψιν μόνο η απόσταση.

Θεωρούμε ότι ο καλύτερος πελάτης που μπορεί να επιλεχθεί είναι αυτός που βρίσκεται στη θέση 1 στον πίνακα NN1, γιατί στην αντίστοιχη θέση ο πίνακας J έχει την μικρότερη τιμή. Άρα επιλέγουμε, ο επόμενος κόμβος που θα επισκεφθούμε είναι ο πελάτης NN1(1). Η χρονική στιγμή που θα γίνει η άφιξη σε αυτόν τον πελάτη, είναι η χρονική στιγμή που ήμασταν στον προηγούμενο πελάτη προσθέτοντας το χρόνο εξυπηρέτησης του προηγούμενου και το χρόνο που απαιτείται για να μεταβούμε από τον προηγούμενο στον επόμενο πελάτη. Συνεπώς θα έχουμε

$$node = NN1(1)$$
$$At = Arriving\_time(V, metritis1) + service\_time(pnode) + A(pnode, node)$$

Όπως αναφέραμε παραπάνω τον πίνακα Arriving_time τον κατασκευάζουμε κατά τη διάρκεια του αλγορίθμου. Οπότε ο metritis1, είναι μία μεταβλητή που χρησιμοποιούμε για να ξέρουμε σε ποιά στήλη βρισκόμαστε κάθε στιγμή. Επομένως τη δεδομένη στιγμή βρισκόμαστε στο Arriving_time(V,metritis1) και θέλουμε να μεταβούμε στο Arriving_time(V,metritis1+1).

Γνωρίζουμε τον πελάτη που φαινομενικά είναι αυτός που δίνει το καλύτερο αποτέλεσμα αν τον επισκεφθούμε, δεν γνωρίζουμε ακόμα όμως αν είναι εφικτή η επίσκεψη στον πελάτη, δηλαδή αν ικανοποιούνται οι περιορισμοί του προβλήματος, κάτι που θα εξεταστεί στη συνέχεια. Οι περιορισμοί που πρέπει να ελεγχθούν είναι

- Αν ο χρόνος που θα φτάσουμε τον πελάτη είναι μικρότερος από το βραδύτερο χρόνο που μπορεί να δεχθεί επίσκεψη ο πελάτης, δηλαδή $At \le Dt(node)$
- Ο πελάτης δεν πρέπει να έχει εξυπηρετηθεί, δηλαδή $y(node) \ne 1$
- Η ποσότητα προϊόντος θα πρέπει να ικανοποιεί τη ζήτηση του πελάτη, δηλαδή $cap \ge dem(node)$
- Το φορτηγό στη συνέχεια θα πρέπει να μπορεί να επιστρέψει στην αποθήκη χωρίς να υπερβεί το βραδύτερο χρόνο της αποθήκης, δηλαδή $At + service\_time(node) + A(node,1) < \max\_service\_time$

Εάν ένα από αυτά δεν συμβαίνει και δεν ικανοποιείται κάποιος περιορισμός, τότε θα επλεχθεί ο επόμενος πιο ωφέλιμος πελάτης από τον πίνακα NN1, και θα ανανεωθούν οι μεταβλητές node και At. Σε περίπτωση που ελεγχθεί όλη η λίστα αλλά δε βρεθεί διαθέσιμος πελάτης τότε σημαίνει πως δεν

υπάρχει κάποιος πελάτης που πληροί τα κριτήρια και θα γίνει επιστροφή στην αποθήκη. Όταν επιστρέψουμε στην αποθήκη θα σταλθεί νέο φορτηγό τη χρονική στιγμή 0, με νέο διαθέσιμο φορτίο, θα ξεκινήσει η αναζήτηση νέου πελάτη από την αποθήκη, και θα επαναληφθεί η ίδια διαδικασία, δηλαδή

pnode = 1

V = V + 1, εφόσον στέλνουμε νέο όχημα

metritis1 = 1, εφόσον ξεκινάμε από νέα γραμμή

cap = capacity

At = 0

Στην περίπτωση όμως που ικανοποιούνται οι περιορισμοί που αναφέρθηκαν παραπάνω, έχουμε βρει τον επόμενο κόμβο που θα μετακινηθούμε και θα πραγματοποιήσουμε όλες τις αλλαγές στις μεταβλητές για τη συνέχιση του προβλήματος.

metritis1 = metritis1 +1, μετακινούμαστε στους πίνακες Arriving_time, routes μία θέση δεξιά

Σε αυτό το σημείο θα συναντήσουμε έναν επιπλέον περιορισμό, που σχετίζεται με το χρόνο άφιξης. Αυτό, σημαίνει ότι:

- Εάν το φορτηγό φτάσει πριν τον ενωρίτερο χρόνο εξυπηρέτησης του πελάτη, τότε ο οδηγός θα περιμένει στην ίδια τοποθεσία και η εξυπηρέτηση θα ξεκινήσει τη χρονική στιγμή που μπορεί να εξυπηρετηθεί ο πελάτης. Αυτό μεταφράζεται ως $Arriving\_time(V, metritis1) = \max(Rt(node), At)$, με το At, να περιλαμβάνει το χρόνο άφιξης του προηγούμενου πελάτη + το χρόνο εξυπηρέτησης του + το χρόνο μέχρι να μεταβεί στον πελάτη που βρισκόμαστε τώρα.

Υπολογίζουμε το νέο κόστος

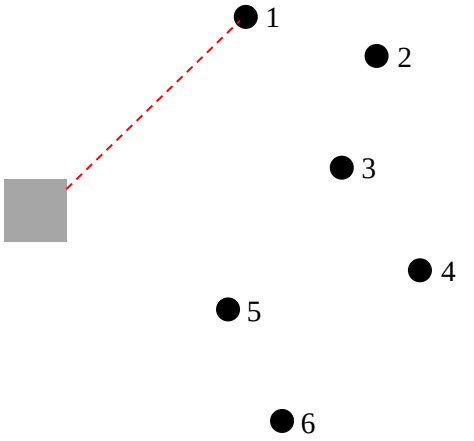
$$cost = cost + A(pnode, node)$$

Ανάλογα ανανεώνονται και οι άλλες μεταβλητές και πίνακες,

$$routes(V, metritis1) = node$$
$$cap = cap - dem(node)$$
$$pnode = node$$
$$y(node) = 1$$
$$N = N - 1$$

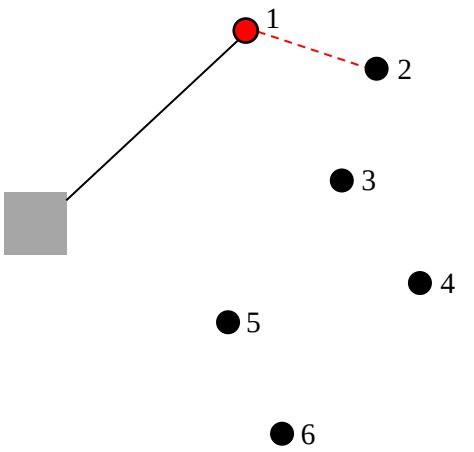
Εφόσον ανανεωθούν οι μεταβλητές συνεχίζεται η αναζήτηση του επόμενου πελάτη που θα επισκεφθούμε με την ίδια επαναληπτική διαδικασία. Ο αλγόριθμος περατώνεται όταν ο αριθμός των πελατών που απομένουν να εξυπηρετηθούν μηδενιστεί, δηλαδή όταν η μεταβλητή N πάρει την τιμή 0. Στις επόμενες σελίδες απεικονίζεται γραφικά η διαδικασία.

Βήμα 0. Βρισκόμαστε στην αποθήκη και επιλέγουμε τον πιο ωφέλιμο πελάτη βάση απόστασης και χρονικού παραθύρου. Έστω ότι αυτός είναι ο κόμβος 1.



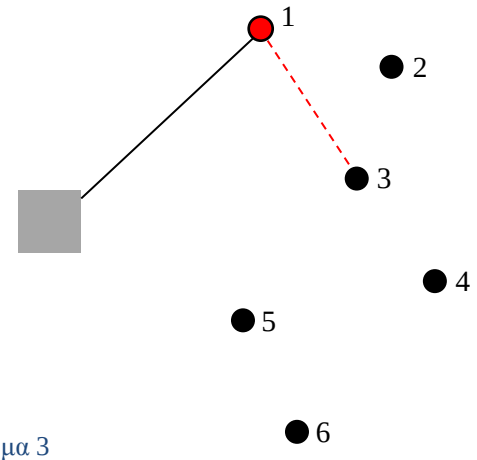
Σχήμα 1

Βήμα 1. Ελέγχουμε αν ο πελάτης ικανοποιεί τους περιορισμούς χωρητικότητας και χρονικού παραθύρου. Έστω ότι τους ικανοποιεί. Στη συνέχεια επιλέγουμε από τον πελάτη που βρισκόμαστε τον επόμενο, και έστω ότι αυτός είναι ο 2. Ελέγχουμε αν ικανοποιεί τους περιορισμούς.



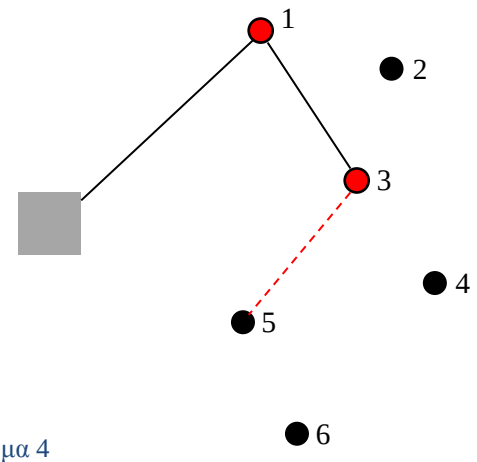
Σχήμα 2

Βήμα 2. Έγινε ο έλεγχος των περιορισμών και έστω ότι δεν ικανοποιείται ο περιορισμός βραδύτερου χρονικού παραθύρου. Ο επόμενος πιο ωφέλιμος πελάτης είναι ο 3. Ελέγχουμε τους περιορισμούς για τον πελάτη 3.



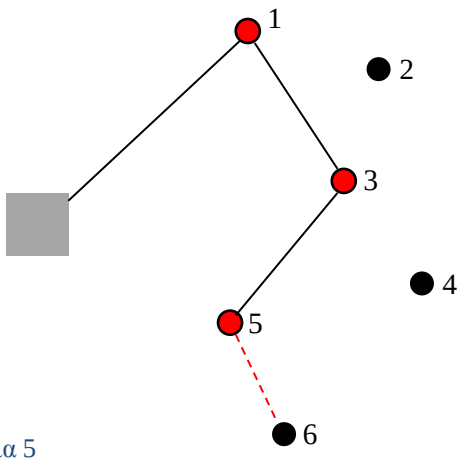
Σχήμα 3

Βήμα 3. Ο πελάτης 3 ικανοποιεί τους περιορισμούς φορτίου και χρονικού παραθύρου, οπότε επιλέγεται. Από τον 3 αναζητούμε τον επόμενο καλύτερο προς εξυπηρέτηση πελάτη και έστω αυτός πως είναι ο 5. Ελέγχουμε τους περιορισμούς.



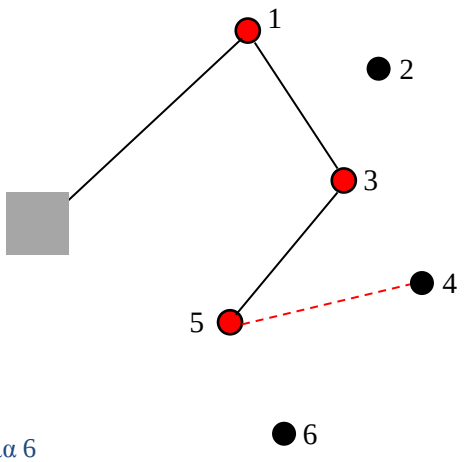
Σχήμα 4

Βήμα 4. Ο πελάτης 5 ικανοποιεί τους περιορισμούς φορτίου και χρονικού παραθύρου, οπότε επιλέγεται. Από τον 5 αναζητούμε τον επόμενο καλύτερο προς εξυπηρέτηση πελάτη και έστω αυτός πως είναι ο 6. Ελέγχουμε τους περιορισμούς.



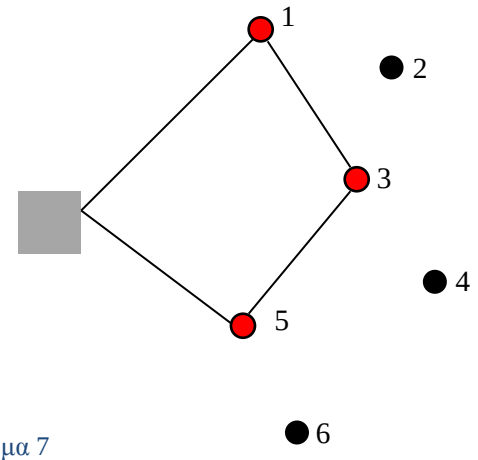
Σχήμα 5

Βήμα 5. Η μετάβαση στον πελάτη δεν ικανοποιεί τον περιορισμό φορτίου. Ελέγχεται η μετάβαση στον πελάτη 4, ο 2 απορρίπτεται ήδη επειδή δεν ικανοποιεί τον περιορισμό χρονικού παραθύρου.



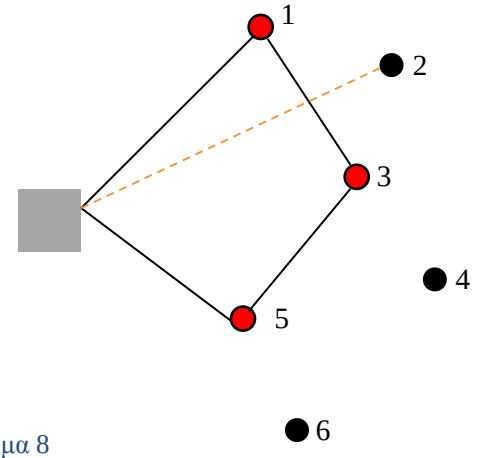
Σχήμα 6

Βήμα 6. Η μετάβαση στον πελάτη 4 δεν ικανοποιεί τον περιορισμό φορτίου. Δεν υπάρχει κανένας διαθέσιμος προς επίσκεψη, κόμβος. Επιστρέφουμε στην αποθήκη.



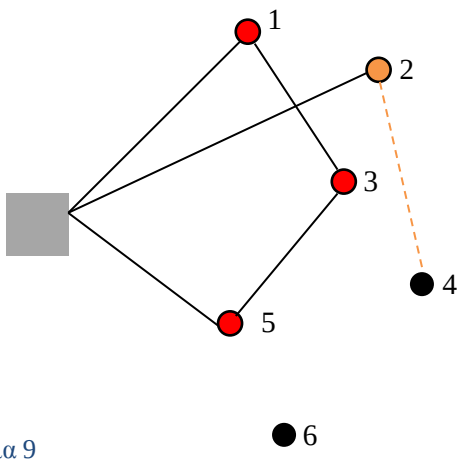
Σχήμα 7

Βήμα 7. Από την αποθήκη θα ξεκινήσει νέο φορτηγό τη χρονική στιγμή t=0. Ελέγχεται η μετάβαση στον κόμβο 2.



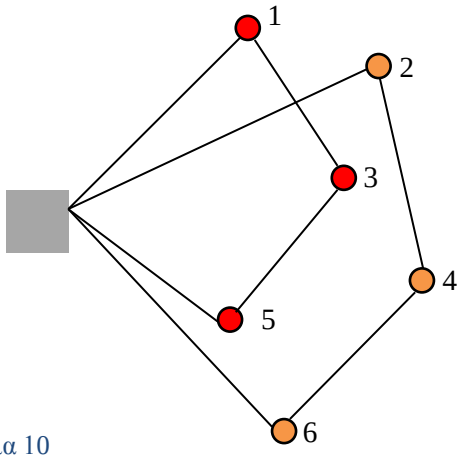
Σχήμα 8

Βήμα 8. Η μετάβαση στον πελάτη 2 είναι εφικτή, πραγματοποιείται και ελέγχεται η μετάβαση στον πελάτη 4.



Σχήμα 9

Βήμα 9. Η διαδικασία επαναλαμβάνεται όπως πριν. Γίνεται η μετάβαση από τον 2 στον 4 και στη συνέχεια από τον 4 στον 6. Οι τελικές διαδρομές είναι οι παρακάτω.



Σχήμα 10

4.3 Εύρεση της βέλτιστης λύσης με τη χρήση του αλγορίθμου περιορισμένης αναζήτησης

Η λύση που βρέθηκε προηγουμένως είναι εφικτή, αλλά δεν είναι βέλτιστη. Για να βελτιστοποιηθεί το συνολικό κόστος και για να μειωθεί ο αριθμός των διαδρομών γίνεται χρήση του αλγορίθμου περιορισμένης αναζήτησης (Tabu Search). Η μέθοδος αυτή παρότι είναι αρκετά περίπλοκη και δύσκολο να εφαρμοστεί εξ' ολοκλήρου, αποδίδει, δίνοντας αρκετά καλά αποτελέσματα. Η μέθοδος στηρίζεται σε εφαρμογές της τοπικής αναζήτησης, όπως είναι οι ανταλλαγές κόμβων μεταξύ διαφορετικών διαδρομών, οι εναλλαγές κόμβων ίδιας διαδρομής και η μεταφορά ενός κόμβου από μία διαδρομή σε μία άλλη. Το κύριο χαρακτηριστικό της περιορισμένης αναζήτησης είναι ότι, χρησιμοποιεί μνήμη για την αποφυγή επανάληψης ίδιων κινήσεων. Για να εφαρμοστεί η μέθοδος απαιτείται η ύπαρξη μίας αρχικής εφικτής λύσης, την οποία καλούμαστε να βελτιώσουμε με διάφορες στρατηγικές.

Σε πρώτο βήμα ορίζουμε τις μεταβλητές από τα υπάρχοντα δεδομένα, που θα χρειαστούν στη συνέχεια. Από το πρώτο μέρος, θα χρειαστούμε τους πίνακες routes, Arriving_time, και τις μεταβλητές V και capacity που περιέχουν των αριθμό των διαδρομών και τη συνολική χωρητικότητα των οχημάτων. Επιπλέον θα χρειαστεί ένας πίνακας tabu_list, που θα αρχικοποιηθεί με μηδενικά και θα είναι διαστάσεων (customers+1,customers+1). Αυτός είναι ο πίνακας απαγορευμένων κινήσεων και όταν πραγματοποιείται κάποια αλλαγή, τα στοιχεία που συμμετέχουν στην αλλαγή θα παίρνουν την τιμή customers/10. Για παράδειγμα, αν ανταλλάξουμε τους κόμβους 42 και 58 και έχουμε 100 πελάτες, το tabu_list(42,58) θα πάρει την τιμή 10. Έτσι θα αποκλειστεί η περίπτωση να ξαναπραγματοποιηθεί αυτή η εναλλαγή για έναν αριθμό επαναλήψεων. Σε κάθε εναλλαγή, κάποια νέα στοιχεία θα πάρουν την μέγιστη τιμή και τα υπόλοιπα θα μειωθούν κατά μία μονάδα. Έτσι θα αποκλειστεί το ενδεχόμενο να οδηγηθούμε σε κύκλους γύρω από μία λύση. Ένας άλλος πίνακας που είναι απαραίτητος, είναι ο πίνακας x διαστάσεων ίδιων με τον πίνακα tabu_list. Στο συγκεκριμένο πίνακα σημειώνεται το αντίστοιχο συνολικό κόστος που θα αποδώσει η συγκεκριμένη ανταλλαγή των κόμβων.

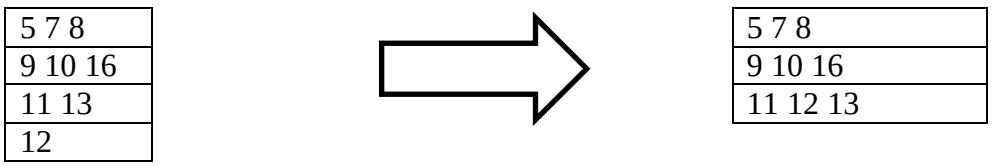
Για την επίλυση του προβλήματος αρχικοποιούμε ως βέλτιστη λύση την εφικτή λύση που βρέθηκε στο πρώτο μέρος της εργασίας, δηλαδή τον τελικό πίνακα των διαδρομών, και ως βέλτιστο κόστος το αρχικό. Κατά την εκτέλεση του αλγορίθμου, οι μετατροπές εκτελούνται σε έναν πίνακα trexwn_lisi, που αρχικοποιείται με τον πίνακα των διαδρομών και τη μεταβλητή trexwn_kostos, που αρχικοποιείται με το αρχικό κόστος. Όταν μία νόμιμη μετακίνηση πραγματοποιηθεί και έχει καλύτερο τελικό αποτέλεσμα, τότε η μεταβλητή veltisto_kostos και ο πίνακας veltisti_lisi θα ανανεωθούν. Αν όχι θα συνεχιστούν οι εναλλαγές και οι μετατροπές στον πίνακα.

Στον αλγόριθμο χρησιμοποιήθηκαν τέσσερις στρατηγικές εμπνευσμένες από την τοπική αναζήτηση, για τη βελτίωση του αποτελέσματος. Η πρώτη είναι η στρατηγική μείωσης των οχημάτων, η δεύτερη είναι η στρατηγική ανταλλαγής κόμβων μεταξύ

διαφορετικών διαδρομών, η τρίτη είναι παρόμοια με τη δεύτερη με μία μικρή αλλαγή και η τέταρτη είναι η στρατηγική εναλλαγής κόμβων μέσα στην ίδια διαδρομή. Στη συνέχεια θα αναλυθεί ο τρόπος λειτουργίας της κάθε μίας και τι αποτέλεσμα μας αποφέρει.

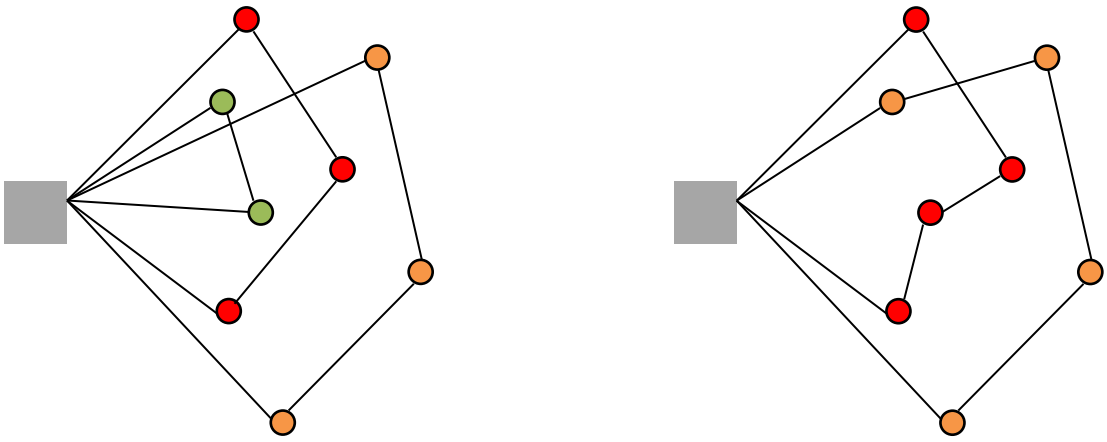
4.3.1 Ελαχιστοποίηση Αριθμού διαδρομών

Το πρώτο τμήμα του αλγορίθμου βελτιστοποίησης εξετάζει αν μπορούν να μειωθεί ο συνολικός αριθμός των διαδρομών, συνεπώς και ο αριθμός τον χρησιμοποιηθέντων οχημάτων. Το συγκεκριμένο τμήμα έχει διαφορετική λειτουργία και λογική, συγκριτικά με τα άλλα. Εδώ επιχειρείται η μεταφορά των κόμβων σε διαφορετικές διαδρομές για την εξάλειψη της διαδρομής. Εάν δεν μπορούν να μετακινηθούν όλα τα στοιχεία, τότε δεν γίνεται να μειωθούν τα δρομολόγια και δεν θα πραγματοποιηθεί καμία μετακίνηση. Το τμήμα του αλγορίθμου, χρησιμοποιεί τη λογική του αλγορίθμου της τοπικής αναζήτησης. Πιο συγκεκριμένα, εκτελεί διαδοχικά 1-0 επανατοποθετήσεις (1-0 relocate), δηλαδή διαγράφει ένα πελάτη από μία διαδρομή και τον τοποθετεί σε μία άλλη. Η λειτουργία της μεθόδου στη γενική περίπτωση φαίνεται στα παρακάτω σχήματα.



Σχήμα 4.3.1.1. Μετακίνηση κόμβου σε διαφορετική διαδρομή, και μείωση των διαδρομών

Και τα αντίστοιχα γραφήματα:



Σχήμα 4.3.1 Στο αριστερό γράφημα έχουμε τρεις διαδρομές. Αριστερά έχει εφαρμοστεί ο αλγόριθμος ελαχιστοποίησης οχημάτων και έχουν μειωθεί σε 2

Το πρώτο βήμα στο πρόβλημα μας είναι η εύρεση της υποψήφιας προς διαγραφή, διαδρομή. Αντιλαμβανόμαστε πως δεν γίνεται να επιλεχθεί η διαδρομή που

εξυπηρετεί τους περισσότερους πελάτες, γιατί θα είναι δύσκολη η μετακίνηση όλων των πελατών. Επιλέγεται το φορτηγό που εξυπηρετεί τον μικρότερο αριθμό πελατών. Εφόσον είναι γνωστή η υποψήφια για διαγραφή διαδρομή, πρέπει να εξεταστούν όλοι οι πελάτες που εμπεριέχονται σε αυτήν, αν μπορούν να μετακινηθούν σε άλλη διαδρομή χωρίς την παραβίαση των περιορισμών.

Οι περιορισμοί που πρέπει να ελεγχθούν αφορούν:

- Το συνολικό φορτίο. Ο πελάτης δεν μπορεί να μετακινηθεί σε μία διαδρομή με διαθέσιμη υπολειπόμενη χωρητικότητα μικρότερη από τη ζήτηση του.
- Το χρόνο άφιξης του. Ο χρόνος άφιξης του πελάτη θα πρέπει να είναι εντός του χρονικού του παραθύρου.
- Την λειτουργία του δρομολογίου. Ο χρόνος άφιξης του πελάτη, δεν πρέπει να επηρεάσει το χρόνο άφιξης των υπόλοιπων πελατών, παραβιάζοντας το χρονικό τους παράθυρο.

Για τον έλεγχο του συνολικού φορτίου, κατασκευάζεται ένα διάνυσμα load, που υπολογίζει το φορτίο του κάθε φορτηγού. Συνεπώς, για μία διαδρομή i, $load(i) + dem(node) <= capacity$, όπου node ο υποψήφιος προς μετακίνηση πελάτης από τη διαδρομή που θέλουμε να αφαιρέσουμε.

Για το χρόνο άφιξης του υποψήφιου πελάτη, κατασκευάζεται μία μεταβλητή που περιέχει τη χρονική στιγμή που θα φτάσει ο πελάτης αν επιχειρηθεί η εξυπηρέτηση του μετά από κάποιον άλλον πελάτη. Αυτή η μεταβλητή κατασκευάζεται σε κάθε έλεγχο που γίνεται για να δούμε, που μπορεί να χωρέσει το στοιχείο που θέλουμε να μετακινήσουμε. Η μεταβλητή αυτή ονομάζεται Arriving_node, και δηλώνεται ως:

$$Arriving\_node = Arriving\_time(\text{pnode}) + A(\text{pnode}, \text{node}) + \text{service\_time}(\text{pnode})$$

, όπου pnode, ο προηγούμενος πελάτης που θα εξυπηρετήσουμε.

Είναι εύκολα παρατηρήσιμο ότι οι χρόνοι άφιξης των πελατών του πίνακα Arriving_time είναι αρκετά κλειστοί και δύσκολα μπορεί να υπάρξει κάποια μετακίνηση. Παρόλα αυτά, υπάρχει η δυνατότητα να γίνουν πιο ελαστικά τα παράθυρα χωρίς να παραβιαστούν οι περιορισμοί των χρονικών παραθύρων. Ο κάθε πελάτης έχει ένα χρονικό παράθυρο και η εξυπηρέτηση του πρέπει να ξεκινήσει σε ένα παράθυρο [ενωρίτερος χρόνος, βραδύτερος χρόνος]. Στις πιο πολλές περιπτώσεις η εξυπηρέτηση γίνεται, πολύ πριν το αργότερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί. Με εύκολη παρατήρηση, είναι αντιληπτό, ότι αν κάποιοι πελάτες, δέχονταν λίγο αργότερα την εξυπηρέτηση αλλά εντός του χρονικού τους παραθύρου, ίσως γινόταν εφικτή η εξυπηρέτηση ενός άλλου πελάτη πρώτα. Πραγματοποιώντας αυτή την ιδέα, θα μπορούσαν να γίνουν πιο εύκολα περισσότερες μετακινήσεις πελατών, επιτυγχάνοντας την τελική διαγραφή κάποιων διαδρομών. Για την εφαρμογή αυτής της ιδέας δημιουργήθηκε ενός πίνακας Delay, που πραγματοποιεί την προαναφερθείσα σκέψη. Με λίγα λόγια δείχνει πόσο μπορεί να καθυστερήσει η εξυπηρέτηση των πελατών χωρίς να παραβιαστεί κανένας χρονικός περιορισμός.

Αυτός ο πίνακας έχει τις ίδιες διαστάσεις με τον πίνακα Arriving_time. Αναλυτικά η κατασκευή του πίνακα:

- Ξεκινάμε από το τελευταίο στοιχείο της κάθε διαδρομής, που αντιστοιχεί στην αποθήκη. Η αποθήκη κλείνει συγκεκριμένη χρονική στιγμή, οπότε το τελευταίο στοιχείο της κάθε διαδρομής θα έχει το χρόνο κλεισίματος της αποθήκης. Οπότε το Delay(i,j) σε αυτή τη θέση ισούται με Dt(1)
- Στη συνέχεια για να βρούμε την μέγιστη καθυστέρηση της εξυπηρέτησης των πελατών, χωρίς την παραβίαση των περιορισμών ξεκινάμε ανάποδα. Η εξυπηρέτηση του πελάτη θα είναι το ελάχιστο αποτέλεσμα ανάμεσα στο προηγούμενο χρόνο καθυστέρησης, μείων την χρονική διάρκεια μετάβασης μείων το χρόνο εξυπηρέτησης, και το βραδύτερο χρόνο εξυπηρέτησης. Δηλαδή ισχύει:
  $Delay(i, j) = \min(Dt(\mathrm{j}), \mathrm{Delay(i, j+1)} - \mathrm{A(j, j+1)} - \mathrm{service\_time(j)})$
- Το πρώτο στοιχείο της κάθε διαδρομής παίρνει μηδενική τιμή, εφόσον αντιστοιχεί στη χρονική στιγμή t=0, όπου το φορτηγό βρίσκεται στην αποθήκη.

Η κατασκευή αυτού του πίνακα βοηθάει στην πιο εύκολη μετακίνηση των πελατών, χωρίς να παραβιάζονται οι περιορισμοί αφού γίνεται έλεγχος της μέγιστης καθυστέρησης που επιδέχεται ο κάθε πελάτης, χωρίς να επηρεαστεί η ροή.

Οι μετακινήσεις των πελατών γίνονται εντός μίας επαναληπτικής διαδικασίας. Σε κάθε μετακίνηση που εκτελείται, ανανεώνεται ο πίνακας αφίξεων, όπως και ο πίνακας μέγιστων καθυστερήσεων και φορτίων των διαδρομών. Η επαναληπτική διαδικασία θα τελειώσει όταν για τον κάθε κόμβο της μικρότερης διαδρομής, εξεταστούν όλες οι πιθανές μετακινήσεις.

Όταν τερματιστεί η επαναληπτική διαδικασία θα ελεγχθεί αν έχουν μετακινηθεί όλοι οι πελάτες που περιλαμβάνονταν στη μικρότερη διαδρομή και αν παραβιάζεται ο περιορισμός της επιστροφής στην αποθήκη. Αν έχουν μετακινηθεί όλοι οι πελάτες στις υπόλοιπες διαδρομές, τότε:

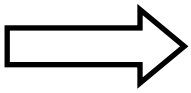
- Μειώνονται τα οχήματα
- Μειώνονται οι διαδρομές από τον πίνακα διαδρομών και χρόνου αφίξεων
- Υπολογίζεται το συνολικό κόστος
- Ο πίνακας veltisti_lisi θα πάρει τη λύση που προέκυψε ,όπως και το veltisto_kostos θα πάρει το υπάρχον κόστος. Σε αυτό το σημείο να αναφέρουμε ότι το κόστος που θα προκύψει μπορεί να είναι μεγαλύτερο από το προηγούμενο βέλτιστο κόστος. Όμως, επειδή ο στόχος του προβλήματος είναι διπλός, η μείωση της συνολικής απόστασης και των διαδρομών, το αποδεχόμαστε. Το κόστος μπορεί εύκολα να αλλάξει και να βελτιωθεί με κατάλληλες ανταλλαγές. Η μείωση των οχημάτων όμως είναι δύσκολο να επιτευχθεί και δεν πρέπει να την απορρίπτουμε.

Σε διαφορετική περίπτωση ο πίνακας των διαδρομών και των αφίξεων δεν θα αλλάξει, και θα γίνει όπως ήταν πριν τις μετακινήσεις.

4.3.2 Ανταλλαγή μεταξύ κόμβων διαφορετικών διαδρομών

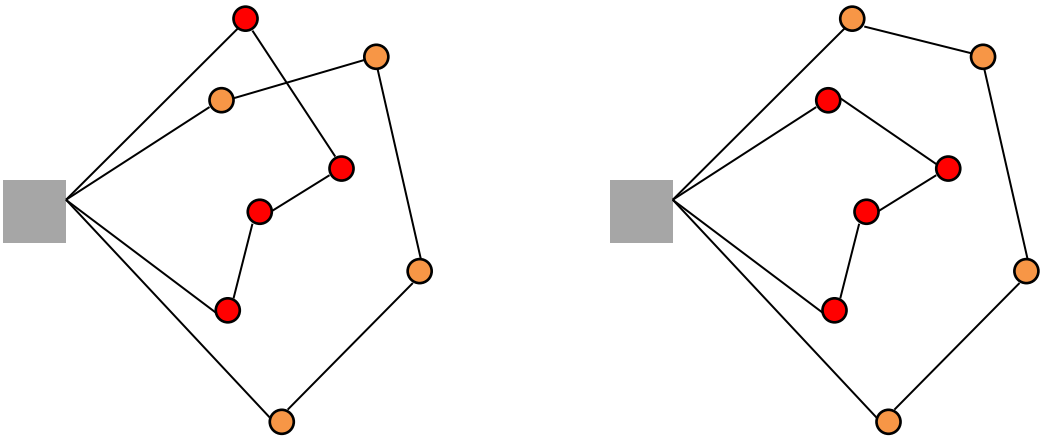
Στο δεύτερο τμήμα του αλγορίθμου θα εξεταστεί αν μπορεί να βελτιωθεί το αποτέλεσμα και να μειωθεί το συνολικό κόστος. Στο αλγοριθμικό κομμάτι αυτό, θα πραγματοποιηθεί η ανταλλαγή κόμβων μεταξύ διαφορετικών διαδρομών. Η ανταλλαγή κόμβων μεταξύ διαφορετικών διαδρομών είναι μία από τις διαδικασίες της τοπικής αναζήτησης. Στη διαδικασία αυτή, επιλέγονται δύο κόμβοι, διαφορετικών διαδρομών που ικανοποιούν τους περιορισμούς, και γίνεται ταυτόχρονη ανταλλαγή. Η διαδικασία φαίνεται στα παρακάτω σχήματα.

1	15	8	22	1
1	2	7	13	1



1	2	8	22	1
1	15	7	13	1

Σχήμα 4.3.2.1 Ανταλλαγή του 15 με το 2.



Σχήμα 4.3.2.2 Αριστερά οι διαδρομές πριν την ανταλλαγή και δεξιά οι διαδρομές μετά την ανταλλαγή

Η βασική ιδέα του τμήματος αυτού είναι ότι αν αλλάξει η σειρά εξυπηρέτησης ανάμεσα σε δύο πελάτες, χωρίς την παραβίαση των περιορισμών, υπάρχει η πιθανότητα να μειωθεί η συνολική απόσταση. Για αυτό σε κάθε επανάληψη θα επιχειρηθούν αρκετές αλλαγές, για να αναζητηθεί ο συνδυασμός εκείνος που βρίσκει το βέλτιστο αποτέλεσμα, χωρίς υπάρχει πρόβλημα καταπάτησης των περιορισμών.

Η πρώτη διαδικασία που εκτελείται, είναι η αναζήτηση των κόμβων που θα γίνει η ανταλλαγή. Συνεπώς, το πρώτο βήμα είναι η εύρεση γειτονιάς λύσεων. Αρχικά επιλέγεται ένας τυχαίος κόμβος. Από αυτόν τον κόμβο θα αναζητήσουμε και θα εξετάσουμε , όλες τις πιθανές ανταλλαγές που δύναται να πραγματοποιηθούν. Δοκιμάζουμε δηλαδή την ανταλλαγή με όλους τους άλλους κόμβους που ανήκουν σε διαφορετικές διαδρομές. Η ανταλλαγή όμως, δεν μπορεί να πραγματοποιηθεί με οποιονδήποτε κόμβο και σε αυτό το σημείο ελέγχεται αν είναι εφικτή η ανταλλαγή με τον εκάστοτε κόμβο. Ο έλεγχος γίνεται στον πίνακα Arriving_time, δηλαδή γίνονται

οι ανταλλαγές μεταξύ των διαδρομών και στη συνέχεια ελέγχεται αν οι χρόνοι αφίξεων, όπως έχουν διαμορφωθεί μετά την ανταλλαγή. Εάν η άφιξη έστω και ενός πελάτη οριστεί εκτός του χρονικού του παραθύρου, το στοιχείο x(pnode,node) θα απειριστεί. Σε διαφορετική περίπτωση το στοιχείο του πίνακα θα πάρει την τιμή που δίνει το ολικό κόστος μετά από αυτή την ανταλλαγή. Το ίδιο θα συμβεί με όλες τις υπόλοιπες ανταλλαγές, που θα πραγματοποιηθούν. Όταν γίνουν όλες οι πιθανές ανταλλαγές έχουμε βρει την γειτονιά λύσεων.

Στη συνέχεια θα πρέπει να βρεθεί ποια ανταλλαγή θα πραγματοποιηθεί τελικά. Αρχικά ορίζουμε ως πιθανή ανταλλαγή αυτή που έχει το μικρότερο κόστος στον πίνακα x. Ακόμα δεν ξέρουμε όμως αν είναι εφικτή και εδώ θα γίνει ο έλεγχος της tabu_list. Αν η κίνηση έχει θετικό αριθμό στον πίνακα tabu_list, τότε δεν μπορεί να πραγματοποιηθεί. Επομένως στη συνέχεια θα επιλεχθεί το δεύτερο μικρότερο στοιχείο του πίνακα x, το τρίτο κ.ο.κ. μέχρι να βρεθεί η ανταλλαγή εκείνη που δεν βρίσκεται στην απαγορευμένη λίστα. Υπάρχει όμως μία εξαίρεση, σε περίπτωση που το νέο αποτέλεσμα είναι καλύτερο από το βέλτιστο κόστος που έχουμε. Σε αυτή την περίπτωση ενεργοποιείται το κριτήριο φιλοδοξίας, το οποίο χρησιμοποιείται όταν η κίνηση είναι απαγορευμένη αλλά μας δίνει καλύτερο αποτέλεσμα, οπότε η κίνηση θα πραγματοποιηθεί

Όταν πραγματοποιηθεί η ανταλλαγή θα ανανεωθούν οι πίνακες Arriving_time και trexwn_lisi. Επίσης θα υπολογιστεί το νέο φορτίο που έχει η κάθε διαδρομή. Εδώ είναι η ιδιαιτερότητα του αλγορίθμου που χρησιμοποιήσαμε. Το φορτίο σε κάποια από τις δύο διαδρομές μπορεί να έχει φορτίο μεγαλύτερο από το διαθέσιμο και να έχει παραβιαστεί ο περιορισμός χωρητικότητας. Αυτό πραγματοποιήθηκε ηθελημένα, γιατί μπορεί να παραβιάζεται ο περιορισμός, αλλά η κίνηση αυτή μπορεί να μας οδηγήσει σε μία γειτονιά που πιθανώς θα μας δώσει καλύτερο αποτέλεσμα. Σε αυτό το σημείο θα χρησιμοποιηθεί ένας νέος πίνακας, και μία νέα μεταβλητή. Ο πίνακας είναι ο penalty και η μεταβλητή η is_penalty που παίρνει τη τιμή 0 και 1. Αναλυτικά οι νέες μεταβλητές:

- penalty(i)=load(i)- capacity, και ελέγχει αν υπάρχει και πόσο είναι η διαφορά ανάμεσα στο φορτίο που έχουμε και στη συνολική διαθέσιμη χωρητικότητα. Σε περίπτωση που penalty<0 η μεταβλητή θα πάρει την τιμή 0. Αλλιώς θα υπάρχει το έξτρα φορτίο του φορτηγού
- is_penalty, παίρνει τιμή ίση με το 1 αν ο πίνακας penalty έχει φορτίο παραπάνω από τη διαθέσιμη χωρητικότητα και τιμή ίση με 0 σε αντίθετη περίπτωση

Σε αυτό το σημείο να αναφερθεί ότι η βέλτιστη λύση δεν μπορεί να πάρει τη λύση που δίνει μία ανταλλαγή που παραβιάζει τους περιορισμούς. Κρατάμε όμως την ανταλλαγή προσωρινά, και θα την αλλάξουμε στο επόμενο κομμάτι. Επιλέξαμε να δεχθούμε αυτή τη λύση μόνο και μόνο επειδή μπορεί να μας οδηγήσει σε νέα γειτονιά και να οδηγηθούμε σε καλύτερο αποτέλεσμα.

Στη συνέχεια, αφού πραγματοποιηθεί η ανταλλαγή, ανανεώνονται οι μεταβλητές και οι πίνακες. Επίσης γίνεται έλεγχος αν το αποτέλεσμα είναι καλύτερο από το βέλτιστο και ταυτόχρονα αν η μεταβλητή is_penalty έχει την τιμή 0. Αν ισχύουν αυτά τα δύο, τότε ανανεώνεται το βέλτιστο κόστος και η βέλτιστη λύση του προβλήματος. Στη συνέχεια ανανεώνεται και ο πίνακας tabu_list και ξαναμηδενίζεται ο πίνακας x.

4.3.3 Περίπτωση ύπαρξης διαδρομής με τιμωρία

Παραπάνω περιγράφηκε η διαδικασία ανταλλαγής κόμβων μεταξύ διαφορετικών διαδρομών. Παραλήφθηκε όμως προσωρινά ο περιορισμός χωρητικότητας των φορτηγών, για την μετάβαση σε νέα καλύτερη γειτονιά. Σε αυτό το κομμάτι επαναλαμβάνεται η ανταλλαγή κόμβων μεταξύ διαφορετικών διαδρομών στην περίπτωση που κάποια από τις διαδρομές μας είναι 'τιμωρημένη'. Το αλγοριθμικό αυτό τμήμα, δεν θα πραγματοποιηθεί στην περίπτωση που δεν παραβιάζεται ο περιορισμός χωρητικότητας.

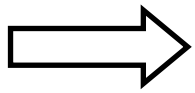
Σκοπός είναι να εξαλείψουμε την διαδρομή με τιμωρία, και την εξισορρόπηση των διαδρομών. Η εξισορρόπηση θα γίνει από την διαδρομή που έχει τιμωρία προς μία από τις υπόλοιπες διαδρομές. Αρχικά επιλέγεται το φορτηγό που έχει φορτίο παραπάνω από τη διαθέσιμη χωρητικότητα του. Η ανταλλαγή που θα πραγματοποιηθεί πρέπει να γίνει με κάποιον από τους πελάτες που εξυπηρετούνται από αυτή τη διαδρομή. Γίνεται έλεγχος για όλους τους πελάτες αυτής της διαδρομής, με όλες τις πιθανές ανταλλαγές. Παρομοίως θα ελεγχθεί ο περιορισμός χρονικού παραθύρου. Αυτή τη φορά όμως, θα γίνει και έλεγχος χωρητικότητας. Η ανταλλαγή που θα πραγματοποιηθεί θα πρέπει να εξισορροπεί τις διαδρομές και το συνολικό φορτίο να μην υπερβαίνει τη μέγιστη χωρητικότητα του οχήματος. Τα κόστη των πιθανών ανταλλαγών θα αποθηκευτούν στον πίνακα x. Η διαδικασία θα επαναληφθεί για όλους τους κόμβους που περιλαμβάνονται στη διαδρομή.

Στη συνέχεια η διαδικασία θα πραγματοποιηθεί όπως προηγουμένως. Θα επιλεγεί η ανταλλαγή εκείνη που δεν είναι απαγορευμένη από την tabu_list, ή είναι απαγορευμένη αλλά επιφέρει καλύτερο βέλτιστο κόστος οπότε ενεργοποιείται το κριτήριο φιλοδοξίας. Αφού επιλεγεί, θα ελεγχθεί αν η τρέχουσα λύση είναι καλύτερη από τη βέλτιστη και θα γίνει ανανέωση αναλόγως. Τέλος ανανεώνονται όλες οι μεταβλητές και οι πίνακες όπως στο προηγούμενο τμήμα.

4.3.4 Ανταλλαγές κόμβων μεταξύ ίδιων διαδρομών

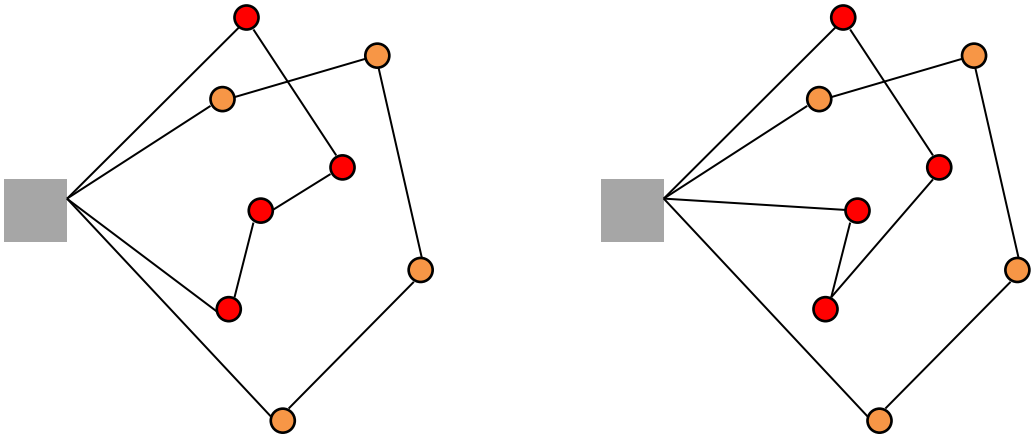
Στο αλγοριθμικό τμήμα αυτό πραγματοποιούνται οι ανταλλαγές μεταξύ κόμβων ίδιας διαδρομής. Η διαδικασία αυτή είναι εφαρμογή του αλγορίθμου τοπικής αναζήτησης (swap). Στο συγκεκριμένο πρόβλημα, λόγω της ύπαρξης των χρονικών παράθυρων είναι δυνατή η εναλλαγή μεταξύ πλησίον κόμβων. Οι εναλλαγές παρουσιάζονται στα παρακάτω σχήματα.

1	15	8	22	1
1	2	7	13	1



1	15	8	22	1
1	7	2	13	1

Σχήμα 4.3.4.1 Ανταλλαγή του 2 με το 7.



Σχήμα 4.3.4.2. Αριστερά οι κόμβοι πριν την εναλλαγή και δεξιά οι κόμβοι μετά την εναλλαγή

Η διαδικασία επίλυσης είναι παρόμοια με πριν. Η αλλαγή υπάρχει στον τρόπο αναζήτησης των κόμβων που θα γίνει η ανταλλαγή και στον έλεγχο των περιορισμών. Εδώ δεν επιλέγεται κάποιο τυχαίο σημείο εκκίνησης, αλλά γίνεται έλεγχος για όλα τα στοιχεία του πίνακα διαδρομών, εκτός των αποθηκών, με το διπλανό τους. Στο συγκεκριμένο πρόβλημα, αν μπορεί να επιτευχθεί κάποια αλλαγή θα είναι μεταξύ πλησίον στοιχείων. Σε κάθε διαδρομή, γίνεται έλεγχος μεταξύ των χρονικών παραθύρων τον πελατών j-1 και j και τον χρόνων άφιξης. Αν ο χρόνος άφιξης του πελάτη j-1 εμπεριέχεται στο χρονικό παράθυρο του πελάτη j και αντίστροφα, τότε γίνεται μία αρχική αποδοχή της ανταλλαγής. Στη συνέχεια κατασκευάζεται ο πίνακας αφίξεων και ελέγχεται αν οι χρόνοι άφιξης στους πελάτες είναι εντός του χρονικού περιθωρίου που έχει ο κάθε πελάτης. Αν παραβιάζεται έστω και ένα χρονικό παράθυρο, τότε το στοιχείο x(j-1,j) θα απειριστεί. Σε διαφορετική περίπτωση θα γίνει αποδεκτό. Στις εναλλαγές μεταξύ κόμβων ίδιας διαδρομής, δεν χρειάζεται να ελεγχθεί ο περιορισμός χωρητικότητας, αφού δεν αλλάζει το συνολικό φορτίο της διαδρομής.

Αφού βρεθεί η γειτονιά, η διαδικασία είναι η ίδια με πριν. Θα επιλεγεί η ανταλλαγή που έχει το μικρότερο κόστος και θα γίνουν οι αντίστοιχοι έλεγχοι στη λίστα απαγορευμένων κινήσεων. Τέλος θα πραγματοποιηθεί η ανανέωση σε όλους τους πίνακες.

4.3.5 Τερματισμός αλγορίθμου και ψευδοκώδικας

Η επαναληπτική διαδικασία επαναλαμβάνεται για πολλές επαναλήψεις τις οποίες τις προκαθορίζουμε εμείς. Όσο πιο πολλές είναι οι επαναλήψεις, τόσο αυξάνονται οι πιθανότητες να βρεθεί καλύτερο αποτέλεσμα. Μετά από έναν αριθμό επαναλήψεων, η λίστα απαγορευμένων κινήσεων καθαρίζει και στο επόμενο βήμα επιτρέπονται όλες οι αλλαγές.

Ο ψευδοκώδικας της διαδικασίας διαμορφώνεται ως εξής:

Δημιουργία εφικτής λύσης

Εισαγωγή των δεδομένων του προβλήματος

Υπολογισμός πρόσθετων μεταβλητών και πινάκων

Εφαρμογή Αλγορίθμου απληστίας

Εύρεση αρχικής εφικτής υποβέλτιστης λύσης

Βελτιστοποίηση της λύσης

Για έναν συγκεκριμένο αριθμό επαναλήψεων

Αλγόριθμος μείωσης οχημάτων

Αν ο αριθμός των πελατών σε μία διαδρομή ισούται με μηδέν

Διέγραψε τη διαδρομή

Ανανέωσε τη βέλτιστη λύση και το βέλτιστο αποτέλεσμα

Τέλος Αν

Αλγόριθμος ανταλλαγών μεταξύ πελατών διαφορετικών διαδρομών

Εύρεση γειτονικών λύσεων που δεν παραβιάζουν τον περιορισμό χρονικών παραθύρων

Αν υπάρχει εφικτή ανταλλαγή

Μέχρι να βρεθεί εφικτή μη περιορισμένη ανταλλαγή ή να ενεργοποιηθεί το κριτήριο της φιλοδοξίας

Τέλος Μέχρι

Ανανέωσε tabu_list

Αν το κόστος είναι μικρότερο από το βέλτιστο κόστος και δεν παραβιάζεται ο περιορισμός χωρητικότητας

Ανανέωσε βέλτιστο κόστος και βέλτιστη λύση

Τέλος Αν

Τέλος Αν

Αν κάποια διαδρομή έχει τιμωρία

Αλγόριθμος ανταλλαγών στην διαδρομή με τιμωρία

Εύρεση γειτονικών λύσεων που δεν παραβιάζουν τους περιορισμούς παραθύρων και χωρητικότητας

Μέχρι να βρεθεί εφικτή μη περιορισμένη ανταλλαγή ή να ενεργοποιηθεί το κριτήριο της φιλοδοξίας

Τέλος Μέχρι

Ανανέωσε tabu_list

Αν το κόστος είναι μικρότερο από το βέλτιστο

Ανανέωσε βέλτιστο κόστος και βέλτιστη λύση

Τέλος Αν

Τέλος Αν

Αλγόριθμος ανταλλαγών μεταξύ κόμβων ίδιας διαδρομής

Εύρεση γειτονικών εφικτών λύσεων που δεν παραβιάζουν τους περιορισμούς

Μέχρι να βρεθεί κάποια εφικτή μη περιορισμένη κίνηση ή ενεργοποιηθεί το κριτήριο φιλοδοξίας

Τέλος Μέχρι

Ανανέωσε tabu_list

Αν το κόστος είναι μικρότερο από το βέλτιστο

Ανανέωσε βέλτιστο κόστος και βέλτιστη λύση

Τέλος Αν

Τέλος Για

Κεφάλαιο 5.
Περιγραφή και ανάλυση των αποτελεσμάτων

5.1 Γενική περιγραφή των αποτελεσμάτων

Για την αποτύπωση της ποιότητας των αποτελεσμάτων του αλγορίθμου τρέχτηκαν μία σειρά προβλημάτων. Τα προβλήματα που διερευνήθηκαν είναι τα γνωστά και ευρέως χρησιμοποιούμενα προβλήματα του Solomon [21]. Από τα προβλήματα αυτά τα δεδομένα μας ήταν ο αριθμός των πελατών, η μέγιστη χωρητικότητα των οχημάτων, η οι τετμημένες και τεταγμένες των πελατών, τα χρονικά παράθυρα, η ζήτηση, ο χρόνος εξυπηρέτησης και ο μέγιστος διαθέσιμος αριθμός οχημάτων, και ο χρόνος που ισούταν με την απόσταση. Στη διάθεση μας είχαμε 56 από αυτά τα προβλήματα. Σε όλα τα προβλήματα μας, σταθερά ήταν ο αριθμός των πελατών που ήταν ίσος με 100 και ο αριθμός των διαθέσιμων οχημάτων που ισούται με 25.

Τα 56 προβλήματα του Solomon, χωρίζονταν σε 6 επιμέρους κατηγορίες προβλημάτων. Ο διαχωρισμός γινόταν, ανάλογα με την χωρητικότητα, το μέγιστο χρονικό παράθυρο της αποθήκης, το χρόνο εξυπηρέτησης, τη διασπορά των πελατών στο χώρο και το εύρος των χρονικών παράθυρων. Οι κατηγορίες των προβλημάτων ήταν C1, C2, R1, R2, RC1, RC2 με την κάθε κατηγορία να περιέχει από 8 έως 12 προβλήματα.

Στις κατηγορίες C1, C2 οι πελάτες είναι τοποθετημένοι ομοιόμορφα. Η κατηγορία C1 περιέχει αυστηρά χρονικά παράθυρα και μέγιστη χωρητικότητα των οχημάτων ίση με 200, Ο χρόνος εξυπηρέτησης των πελατών ισούται με 90 μονάδες. Στα προβλήματα C2 τα παράθυρα είναι πιο ανοιχτά, η μέγιστη χωρητικότητα ισούται με 700 μονάδες και ο χρόνος εξυπηρέτησης ισούται με 90 μονάδες. Στα προβλήματα C2, μεγαλώνει και ο χρόνος κλεισίματος της αποθήκης. Η κατηγορία C1 περιέχει 9 προβλήματα προς επίλυση και η κατηγορία C2, 8 προβλήματα προς επίλυση

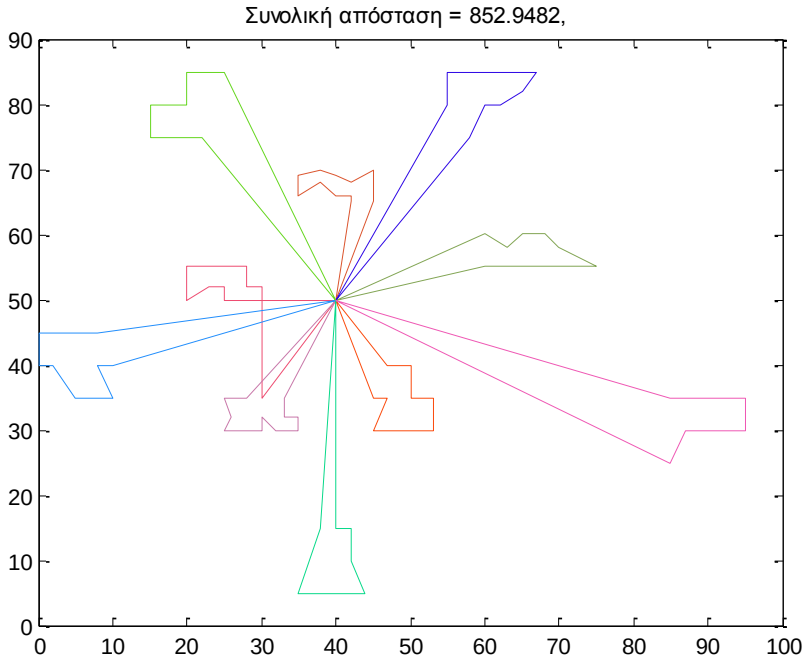
Στις κατηγορίες προβλημάτων R1 και R2 οι πελάτες είναι κατανεμημένοι τυχαία και ανομοιόμορφα. Ο χρόνος εξυπηρέτησης είναι κοινός και ισούται με 10 μονάδες. Η χωρητικότητα στα προβλήματα R1 είναι 200 μονάδες και στα προβλήματα R2 1000 μονάδες. Τα χρονικά παράθυρα στα προβλήματα R1 είναι κλειστά, ενώ τα προβλήματα R2 χαλαρά. Η αποθήκη κλείνει πολύ πιο αργά στα προβλήματα R2. Τα προβλήματα R1 είναι 12 στον αριθμό και τα προβλήματα R2 11.

Στα προβλήματα RC υπάρχει συνδυασμός των δύο προηγούμενων κατηγοριών. Κάποιοι από τους πελάτες είναι τοποθετημένοι σε ομάδες και άλλοι τυχαία. Τα προβλήματα της κατηγορίας RC1 έχουν σχετικά κλειστά παράθυρα και τα προβλήματα της κατηγορίας RC2 έχουν κάποιους πελάτες με πιο κλειστά παράθυρα και κάποιους με πιο χαλαρά χρονικά παράθυρα. Ο χρόνος εξυπηρέτησης είναι 10 μονάδες. Η μέγιστη χωρητικότητα είναι 200 μονάδες στα RC1 και 1000 μονάδες στα RC2. Ο βραδύτερος χρόνος στα προβλήματα RC2 είναι πολύ μεγαλύτερος από ότι στα προβλήματα RC1. Επιλύσαμε 8 προβλήματα από την κάθε κατηγορία.

Το κάθε πρόβλημα τρέχτηκε σε 100.000 επαναλήψεις 5-10 φορές το καθένα ξεχωριστά. Από αυτές κρατήσαμε το καλύτερο αποτέλεσμα και την καλύτερη λύση που προέκυψε. Αναλυτικά το βέλτιστο κόστος και η βέλτιστη λύση καθώς και το τελικό γράφημα θα παρουσιαστούν στις επόμενες σελίδες.

5.2 Αναλυτικά αποτελέσματα των προβλημάτων

5.2.1 Αποτελέσματα προβλήματος C101

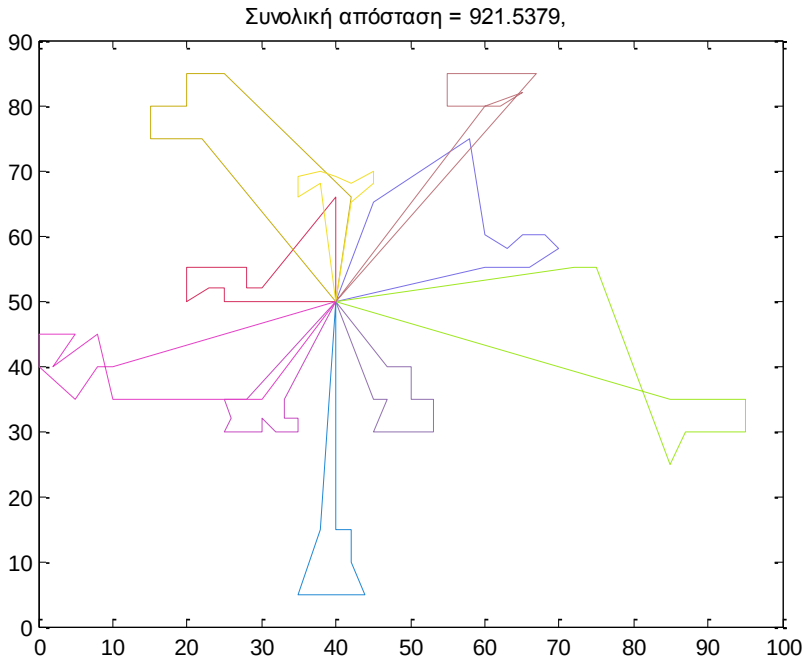


Σχήμα 5.1 Δρομολόγηση οχημάτων C101

Πίνακας Δρομολογίων του προβλήματος C101

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

5.2.2 Αποτελέσματα προβλήματος C102

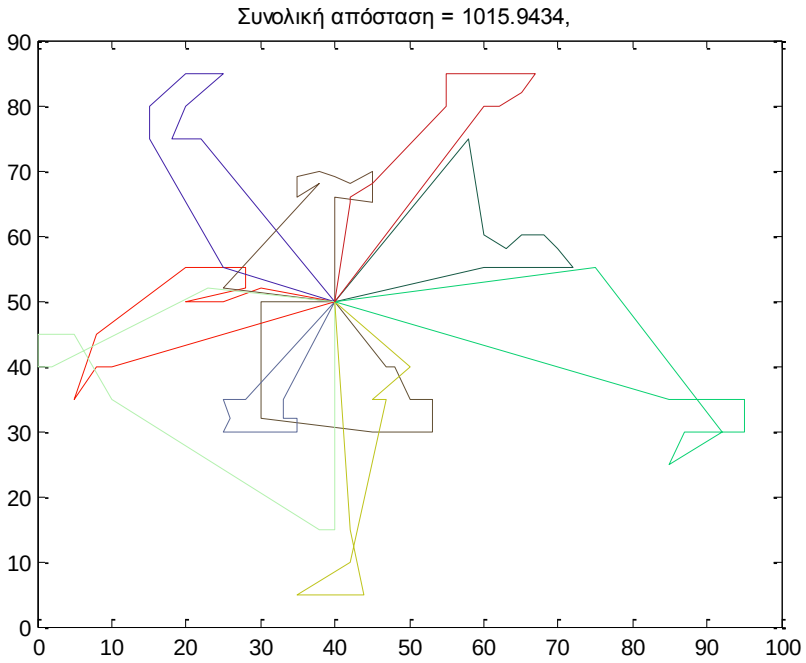


Σχήμα 5.2 Δρομολόγηση οχημάτων C102

Πίνακας δρομολογίων προβλήματος C102

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

5.2.3 Αποτελέσματα προβλήματος C103

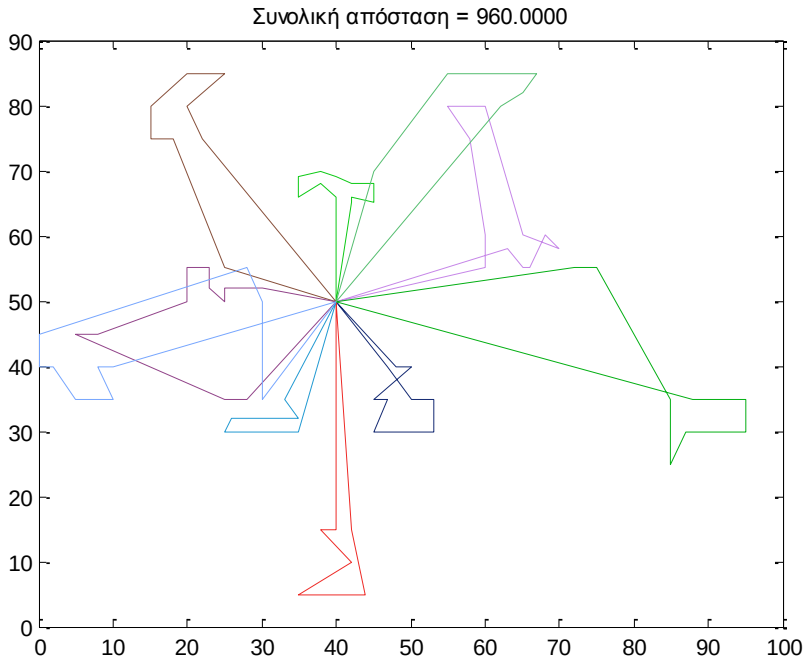


Σχήμα 5.3 Δρομολόγηση οχημάτων C103

Πίνακας δρομολογίων προβλήματος C103

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

5.2.4 Αποτελέσματα προβλήματος C104

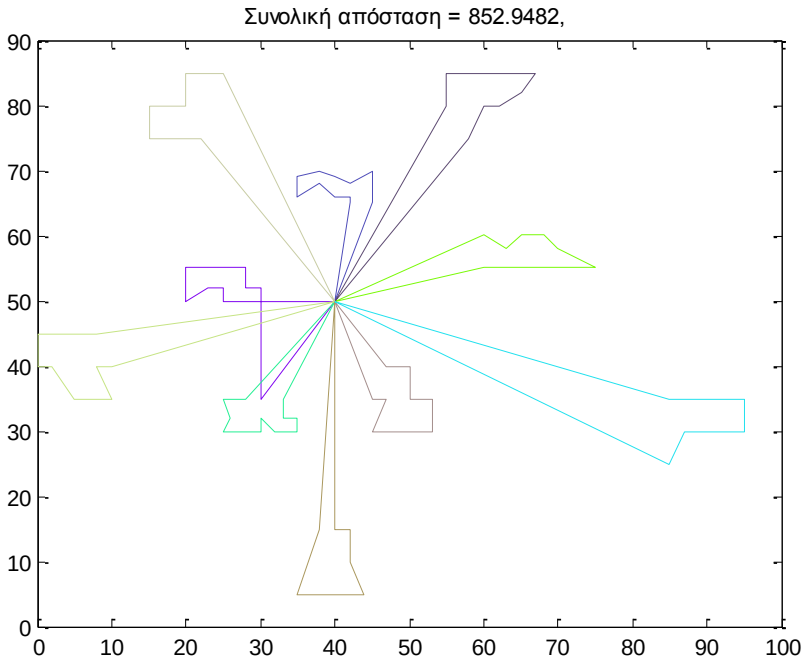


Σχήμα 5.4 Δρομολόγηση οχημάτων C104

Πίνακας δρομολογίων προβλήματος C104

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

5.2.5 Αποτελέσματα προβλήματος C105

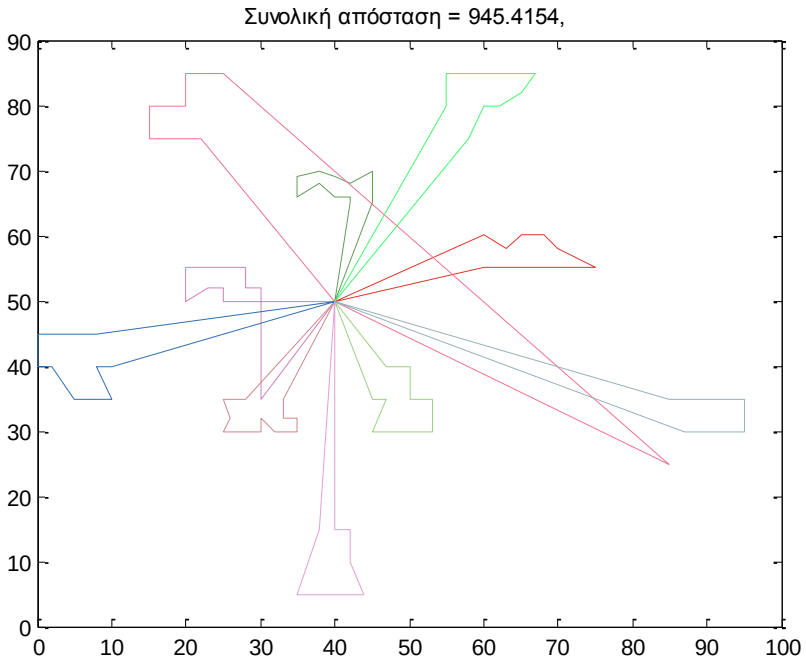


Σχήμα 5.5 Δρομολόγηση οχημάτων C105

Πίνακας δρομολογίων προβλήματος C105

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

5.2.6 Αποτελέσματα προβλήματος C106

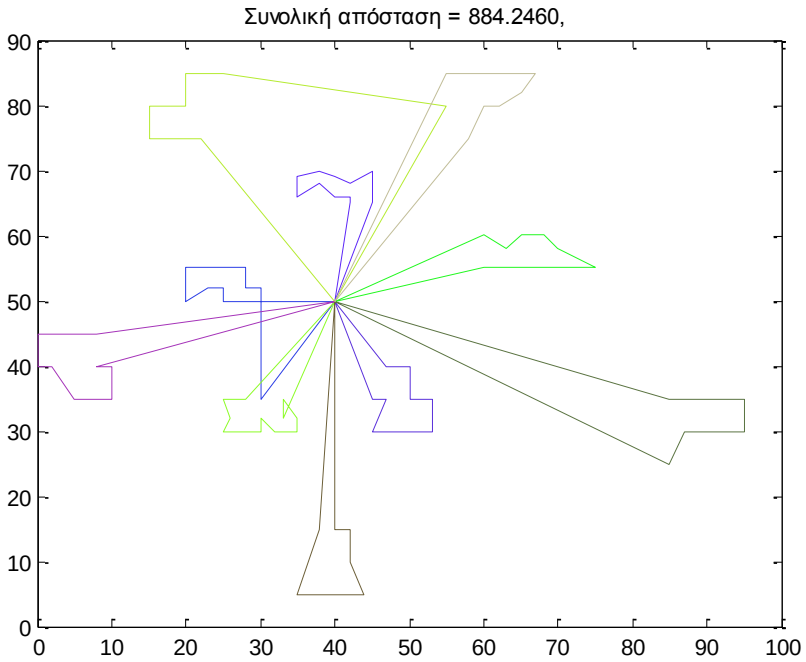


Σχήμα 5.6 Δρομολόγηση οχημάτων C106

Πίνακας δρομολογίων προβλήματος C106

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

5.2.7 Αποτελέσματα προβλήματος C107

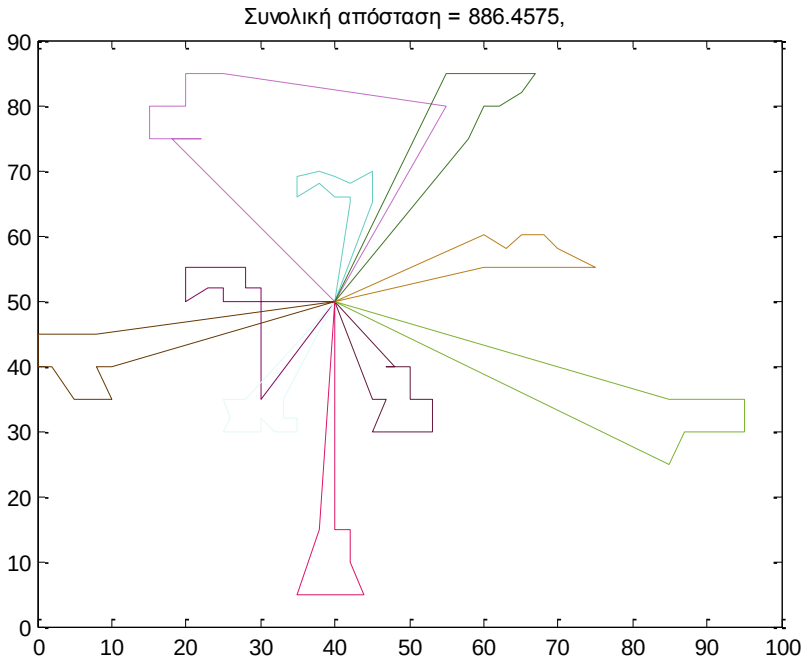


Σχήμα 5.7 Δρομολόγηση οχημάτων C107

Πίνακας δρομολογίων προβλήματος C107

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

5.2.8 Αποτελέσματα προβλήματος C108

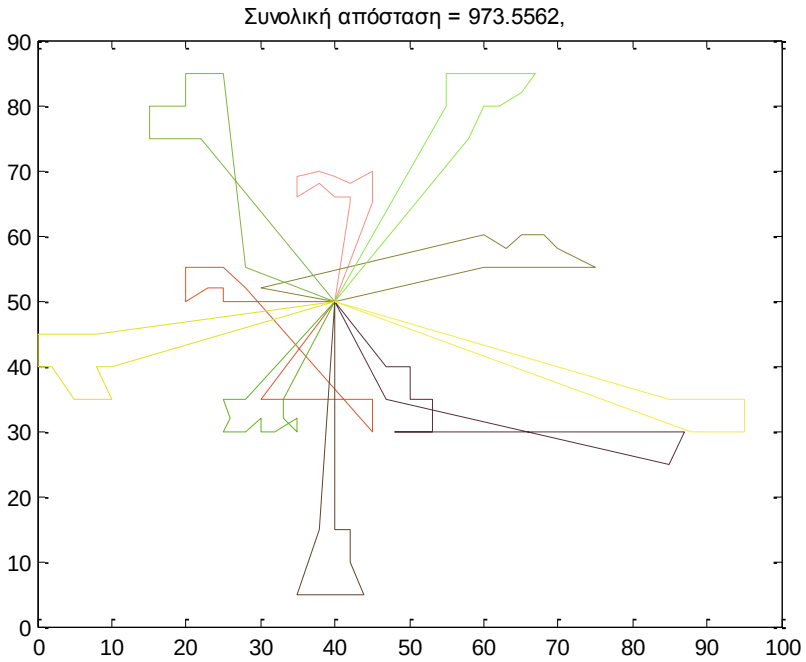


Σχήμα 5.8 Δρομολόγηση οχημάτων C108

Πίνακας δρομολογίων προβλήματος C108

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

5.2.9 Αποτελέσματα προβλήματος C109

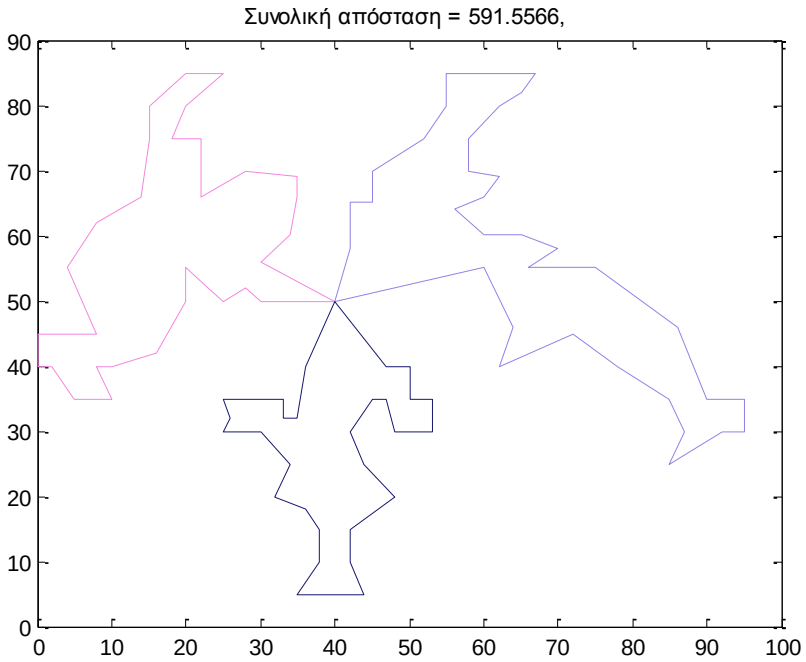


Σχήμα 5.9 Δρομολόγηση οχημάτων C109

Πίνακας δρομολογίων προβλήματος C109

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

5.2.10 Αποτελέσματα προβλήματος C201



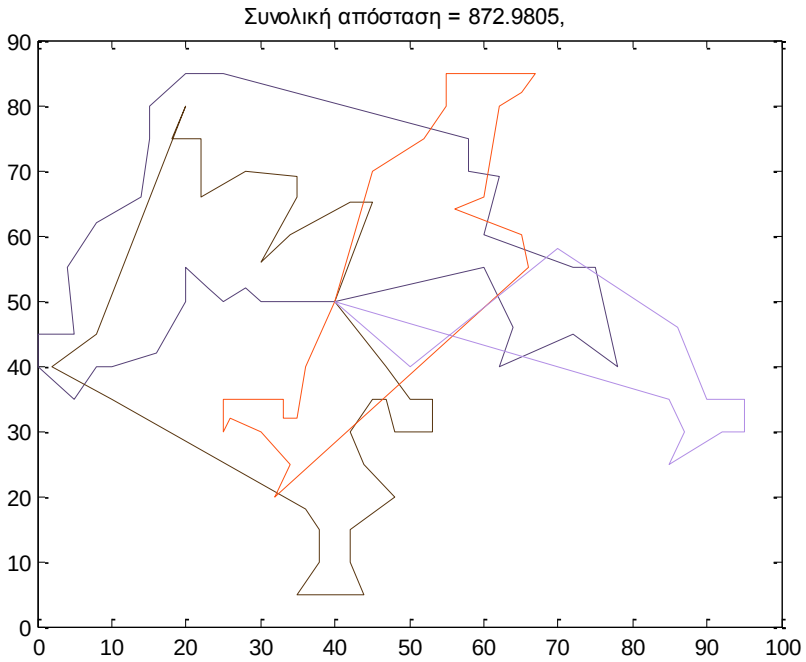
Σχήμα 5.10 Δρομολόγηση οχημάτων C201

Πίνακας δρομολογίων προβλήματος C201

1	94	6	76	3	2	100	101	98	93	95	96	99	8	4	5	90	92	89
1	21	23	25	28	31	30	7	33	34	32	36	38	39	40	37	35	29	27
1	68	64	63	75	73	62	65	67	70	69	66	50	56	55	54	57	59	61

85	87	84	83	86	77	72	71	74	81	80	82	79	78	97	88	91	1
24	19	20	17	15	13	16	18	14	26	10	12	11	9	22	1		
60	58	41	45	47	46	52	51	53	48	44	43	42	49	1			

5.2.11 Αποτελέσματα προβλήματος C202



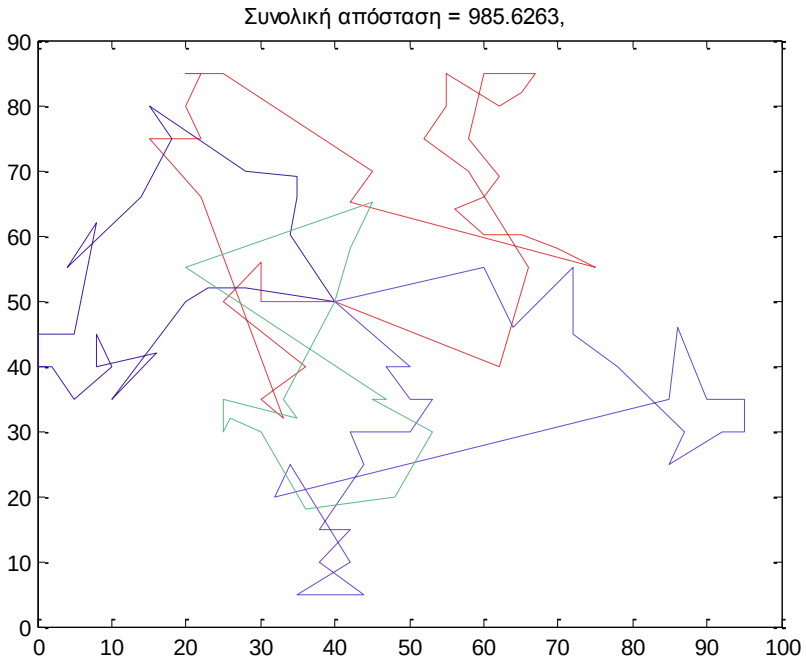
Σχήμα 5.11 Δρομολόγηση οχημάτων C202

Πίνακας δρομολογίων προβλήματος C202

1	21	23	25	28	31	30	7	33	34	36	39	40	37	29	27	24
1	68	63	75	73	62	65	67	70	69	66	50	56	55	54	57	59
1	94	3	2	100	101	98	93	95	96	5	90	89	87	45	47	46
1	64	85	86	77	72	71	74	81	80	82	1					

19	20	17	15	13	99	8	4	92	84	83	79	78	97	88	91	1		
61	60	58	41	32	38	35	16	18	14	26	10	12	11	22	9	6	76	1
51	52	53	48	44	43	42	49	1										

5.2.12 Αποτελέσματα προβλήματος C203



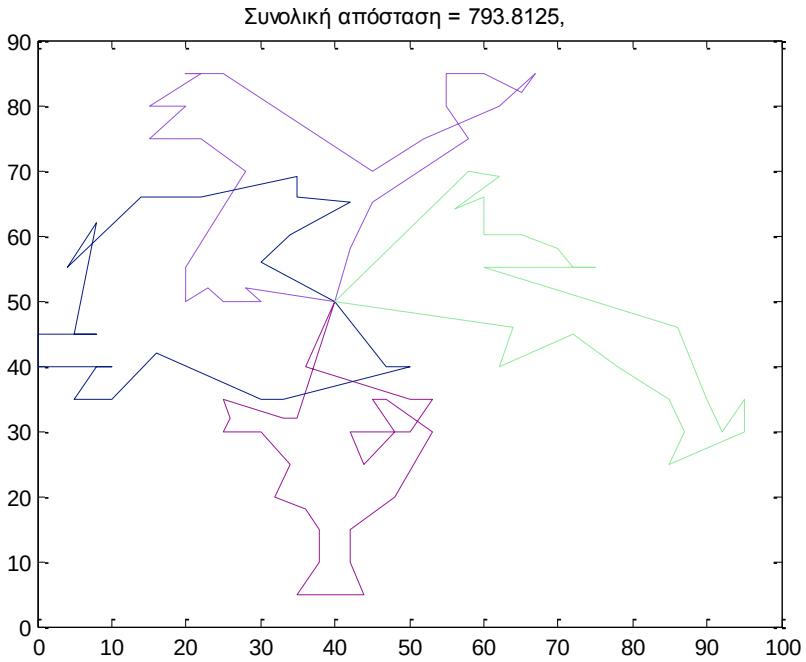
Σχήμα 5.12 Δρομολόγηση οχημάτων C203

Πίνακας δρομολογίων προβλήματος C203

1	94	3	2	100	101	98	93	96	95	99	4	8	5	90	76	24
1	22	23	53	28	31	7	35	33	34	32	36	39	38	40	37	91
1	92	63	75	62	73	65	66	50	56	60	55	57	59	58	45	46
1	64	68	67	70	69	47	54	61	41	43	42	44	48	1		

19	20	17	13	15	16	18	14	10	12	11	6	25	30	49	1		
89	85	87	84	83	82	77	72	71	74	81	80	86	79	78	88	97	1
51	52	27	29	26	21	9	1										

5.2.13 Αποτελέσματα προβλήματος C204



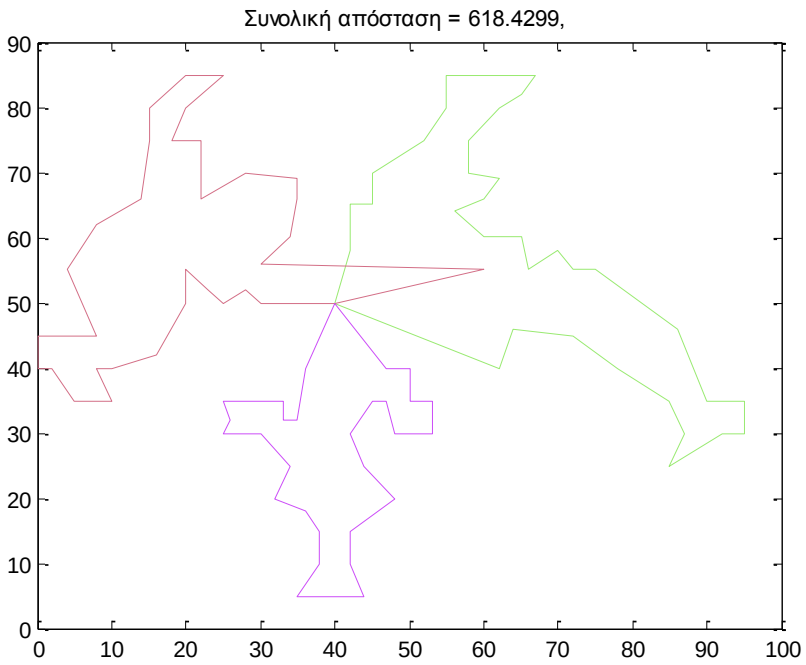
Σχήμα 5.13 Δρομολόγηση οχημάτων C204

Πίνακας δρομολογίων προβλήματος C204

1		49	63	75	62	69	66	65	70	67	73	50	56	55	54	57	59
1		94	76	99	100	101	98	95	93	96	2	3	13	17	15	20	16
1		68	64	44	48	7	32	36	34	33	38	39	40	35	37	27	29
1		8	4	90	5	92	89	85	84	83	87	91	86	77	74	72	71

61	60	58	41	45	47	46	52	51	53	43	42	1
19	18	14	10	31	30	28	25	21	23	1		
24	26	12	11	6	9	22	1					
81	80	82	79	78	97	88	1					

5.2.14 Αποτελέσματα προβλήματος C205



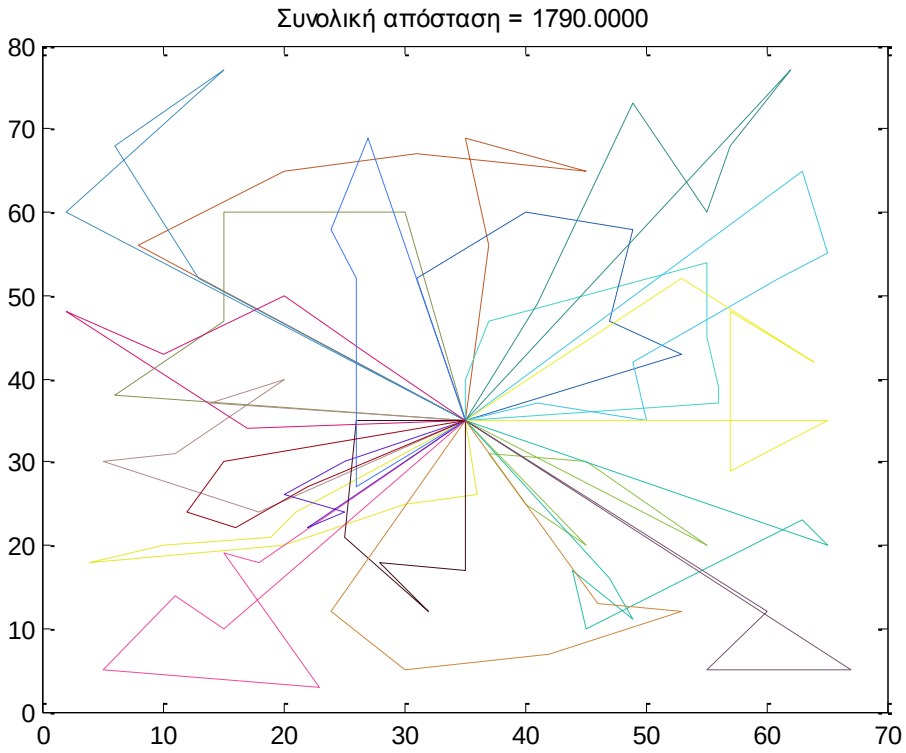
Σχήμα 5.14 Δρομολόγηση οχημάτων C205

Πίνακας δρομολογίων προβλήματος C205

1		94	6	76	3	2	100	101	98	93	95	96	99	8	4	5	90
1		21	23	25	28	31	30	7	33	34	32	36	38	39	40	37	35
1		68	64	63	75	73	62	65	67	70	69	66	50	56	55	54	57

92	89	87	85	84	83	86	77	72	71	74	81	80	82	79	78	88	97	1
29	27	24	19	20	17	15	13	16	18	14	26	10	12	11	9	22	91	1
59	61	60	58	41	45	47	46	52	51	53	48	44	43	42	49	1	1	1

5.2.15 Αποτελέσματα προβλήματος R101

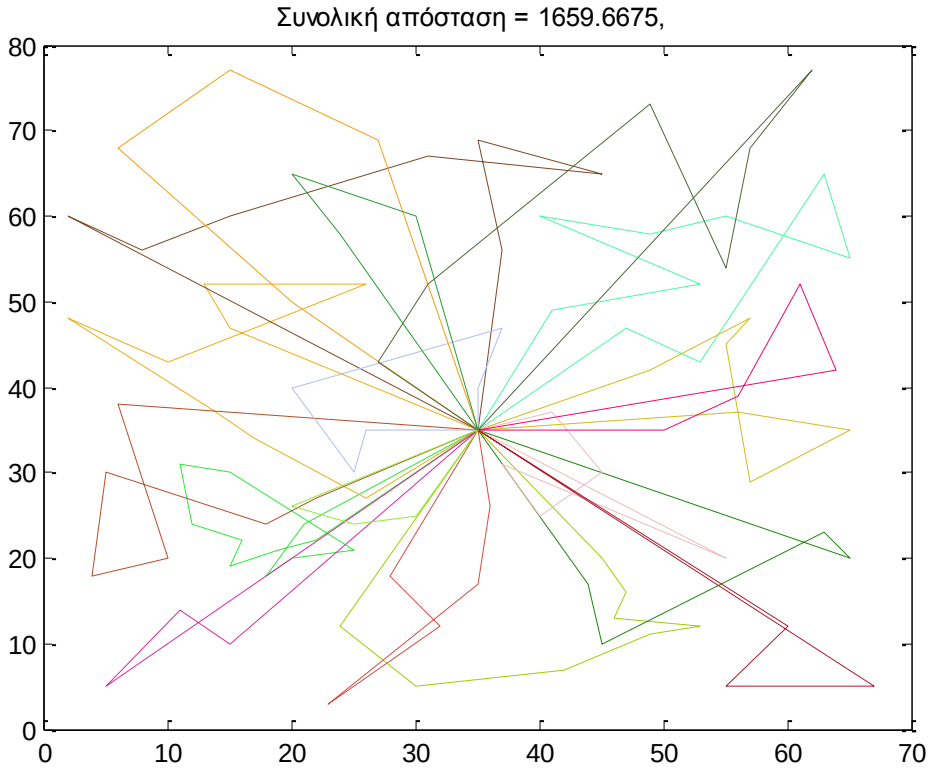


Σχήμα 5.15 Δρομολόγηση οχημάτων R101

Πίνακας δρομολογίων προβλήματος R101

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

5.2.16 Αποτελέσματα προβλήματος R102

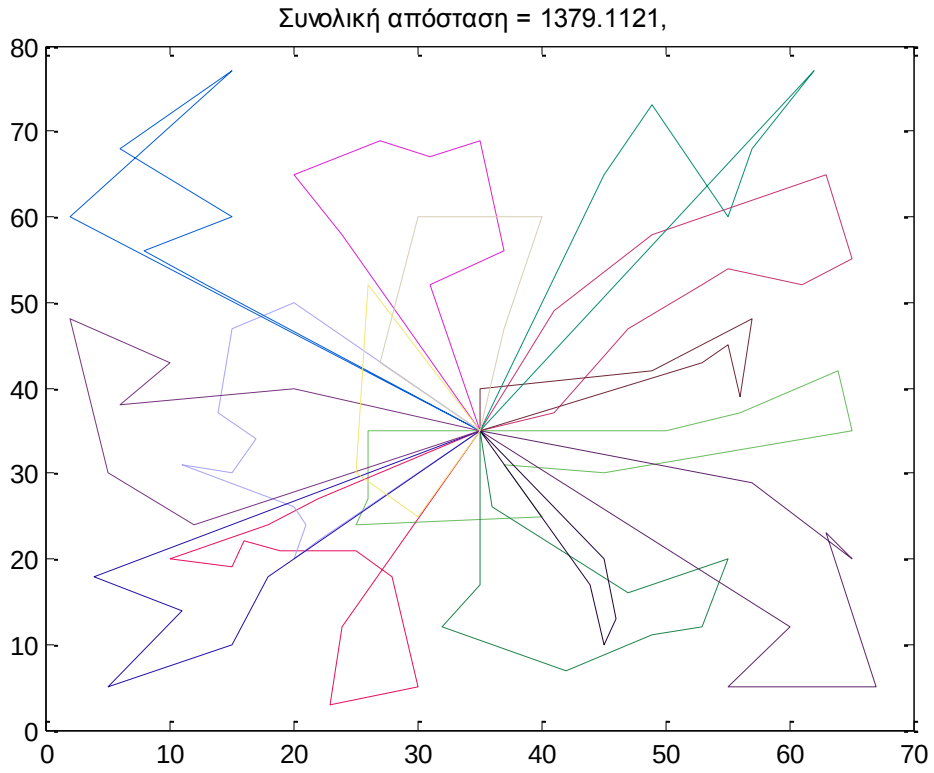


Σχήμα 5.16 Δρομολόγηση οχημάτων R102

Πίνακας δρομολογίων προβλήματος R102

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

5.2.17 Αποτελέσματα προβλήματος R103

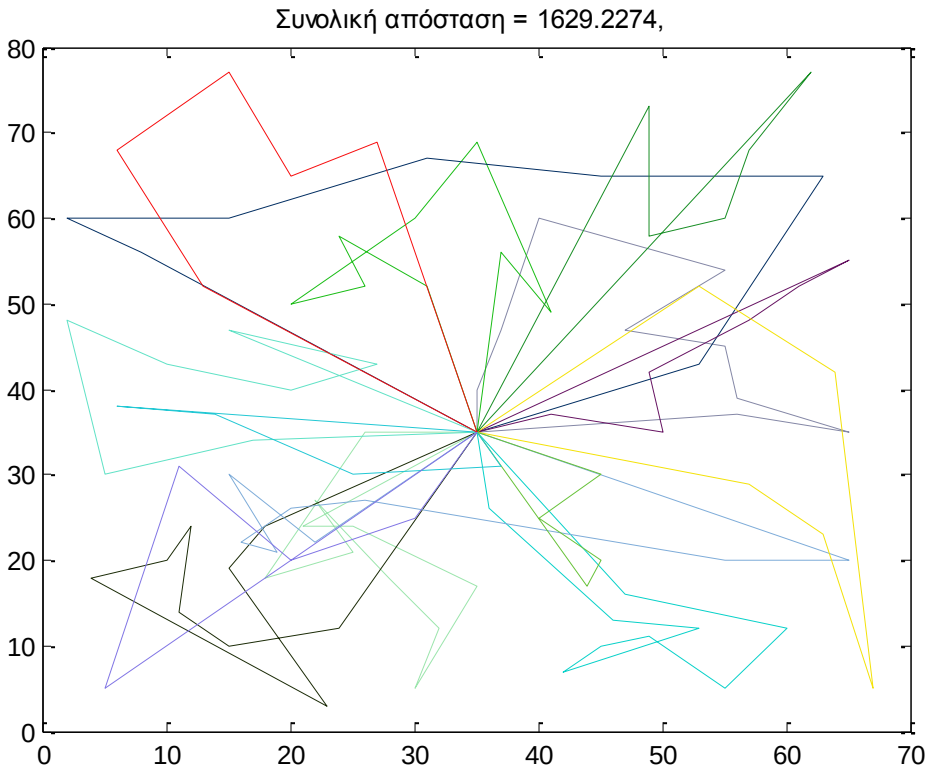


Σχήμα 5.17 Δρομολόγηση οχημάτων R103

Πίνακας δρομολογίων προβλήματος R103

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

5.2.18 Αποτελέσματα προβλήματος R105

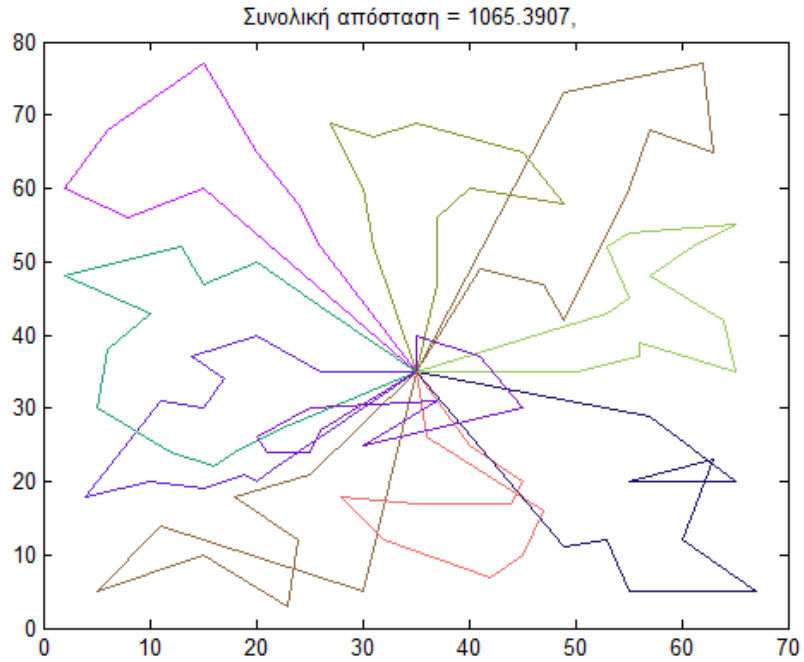


Σχήμα 5.18 Δρομολόγηση οχημάτων R103

Πίνακας δρομολογίων προβλήματος R105

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

5.2.19 Αποτελέσματα προβλήματος R108

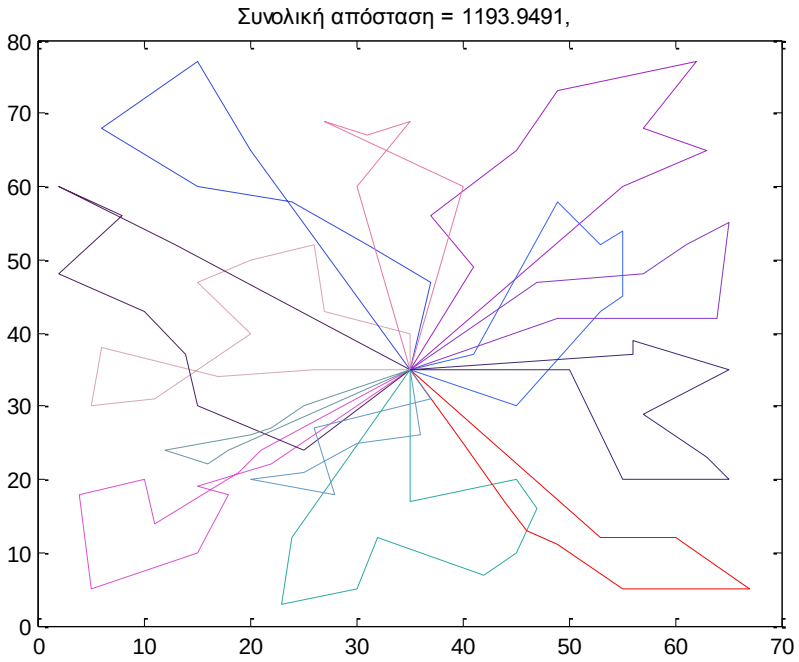


Σχήμα 5.19 Δρομολόγηση οχημάτων R108

Πίνακας δρομολογίων προβλήματος R108

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

5.2.20 Αποτελέσματα προβλήματος R110

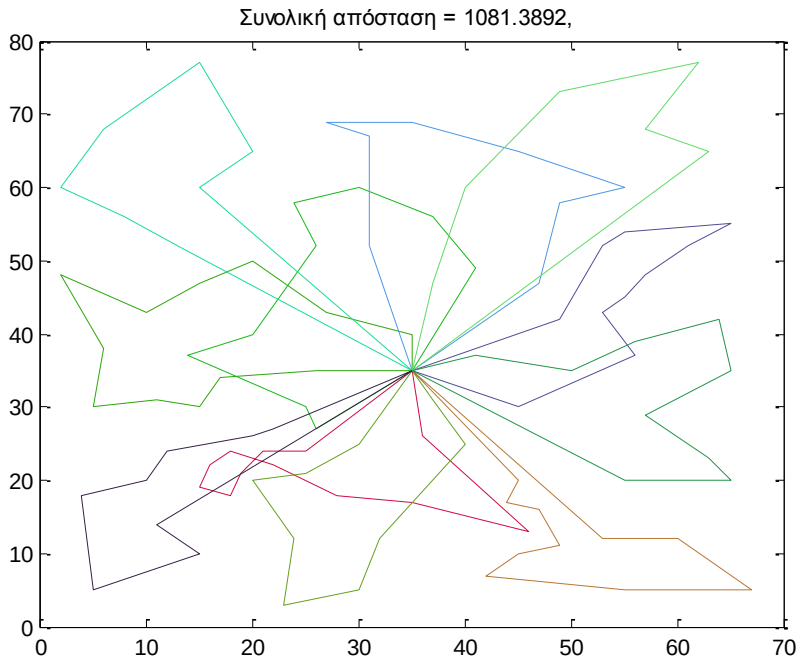


Σχήμα 5.20 Δρομολόγηση οχημάτων R110

Πίνακας δρομολογίων προβλήματος R110

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

5.2.21 Αποτελέσματα προβλήματος R112

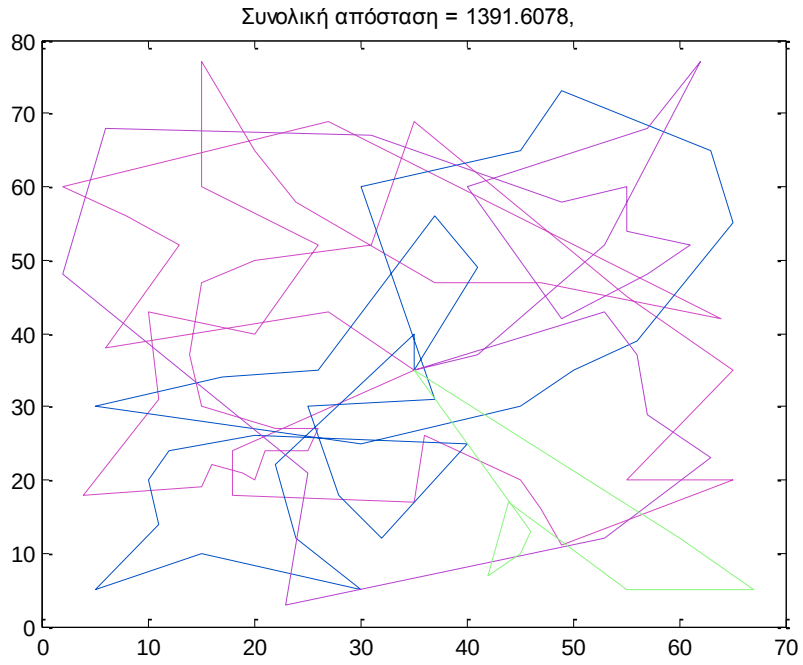


Σχήμα 5.21 Δρομολόγηση οχημάτων R112

Πίνακας δρομολογίων προβλήματος R112

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

5.2.22 Αποτελέσματα προβλήματος R202



Σχήμα 5.22 Δρομολόγηση οχημάτων R202

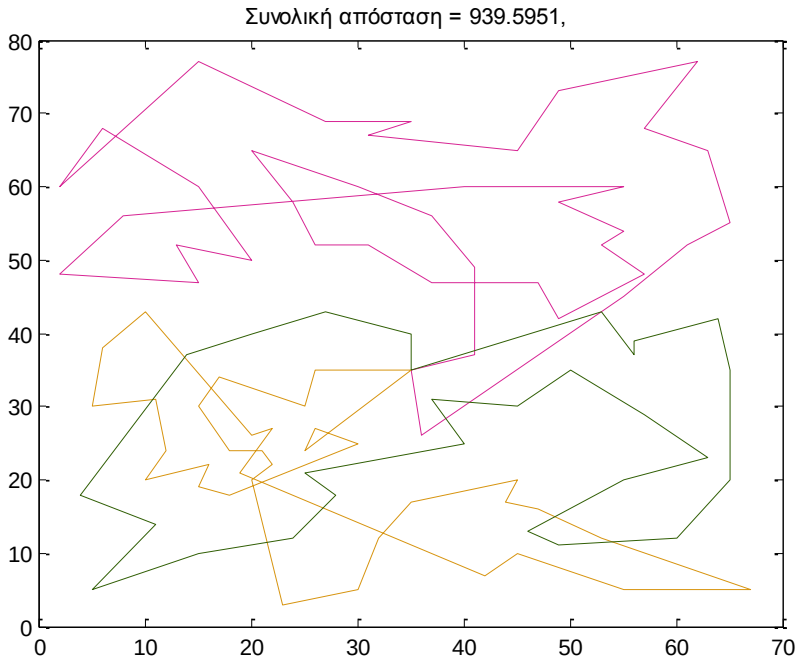
Πίνακας δρομολογίων προβλήματος R202

1	53	46	49	48	37	64	30	51	70	63	12	65	20	89	19	9
1	28	93	43	16	15	39	45	17	62	100	41	58	88	7	54	11
1	29	34	66	72	31	77	80	79	82	10	52	91	50	47	98	44
1	40	68	24	74	42	23	75	1								

85	87	92	86	99	38	60	96	95	97	6	84	83	8	32	33	4
21	67	36	35	69	13	27	14	18	61	90	71	2	1			
57	56	55	81	78	1											

25	5	26	76	73	22	59	3	101	94	1

5.2.23 Αποτελέσματα προβλήματος R204



Σχήμα 5.23 Δρομολόγηση οχημάτων R204

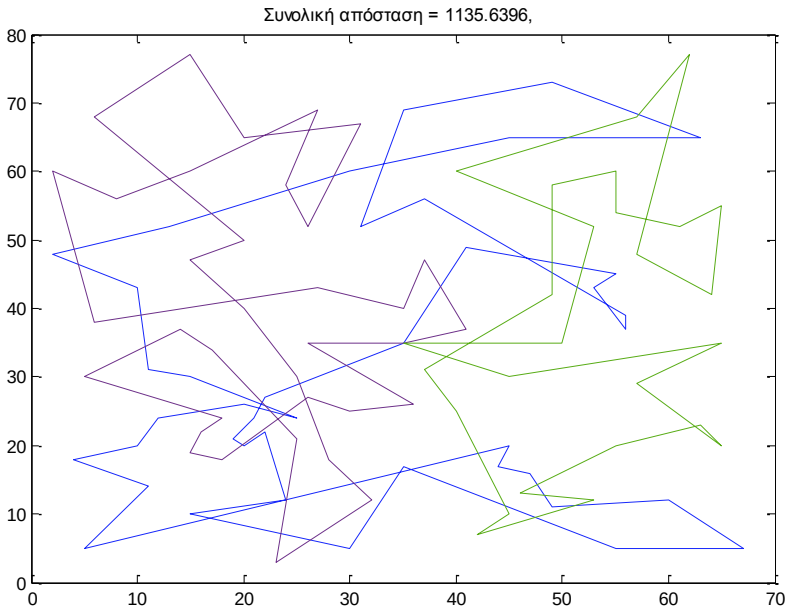
Πίνακας δρομολογίων προβλήματος R204

1	29	2	71	11	12	63	89	32	70	51	77	80	34
1	90	7	61	6	94	60	93	38	44	16	58	3	22
1	28	53	19	84	87	45	39	15	43	88	98	41	54

82	52	10	31	48	47	83	49	8	20	50	37	65	64
74	73	57	68	24	23	42	99	97	100	9	46	18	85
27	13	55	56	5	75	76	40	26	25	30	69	81	78

33	91	21	67	66	72	36	35	79	4	59	1
62	17	86	92	101	14	95	96	1			
1											

5.2.24 Αποτελέσματα προβλήματος R206



Σχήμα 5.24 Δρομολόγηση οχημάτων R206

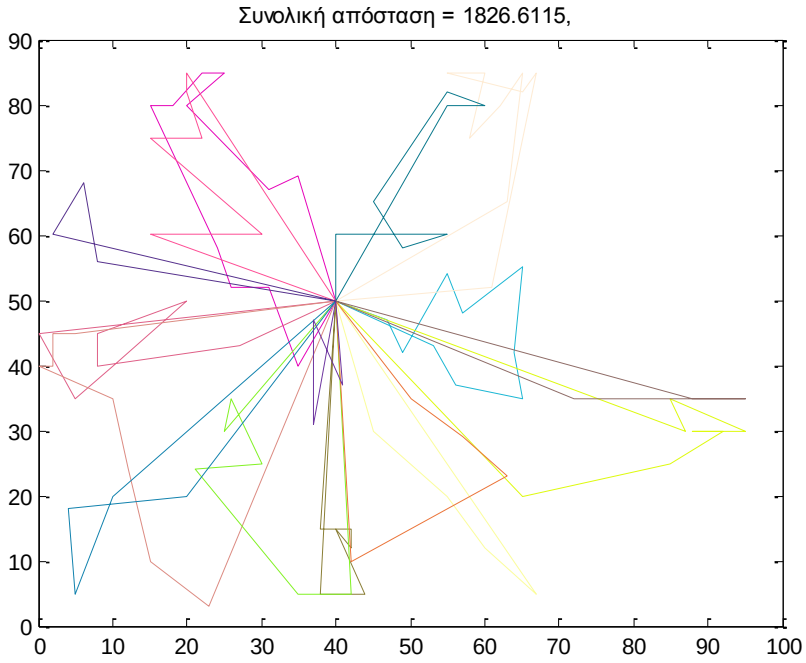
Πίνακας δρομολογίων προβλήματος R206

1	97	60	99	38	93	43	15	16	3	24	68	40	76
1	29	70	28	53	46	37	48	20	64	63	89	91	12
1	13	34	31	72	66	80	30	35	79	82	10	52	77

73	74	22	39	45	87	17	62	100	96	6	85	9	47	49	11
65	50	8	83	19	7	88	58	44	98	61	84	18	94	86	92
54	41	23	42	57	75	5	56	26	55	25	27	1			

21	36	67	33	32	71	51	69	81	78	4	2	1
101	95	14	59	90	1							

5.2.25 Αποτελέσματα προβλήματος RC101

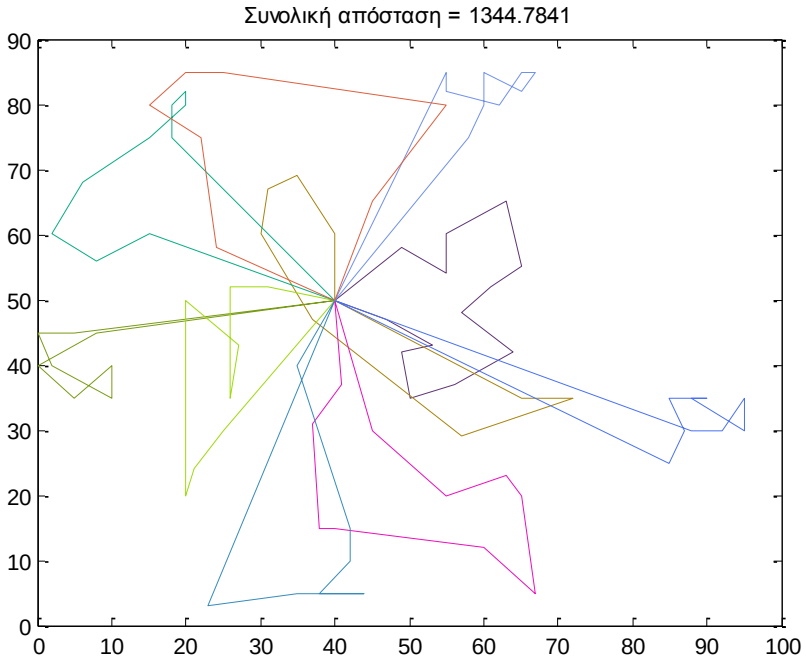


Σχήμα 5.25 Δρομολόγηση οχημάτων RC101

Πίνακας δρομολογίων προβλήματος RC101

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

5.2.26 Αποτελέσματα προβλήματος RC104

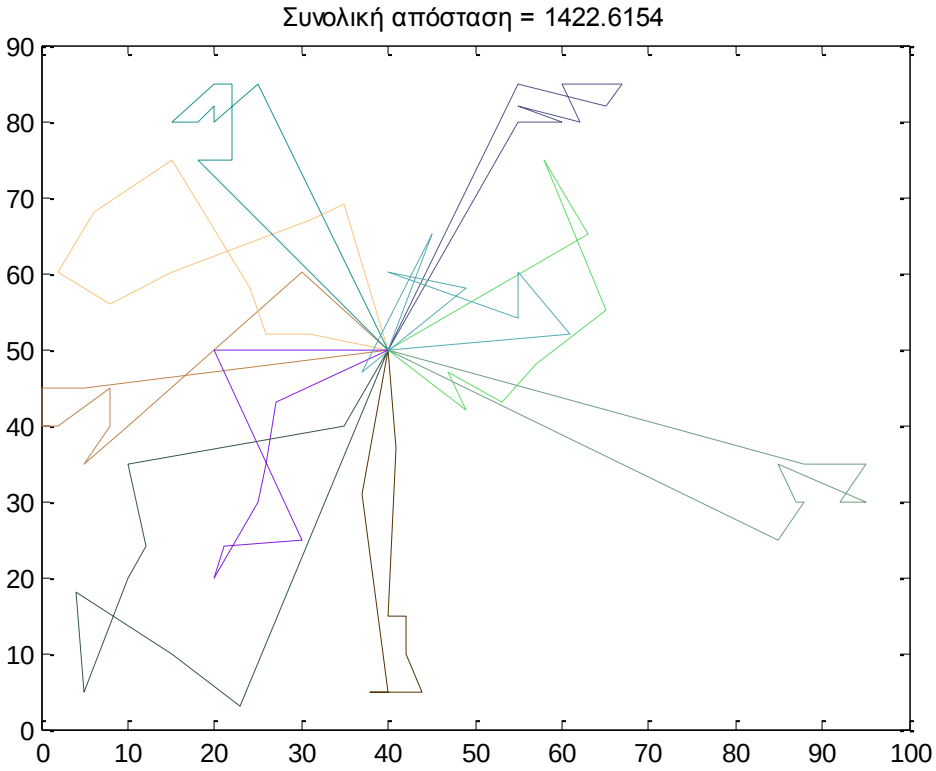


Σχήμα 5.26 Δρομολόγηση οχημάτων RC104

Πίνακας δρομολογίων προβλήματος RC104

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

5.2.27 Αποτελέσματα προβλήματος RC107

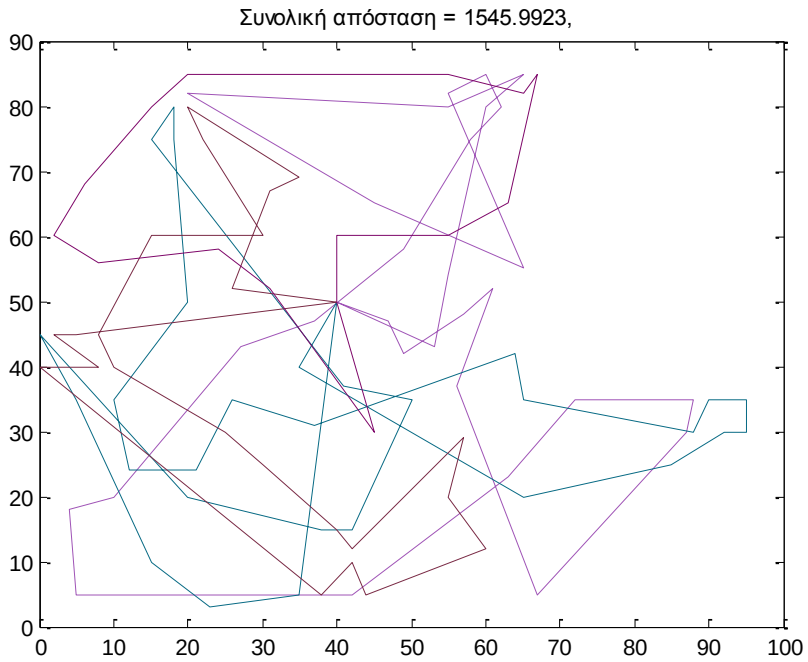


Σχήμα 5.27 Δρομολόγηση οχημάτων RC107

Πίνακας δρομολογίων προβλήματος RC107

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

5.2.28 Αποτελέσματα προβλήματος RC202



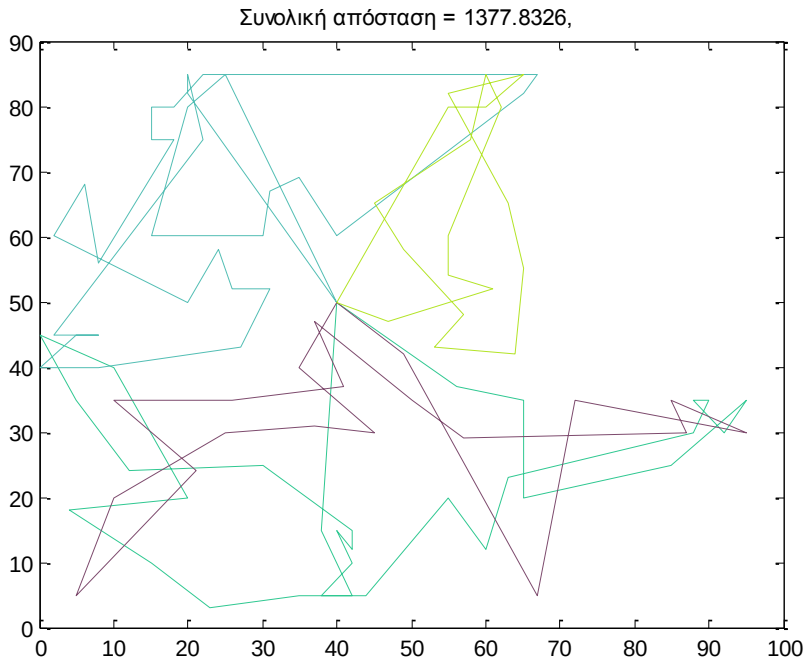
Σχήμα 5.28 Δρομολόγηση οχημάτων RC202

Πίνακας δρομολογίων προβλήματος RC202

1	93	97	40	37	43	46	62	72	45	41	39	42	82	91	83
1	66	64	34	29	27	28	30	31	63	68	84	100	87	88	10
1	15	48	12	16	17	24	20	19	77	52	85	50	23	58	53
1	65	70	89	79	74	80	9	6	4	2	44	38	36	73	55

60	98	76	22	49	86	51	35	32	33	90	96	94	95	92	81	1
54	7	47	8	67	57	21	25	75	18	14	59	78	26	1		
11	13	61	56	3	5	71	101	99	1							
69	1															

5.2.29 Αποτελέσματα προβλήματος RC206



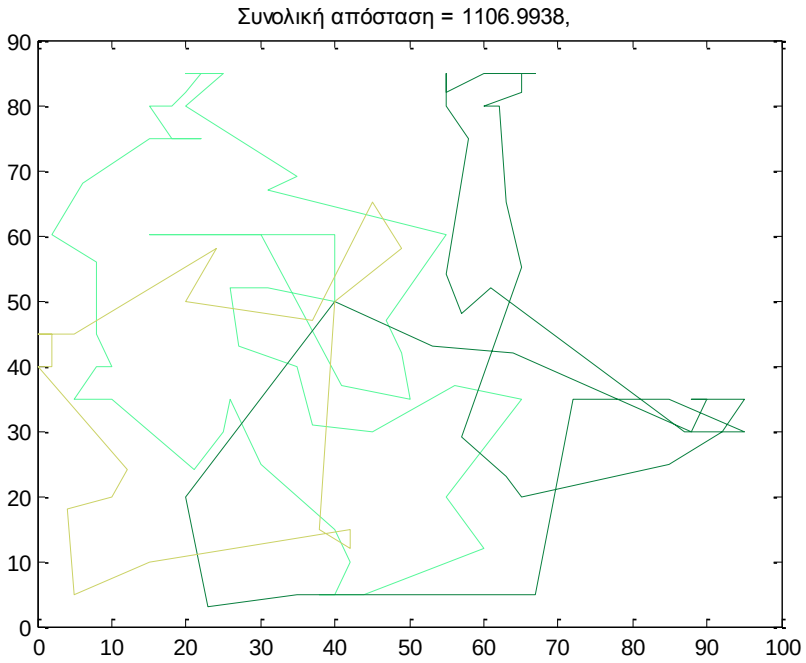
Σχήμα 5.29 Δρομολόγηση οχημάτων RC206

Πίνακας δρομολογίων προβλήματος RC206

1	96	63	64	34	28	29	32	30	31	86	77	52	19	22	24	20	23	50
1	46	6	3	48	13	15	17	16	12	83	70	99	89	54	74	80	79	7
1	66	65	84	53	60	76	87	10	100	67	91	57	85	33	35	27	51	90
1	43	40	37	45	73	72	68	93	95	82	62	42	41	39	55	97	94	81

21	58	88	14	18	11	75	98	59	78	26	49	25	1	
8	9	47	4	44	36	38	69	71	101	56	61	5	2	1
92	1													
1														

5.2.30 Αποτελέσματα προβλήματος RC208



Σχήμα 5.30 Δρομολόγηση οχημάτων RC208

Πίνακας δρομολογίων προβλήματος RC208

1	70	99	83	66	84	65	96	63	52	77	19	24	22	20	23	58	100	53
1	93	68	31	30	32	28	29	34	64	86	85	72	73	39	40	38	37	36
1	82	62	91	54	89	15	18	48	16	17	88	60	98	76	59	21	50	25

87	10	14	12	11	13	79	74	80	8	3	7	9	47	46
41	45	44	43	42	97	95	94	33	27	35	51	90	49	26
1														

4	6	2	5	71	101	55	81	92	57	67	56	61	69	1
78	75	1												

5.3 Πίνακες αποτελεσμάτων

Παραπάνω παρουσιάστηκε το γράφημα και ο πίνακας διαδρομών από ένα δείγμα προβλημάτων. Στη συνέχεια συνοψίζονται όλα τα αποτελέσματα στους παρακάτω πίνακες που περιλαμβάνουν την αρχική εφικτή λύση, τη βέλτιστη λύση που βρέθηκε από τον Tabu Search και η παγκοσμίως βέλτιστη λύση

Πρόβλημα	**Αποτέλεσμα Αλγορίθμου Απληστίας**	**Οχήματα στον αλγόριθμο Απληστίας**	**Κόστος στον αλγόριθμο Tabu Search**	**Οχήματα στον αλγόριθμο Tabu Search**	**Παγκοσμίως βέλτιστη λύση**	**Παγκοσμίως βέλτιστα οχήματα**
C101	855.07	10	852.95	10	828.94	10
C102	1209.39	12	921.54	10	828.94	10
C103	1561.70	16	1015.94	10	828.06	10
C104	1389.94	13	960.00	10	824.78	10
C015	855.07	10	852.95	10	828.94	10
C106	1019.93	10	945.42	10	828.94	10
C107	1029.27	10	884.25	10	828.94	10
C108	1029.27	10	886.46	10	828.94	10
C109	1174.63	10	973.56	10	828.94	10
C201	591.56	3	591.56	3	591.56	3
C202	1268.56	5	872.98	4	591.56	3
C203	1255.00	6	985.63	4	591.17	3
C204	1191.09	5	793.81	4	590.6	3
C205	621.11	3	618.43	3	588.88	3
C206	629.88	3	605.49	3	588.49	3
C207	763.71	3	676.38	3	588.29	3
C208	634.09	3	618.43	3	588.32	3
R101	2265.69	29	1787.86	20	1650.8	19
R102	2419.88	31	1659.67	18	1486.12	17
R103	1889.33	23	1380.00	15	1292.68	13
R104	1571.13	17	1222.92	11	1007.31	9
R105	2057.76	24	1629.23	15	1377.11	14
R106	1924.68	23	1451.78	13	1252.03	12
R107	1718.94	18	1270.60	12	1104.66	10
R108	1447.02	13	1065.39	10	960.88	9
R109	1639.34	17	1296.47	13	1194.73	11
R110	1549.02	16	1193.95	13	1118.84	10
R111	1552.66	16	1166.49	12	1096.72	10
R112	1315.98	13	1081.39	11	982.14	9
R201	1743.92	7	1591.54	4	1252.37	4
R202	1560.36	7	1391.61	4	1191.7	3

R203	1630.46	7	1271.64	3	939.5	3
R204	1257.08	6	939.60	3	825.52	2
R205	1493.91	4	1272.36	4	994.42	3
R206	1556.23	6	1135.64	3	906.14	3
R207	1489.78	5	1053.02	3	890.61	2
R208	1123.63	4	897.15	3	726.82	2
R209	1448.12	5	1169.24	3	909.16	3
R210	1527.82	6	1204.25	3	939.37	3
R211	1326.28	4	1030.80	3	885.71	2
RC101	2587.17	24	1826.61	16	1696.94	14
RC102	2185.74	18	1682.54	14	1554.75	12
RC103	2024.72	17	1335.73	12	1261.67	11
RC104	1729.47	15	1344.78	11	1135.48	10
RC105	2604.61	21	1712.86	15	1629.44	13
RC106	1896.19	14	1560.46	13	1424.73	11
RC107	2215.95	18	1422.62	12	1230.48	11
RC108	2056.94	17	1327.46	12	1139.82	10
RC201	1993.78	8	1703.02	5	1406.94	4
RC202	1822.04	8	1545.99	4	1365.65	3
RC203	1596.81	7	1300.00	3	1049.62	3
RC204	1205.07	6	1057.34	3	798.46	3
RC205	2050.42	10	1476.98	5	1297.65	4
RC206	1522.17	5	1377.83	4	1146.32	3
RC207	1725.90	5	1317.94	4	1061.14	3
RC208	1383.60	4	1106.99	3	828.14	3

Κεφάλαιο 6.
Σύγκριση αποτελεσμάτων και συμπεράσματα.

6.1 Σύγκριση αποτελεσμάτων

Στο προηγούμενο κεφάλαιο παρουσιάστηκαν τα αποτελέσματα των αλγορίθμων απληστίας και περιορισμένης αναζήτησης. Σε αυτό το κεφάλαιο θα επιχειρηθεί αξιολόγηση αυτών των αποτελεσμάτων. Αρχικά θα παρουσιαστούν πίνακες, που θα περιλαμβάνουν συγκριτικά αποτελέσματα ανάμεσα στην αρχική λύση και την βέλτιστη της και ανάμεσα στην βέλτιστη λύση και την παγκοσμίως βέλτιστη. Η σύγκριση των αποτελεσμάτων αναγράφεται σε % απόκλιση από τη βέλτιστη λύση, σε % βελτίωση από την αρχική λύση, και η διαφορά των οχημάτων σε κάθε περίπτωση. Αναλυτικά παρουσιάζονται στο παρακάτω πίνακα.

Πίνακας 6.1

Πρόβλημα	Αποτέλεσμα Αλγορίθμου Απληστίας	Οχήματα στον αλγόριθμο Απληστίας	Κόστος στον αλγόριθμο Tabu Search	Οχήματα στον αλγόριθμο Tabu Search	Παγκοσμίως βέλτιστη λύση	Παγκοσμίως βέλτιστα οχήματα	%απόκλιση	διαφορά οχημάτων από βέλτιστο	%βελτίωση	βελτίωση οχημάτων από αρχική λύση
C101	855.07	10	852.95	10	828.94	10	2.90%	0	0.25%	0
C102	1209.39	12	921.54	10	828.94	10	11.17%	0	23.80%	2
C103	1561.70	16	1015.94	10	828.06	10	22.69%	0	34.95%	6
C104	1389.94	13	960.00	10	824.78	10	16.39%	0	30.93%	3
C015	855.07	10	852.95	10	828.94	10	2.90%	0	0.25%	0
C106	1019.93	10	945.42	10	828.94	10	14.05%	0	7.31%	0
C107	1029.27	10	884.25	10	828.94	10	6.67%	0	14.09%	0
C108	1029.27	10	886.46	10	828.94	10	6.94%	0	13.88%	0
C109	1174.63	10	973.56	10	828.94	10	17.45%	0	17.12%	0
C201	591.56	3	591.56	3	591.56	3	0.00%	0	0.00%	0
C202	1268.56	5	872.98	4	591.56	3	47.57%	1	31.18%	1
C203	1255.00	6	985.63	4	591.17	3	66.72%	1	21.46%	2
C204	1191.09	5	793.81	4	590.6	3	34.41%	1	33.35%	1
C205	621.11	3	618.43	3	588.88	3	5.02%	0	0.43%	0
C206	629.88	3	605.49	3	588.49	3	2.89%	0	3.87%	0
C207	763.71	3	676.38	3	588.29	3	14.97%	0	11.44%	0
C208	634.09	3	618.43	3	588.32	3	5.12%	0	2.47%	0
R101	2265.69	29	1787.86	20	1650.8	19	8.30%	1	21.09%	9
R102	2419.88	31	1659.6	18	1486.12	17	11.68%	1	31.42%	13

			7							
R103	1889.33	23	1380.00	15	1292.68	13	6.75%	2	26.96%	8
R104	1571.13	17	1222.92	11	1007.31	9	21.40%	2	22.16%	6
R105	2057.76	24	1629.23	15	1377.11	14	18.31%	1	20.83%	9
R106	1924.68	23	1451.78	13	1252.03	12	15.95%	1	24.57%	10
R107	1718.94	18	1270.60	12	1104.66	10	15.02%	2	26.08%	6
R108	1447.02	13	1065.39	10	960.88	9	10.88%	1	26.37%	3
R109	1639.34	17	1296.47	13	1194.73	11	8.52%	2	20.92%	4
R110	1549.02	16	1193.95	13	1118.84	10	6.71%	3	22.92%	3
R111	1552.66	16	1166.49	12	1096.72	10	6.36%	2	24.87%	4
R112	1315.98	13	1081.39	11	982.14	9	10.11%	2	17.83%	2
R201	1743.92	7	1591.54	4	1252.37	4	27.08%	0	8.74%	3
R202	1560.36	7	1391.61	4	1191.7	3	16.78%	1	10.81%	3
R203	1630.46	7	1271.64	3	939.5	3	35.35%	0	22.01%	4
R204	1257.08	6	939.60	3	825.52	2	13.82%	1	25.26%	3
R205	1493.91	4	1272.36	4	994.42	3	27.95%	1	14.83%	0
R206	1556.23	6	1135.64	3	906.14	3	25.33%	0	27.03%	3
R207	1489.78	5	1053.02	3	890.61	2	18.24%	1	29.32%	2
R208	1123.63	4	897.15	3	726.82	2	23.43%	1	20.16%	1
R209	1448.12	5	1169.24	3	909.16	3	28.61%	0	19.26%	2
R210	1527.82	6	1204.25	3	939.37	3	28.20%	0	21.18%	3
R211	1326.28	4	1030.80	3	885.71	2	16.38%	1	22.28%	1
RC101	2587.17	24	1826.61	16	1696.94	14	7.64%	2	29.40%	8
RC102	2185.74	18	1682.54	14	1554.75	12	8.22%	2	23.02%	4
RC103	2024.72	17	1335.73	12	1261.67	11	5.87%	1	34.03%	5
RC104	1729.47	15	1344.78	11	1135.48	10	18.43%	1	22.24%	4
RC105	2604.61	21	1712.86	15	1629.44	13	5.12%	2	34.24%	6
RC106	1896.19	14	1560.46	13	1424.73	11	9.53%	2	17.71%	1
RC107	2215.95	18	1422.62	12	1230.48	11	15.61%	1	35.80%	6

RC108	2056.94	17	1327.46	12	1139.82	10	16.46%	2	35.46%	5
RC201	1993.78	8	1703.02	5	1406.94	4	21.04%	1	14.58%	3
RC202	1822.04	8	1545.99	4	1365.65	3	13.21%	1	15.15%	4
RC203	1596.81	7	1300.00	3	1049.62	3	23.85%	0	18.59%	4
RC204	1205.07	6	1057.34	3	798.46	3	32.42%	0	12.26%	3
RC205	2050.42	10	1476.98	5	1297.65	4	13.82%	1	27.97%	5
RC206	1522.17	5	1377.83	4	1146.32	3	20.20%	1	9.48%	1
RC207	1725.90	5	1317.94	4	1061.14	3	24.20%	1	23.64%	1
RC208	1383.60	4	1106.99	3	828.14	3	33.67%	0	19.99%	1

6.2 Ανάλυση αποτελεσμάτων και συμπεράσματα

Από τον παραπάνω πίνακα συγκριτικών αποτελεσμάτων οδηγούμαστε σε κάποιες στατιστικές παρατηρήσεις, βάση των οποίων μπορεί να εκτιμηθεί η ποιότητα των αποτελεσμάτων και εν συνέχεια των αλγορίθμων. Οι κυριότερες στατιστικές παρατηρήσεις είναι:

- Ο μέσος όρος απόκλισης από τη παγκοσμίως βέλτιστη λύση είναι 16.93 %. Το εύρος απόκλισης είναι σε κάποια 0 % το ελάχιστο και 66.72 % το μέγιστο.
- Στα οχήματα χρησιμοποιούμε κατά μ.ο. 0.84 οχήματα παραπάνω από το παγκοσμίως βέλτιστο αριθμό οχημάτων. Το εύρος είναι 0 ο ελάχιστος αριθμός επιπλέον οχημάτων και 3 οχήματα ο μέγιστος.
- Ο μέσος όρος βελτίωσης των αρχικών εφικτών λύσεων του αλγορίθμου απληστίας ανέρχεται στο 20.13 %. Το εύρος κυμαίνεται ανάμεσα στο 0 % βελτίωσης της αρχικής λύσης, και 35.08 % το μέγιστο ποσοστό βελτίωσης της αρχικής λύσης.
- Μέσω του αλγορίθμου Tabu Search, βελτιώσαμε πάρα πολύ τον αριθμό των χρησιμοποιούμενων οχημάτων. Κατά μέσο όρο χρησιμοποιήσαμε 3.18 λιγότερα οχήματα συγκριτικά με την αρχική εφικτή λύση. Σε κάποια προβλήματα δεν βελτιώθηκε ο αριθμός των χρησιμοποιούμενων οχημάτων, ενώ σε ένα πρόβλημα μειώθηκαν συνολικά 13 οχήματα.

Συνολικά, μπορούμε να οδηγηθούμε στο συμπέρασμα ότι ο αλγόριθμος βελτιστοποίησης με τη χρήση της περιορισμένης αναζήτησης είναι αρκετά αποτελεσματικός. Αρχικά η απόκλιση από το παγκοσμίως βέλτιστο είναι σχετικά μικρή. Σε αρκετά προβλήματα έχουμε απόκλιση μικρότερη από 10 %. Τις μεγαλύτερες αποκλίσεις στα αποτελέσματα τις παρατηρούμε στα προβλήματα που εκτελούνται λίγες διαδρομές. Ο αριθμός των διαδρομών έχει ικανοποιητική απόκλιση σε όλα τα αποτελέσματα και πλησιάζει πολύ στο βέλτιστο.

Ο αλγόριθμος Tabu Search ήταν ικανοποιητικός και συγκριτικά με τη βελτίωση των αρχικών εφικτών λύσεων. Βελτίωσε τη λύση σχεδόν σε όλα τα προβλήματα κατά αρκετά ικανοποιητικό ποσοστό. Εκεί που συνέβαλλε πάρα πολύ είναι στην μείωση των οχημάτων. Παρατηρούμε ότι στα περισσότερα προβλήματα που ήταν εφικτή η μείωση των οχημάτων, πραγματοποιήθηκε. Στο τέλος είχαμε αριθμό οχημάτων σε κάθε πρόβλημα, αρκετά κοντά στο παγκοσμίως βέλτιστο.

Συνοπτικά, ο κώδικας ήταν ικανοποιητικός. Είναι σίγουρο όμως ότι με επιδέχεται βελτίωση και αλλαγές για τη βελτίωση του και κατ' επέκταση βελτίωση των αποτελεσμάτων. Μερικές διορθωτικές αλλαγές, θα μπορούσαν να είναι:

- Αύξηση του αριθμού των επαναλήψεων. Ο αλγόριθμος, βασίζεται στην τυχαιότητα, για αυτό σε κάθε τρέξιμο βρίσκουμε και διαφορετικό αποτέλεσμα. Με περισσότερες επαναλήψεις, μπορεί να εξερευνηθούν γειτονιές που δεν είχαν εξερευνηθεί και να βελτιωθεί το αποτέλεσμα.
- Εισαγωγή επιπλέον στρατηγικών στον αλγόριθμο Tabu. Στο συγκεκριμένο πρόβλημα χρησιμοποιείται η μικρή μνήμη, και η εισαγωγή των άλλων δύο μνημών ίσως βελτίωνε το αποτέλεσμα.
- Μείωση της τυχαιότητας στην αναζήτηση γειτονιάς. Στο κομμάτι που εκτελούνται οι ανταλλαγές επιλέγεται τυχαία το σημείο εκκίνησης. Αν αντί για αυτό, επιλεγόταν να γίνει αλλαγή από το τόξο με το μεγαλύτερο κόστος ίσως βελτιωνόντουσαν τα αποτελέσματα.
- Ο αλγόριθμος δεν επιτρέπει μετακίνηση ενός στοιχείου, αλλά των στοιχείων ολόκληρης της γραμμής που απαρτίζουν τη διαδρομή. Αυτό έχει σαν αποτέλεσμα το μέγεθος μίας διαδρομής να παραμένει σταθερό, και αρκετές κινήσεις να απαγορεύονται. Μία προσαρμογή ίσως βελτίωνε το ολικό κόστος.

Βιβλιογραφία

[1] Ιωάννης Μαρινάκης, Αθανάσιος Μυγδαλάς, Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας, Εκδόσεις ‘σοφία’, Θεσσαλονίκη 2008

[2]Sotiris Zygiaris, January 2000 , ‘Supply Chain Management’, INNOREGIO: dissemination of innovation and knowledge management techniques, EC funded project

[3] Douglas M., Lambert ,Martha C. Cooper, Janus D. Pagh, 1998, ‘Supply Chain Management: Implementation Issues and Research Opportunities”, The Inernational journal of Logistics Management, vol. 9No. 2, pp.2

[4] https://www.dhl-discoverlogistics.com/cms/en/course/origin/historical_development.jsp

[5] http://www.logistics.org.gr/4/27/136/

[6] Toth P., Vigo D., 2000, ‘An overview of Vehicle Routing Problems’, Vehicle Routing Problems, Philadelphia, PA, USA, Society for industrial and Applied Mathematics,chapter

[7] http://en.wikipedia.org/wiki/NP-hard

[8] Cheeneebash J., Nadal C., 2010, ‘Using Tabu Search Heuristics in Solving the Vehicle Routing Problem with Time Windows : Application to Mauritian Firm’, Research week 2010,Univercity of Mauritius, Reduit, Mauritius

[9] Ombuki B., Ross B., Hanshar F.,2006, ‘Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows’, Department of Computer Science, Brock Univercity St Catharines.

[10] Hebdar A., Bakr M., 2014, ‘Three Strategies Tabu Search for Vehicle Routing Problem with Time Windows’,Horizon Research Publising

[11] Shaw P., 1998, ‘Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems’, Department of computer science, Univercity of Strathclyde, Glasgow

[12] http://neo.lcc.uma.es/vrp/vrp-flavors/multiple-depot-vrp/

[13] Glover F., 1990, ‘Tabu Search: A Tutorial ‘, Center for Applied Artificial Intelligence, p. 74-94

[14] Glover F. and Laguna M., 1997, ‘Tabu Search, Kluwer Academic Publishers, Boston

[15] Kirkpatrick S., Gelatt Jr. C. D., Vecchi M. P., 1983, “Optimization by Simulated Annealing “, Science, Volume 220, Number 4598

[16] Dueck G., Schuerer T., 1990, ‘Threshold Accepting: A General Purpose Optimization Algorithm, Journal of Computational Physics, 90, 161-175

[17] Dorigo M., Birattari M., Stultze T., 2006, ‘ Ant Colony Optimization: Artificial Arts as a Computational Intelligence Technique’’,Universite Libre de Bruxelles, Belgium

[18]
http://el.wikipedia.org/wiki/%CE%91%CE%BB%CE%B3%CF%8C%CF%81%CE%B9%CE%B8%CE%BC%CE%BF%CE%B9_%CE%B2%CE%B5%CE%BB%CF%84%CE%B9%CF%83%CF%84%CE%BF%CF%80%CE%BF%CE%AF%CE%B7%CF%83%CE%B7%CF%82_%CE%B1%CF%80%CE%BF%CE%B9%CE%BA%CE%B9%CF%8E%CE%BD_%CF%84%CF%89%CE%BD_%CE%BC%CF%85%CF%81%CE%BC%CE%B7%CE%B3%CE%BA%CE%B9%CF%8E%CE%BD

[19]
http://el.wikipedia.org/wiki/%CE%93%CE%B5%CE%BD%CE%B5%CF%84%CE%B9%CE%BA%CE%BF%CE%AF_%CE%91%CE%BB%CE%B3%CF%8C%CF%81%CE%B9%CE%B8%CE%BC%CE%BF%CE%B9

[20] Kennedy J., Eberhart R., 1995, 'Particle Swarm Optimization', Proceedings of 1995 IEEE International Conference of Neural Networks, 4, 1942-948

[21] Solomon M.M., 1987, 'Algorithms for the vehicle routing and scheduling problems with time windows constraints',Operations Research, Vol. 35, No. 2, p. 254-265

[22] Larsen A., 2001, 'The Dynamic Vehicle Routing Problem', Lyngby

[23] Cordeau J. F., Laporte G., 2005, 'Tabu Search Heuristics for the Vehicle Routing Problem', Operations Research/Computer Science Interfaces Series, Volume 30, pp. 145-163