ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΧΡΗΣΗ ΤΗΣ ΔΙΑΔΙΚΑΣΙΑΣ ΑΠΛΗΣΤΗ ΤΥΧΑΙΟΠΟΙΗΜΕΝΗ ΠΡΟΣΑΡΜΟΣΤΙΚΗ ΑΝΑΖΗΤΗΣΗ

(Solving vehicle routing problem with greedy randomized adaptive search procedure)

Κοτζαμπάση Δανάη

Επιβλέπων:Δρ Ιωάννης Μαρινάκης, Επίκουρος Καθηγητής

Χανιά 2015

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον καθηγητή μου κ. Ιωάννη Μαρινάκη για την καθοδήγηση και τη βοήθεια που μου προσέφερε στην εκπόνηση της εργασίας. Επίσης θέλω να ευχαριστήσω τον Ηρακλή-Δημήτριο Ψύχα για την εξίσου πολύτιμη βοήθειά του καθώς και τη κα. Μάγδα Μαρινάκη.

Ακόμα οφείλω να ευχαριστήσω την οικογένειά μου που όλα αυτά τα χρόνια αποτελεί το μεγαλύτερο στήριγμα σε όλους τους τομείς της ζωής μου. Τέλος ευχαριστώ πάρα πολύ τους φίλους μου που βρίσκονται πάντα δίπλα μου.

Περίληψη

Η συγκεκριμένη διπλωματική εργασία έχει ως αντικείμενο την υλοποίηση κάποιων μεθόδων δρομολόγησης οχημάτων που χρησιμοποιούνται για τη διανομή των προϊόντων-αγαθών μιας εφοδιαστικής αλυσίδας. Οι μέθοδοι που χρησιμοποιούνται είναι οι κατασκευαστικοί αλγόριθμοι: Άπληστη τυχαιοποιημένη προσαρμοστική αναζήτηση (greedy randomized adaptive procedure ή GRASP) και πλησιέστερος γείτονας (nearest neighbor). Στόχος και των δύο μεθόδων είναι η κατασκευή μιας διαδρομής τέτοιας ώστε να ικανοποιούνται οι ανάγκες των πελατών που ορίζει κάθε φορά το πρόβλημα και να ελαχιστοποιείται το κόστος που προκύπτει από αυτή. Ταυτόχρονα, λαμβάνονται υπόψη περιορισμοί που αφορούν τη δεδομένη χωρητικότητα των οχημάτων και τον προκαθορισμένο χρόνο που κάθε όχημα επιτρέπεται να δαπανήσει στην εξυπηρέτηση. Ακόμα, για το καθορισμό της επιθυμητής διαδρομής δίνονται ως δεδομένα το πλήθος και οι τοποθεσίες των πελατών, η ζήτηση κάθε πελάτη και ο χρόνος εξυπηρέτησής του. Με τη μέθοδο του πλησιέστερου γείτονα, η επιλογή της σειράς εξυπηρέτησης των πελατών καθορίζεται από την απόσταση που απέχουν κάθε φορά οι πελάτες από το όχημα και επιλέγεται αυτός με την ελάχιστη, εφόσον φυσικά τηρούνται οι περιορισμοί που προαναφέρθηκαν. Με τη GRASP, η επιλογή των πελατών γίνεται τυχαία ανάμεσα σε ένα αριθμό πελατών που απέχουν λιγότερο από τη τρέχουσα θέση του οχήματος σε σχέση με τους υπόλοιπους χωρίς απαραίτητα να επιλέγεται ο πλησιέστερος. Και για τους δύο αλγόριθμους ισχύει ότι ένα όχημα ξεκινάει από την αποθήκη, επισκέπτεται τους πελάτες κατά το τρόπο που ορίζει κάθε φορά η μέθοδος, ελέγχοντας ταυτόχρονα τους περιορισμούς. Εάν αυτοί παραβιάζονται τότε το όχημα επιστρέφει στην αποθήκη και ένα νέο όχημα συνεχίζει την εξυπηρέτηση, ενώ οι αλγόριθμοι τερματίζουν όταν όλοι οι πελάτες έχουν εξυπηρετηθεί και τελικά το όχημα επιστρέφει στην αποθήκη. Έπειτα από τη κατασκευή των διαδρομών ακολουθεί η ανταλλαγή δύο τυχαίων πελατών μέσω της μεθόδου 1-1 ανταλλαγής (1-1 exchange). Η μέθοδος αυτή ανήκει στις τεχνικές τοπικής αναζήτησης και χρησιμοποιείται προκειμένου να ελεγχθεί εάν οι αρχικές λύσεις που έχουν κατασκευαστεί προηγουμένως μπορούν να βελτιωθούν περαιτέρω. Τόσο η μέθοδος GRASP όσο και η 1-1 exchange παράγουν τυχαίες λύσεις οι οποίες είναι πιθανό να διαφέρουν από επανάληψη σε επανάληψη γι αυτό και επιλέγουμε να τις επαναλάβουμε παραπάνω φορές συγκρίνοντας κάθε φορά τα αποτελέσματα που επιστρέφουν και επιλέγοντας αυτό που αποδίδει το ελάχιστο κόστος. Η υλοποίηση των παραπάνω μεθόδων πραγματοποιήθηκε στο προγραμματιστικό περιβάλλον MATLAB για ένα πλήθος διαφορετικών επαναλήψεων και δεδομένων προβλήματος ενώ τα αποτελέσματα που επιστρέφουν παρατίθενται και αναλύονται στο τέλος της εργασίας.

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή

1.1 Εφοδιαστική αλυσίδα

Εφοδιαστική αλυσίδα ορίζεται ως η διαδικασία κατά την οποία ένα σύνολο επιχειρηματικών φορέων όπως προμηθευτές, κατασκευαστές, έμποροι και διανομείς συνεργάζονται με στόχο [1]:

- Τη παραγωγή αρχικών προϊόντων.
- Τη μετατροπή των αρχικών προϊόντων στην επιθυμητή τελική μορφή.
- Τη παράδοση των προϊόντων αυτών στους εμπόρους.

Πρακτικά η ιδέα της εφοδιαστικής αλυσίδας ξεκίνησε από αλλαγές στο κατασκευαστικό περιβάλλον όπως τα αυξανόμενα κόστη παραγωγής, το μειωμένο κύκλο ζωής των προϊόντων και τη παγκοσμιοποίηση της αγοράς. Η εφοδιαστική αλυσίδα μπορεί να διαιρεθεί σε δύο επιμέρους βασικές διαδικασίες [1]:

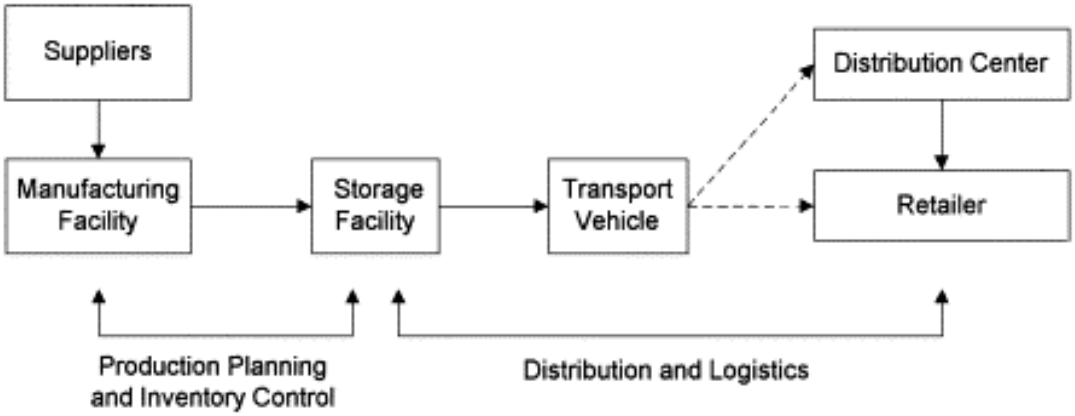
- Το σχεδιασμό παραγωγής και τον έλεγχο αποθεμάτων. (Production planning and inventory control)

Κατά τη διαδικασία αυτή πραγματοποιείται ο σχεδιασμός και η διαχείριση ολόκληρης της κατασκευαστικής διαδικασίας καθώς και των αποθεματικών αρχών και διαδικασιών που αφορούν τις πρώτες ύλες-προϊόντα.

- Τη διανομή και τον εφοδιασμό.(Distribution and logistics)

Η διαδικασία αυτή περιλαμβάνει τη διαχείριση της απόκτησης, μεταφοράς και τελικής παράδοσης των προϊόντων.

Οι δύο παραπάνω διαδικασίες αλληλεπιδρούν μεταξύ τους, προκειμένου να δημιουργήσουν μία ολοκληρωμένη εφοδιαστική αλυσίδα. Ο σχεδιασμός και η διαχείριση τους καθορίζει το ποσοστό κατά το οποίο η εφοδιαστική αλυσίδα αποδίδει σύμφωνα με τους στόχους που έχουν τεθεί. Στο σχήμα που ακολουθεί αναπαριστάται η συνολική διαδικασία που ακολουθεί η εφοδιαστική αλυσίδα ενώ ακόμα διακρίνονται οι επιμέρους διαδικασίες που την απαρτίζουν και οι εμπλεκόμενοι φορείς [1].



Εικόνα 1.1: Εφοδιαστική αλυσίδα

1.2 Μέτρα απόδοσης εφοδιαστικής αλυσίδας

Ένα σημαντικό στοιχείο στο σχεδιασμό και την ανάλυση της εφοδιαστικής αλυσίδας είναι ο καθορισμός των κατάλληλων μέτρων απόδοσης με τα οποία καθορίζεται η αποτελεσματικότητα του συστήματος ενώ ακόμα μπορεί να επιτευχθεί η σύγκρισή του με άλλα ανταγωνιστικά συστήματα. Επιπλέον χρησιμοποιούνται στο σχεδιασμό προτεινόμενων συστημάτων καθορίζοντας τις τιμές των μεταβλητών απόφασης που αποδίδουν περισσότερο. Τα εν λόγω μέτρα μπορούν λάβουν ποιοτικές ή ποσοτικές τιμές όπως φαίνεται στη συνέχεια [1].

1.2.1 Ποιοτικά μέτρα απόδοσης

Η απόδοση των μέτρων αυτών δεν δίδεται απευθείας με αριθμούς, παρόλα αυτά κάποιοι παράμετροί τους μπορούν να λάβουν ποσοτικές τιμές. Τέτοια μέτρα είναι [1]:

- Η ικανοποίηση του πελάτη: Ο βαθμός στον οποίο ο πελάτης θεωρεί ότι τόσο το προϊόν όσο και η εξυπηρέτηση που λαμβάνει ανταποκρίνονται στις προσδοκίες του.
- Ευελιξία: Ο βαθμός στον οποίο η αλυσίδα μπορεί να ανταποκρίνεται σε απρόβλεπτες μεταβολές των απαιτήσεων.
- Ενσωμάτωση πληροφοριών και ροής υλικών: Ο βαθμός στον οποίο όλες οι <<συνιστώσες>> της εφοδιαστικής αλυσίδας ανταλλάζουν πληροφορίες και μεταφέρουν υλικά.
- Αποτελεσματική διαχείριση του ρίσκου: Ο βαθμός στον οποίο επιτυγχάνεται ελαχιστοποίηση των επιπτώσεων από τα ρίσκα που έχουν ληφθεί.
- Απόδοση προμηθευτών: Η συνέπεια με την οποία οι προμηθευτές μεταφέρουν υλικά στις εγκαταστάσεις παραγωγής σε καλή κατάσταση και στο προβλεπόμενο χρόνο.

1.2.2 Ποσοτικά μέτρα απόδοσης

Η απόδοση των μέτρων αυτών σε αντίθεση με των παραπάνω, μετράται με αριθμητικές τιμές και μπορούν να διαχωριστούν σε: α) μέτρα απόδοσης που αφορούν το κέρδος και τα κόστη, β) μέτρα απόδοσης που αφορούν την ανταπόκριση των πελατών.

Μέτρα με γνώμονα το κόστος [1]:

- Ελαχιστοποίηση του κόστους: Το κόστος μπορεί ελαχιστοποιείται είτε συνολικά (ολικό κόστος) είτε για συγκεκριμένα τμήματα της αλυσίδας.
- Μεγιστοποίηση των πωλήσεων: Μεγιστοποίηση των εσόδων από τις πωλήσεις ή του αριθμού των πωληθέντων μονάδων προϊόντων.
- Μεγιστοποίηση του κέρδους: Προκύπτει από το συνδυασμό της μεγιστοποίησης των πωλήσεων και της ελαχιστοποίησης του κόστους που προαναφέρθηκαν.
- Ελαχιστοποίηση του κόστους αποθεμάτων: Περιλαμβάνει τα κόστη των προϊόντων και της αποθήκευσής τους.
- Μεγιστοποίηση της επιστροφής χρημάτων επένδυσης: Μεγιστοποίηση του ρυθμού με τον οποίο το δίκτυο αποδίδει κέρδος στον επενδυτή.

Μέτρα με γνώμονα την ανταπόκριση των πελατών [1]:

- Μεγιστοποίηση παραγγελιών ανά μονάδα χρόνου.
- Ελαχιστοποίηση του χρόνου αναμονής: Ελαχιστοποίηση της απόκλισης του πραγματικού χρόνου παράδοσης από τη προβλεπόμενη χρονική στιγμή.
- Ελαχιστοποίηση του χρόνου ανταπόκρισης των παραγγελιών: Ελαχιστοποίηση του χρόνου από τη στιγμή που λαμβάνεται μία παραγγελία μέχρι τη παράδοση της στον πελάτη.
- Ελαχιστοποίηση του χρόνου επεξεργασίας των προϊόντων.
- Ελαχιστοποίηση των πανομοιότυπων λειτουργιών: Ελαχιστοποίηση των λειτουργιών που παρέχονται από περισσότερες από μία επιχειρηματικές μονάδες.

Κεφάλαιο 2:Δρομολόγηση οχημάτων

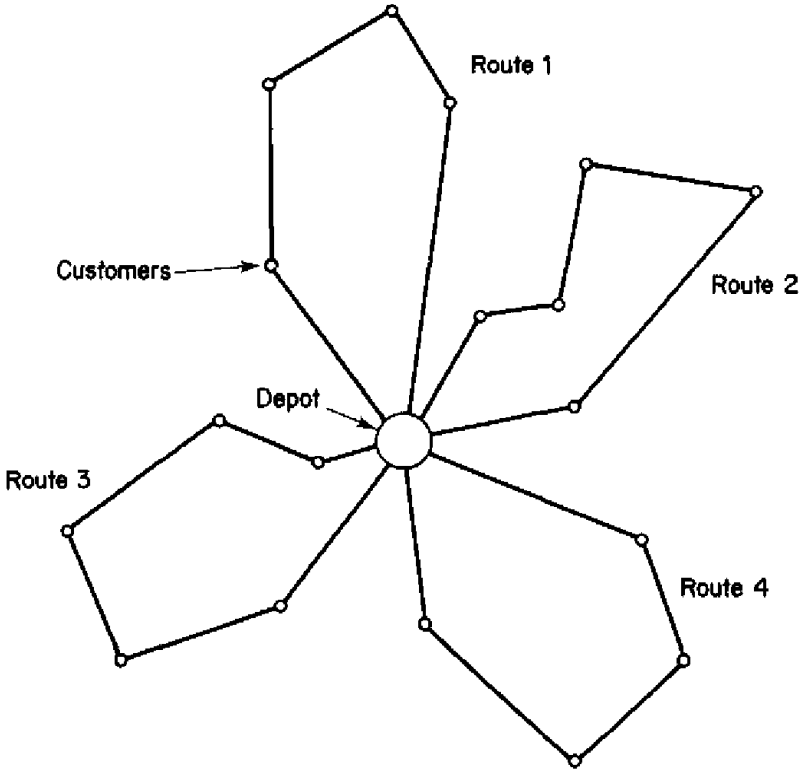
2.1 Πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem ή VRP)

Το πρόβλημα αυτό αφορά κατά γενικό λόγο τη διανομή προϊόντων μεταξύ αποθηκών και πελατών. Η διανομή των αγαθών πραγματοποιείται σε συγκεκριμένα χρονικά πλαίσια όπου μια ομάδα οχημάτων , τα οποία βρίσκονται σε μία ή περισσότερες αποθήκες, καλείται μέσω κατάλληλης διαδρομής να εξυπηρετήσει μία ομάδα πελατών. Συγκεκριμένα στόχος του προβλήματος δρομολόγησης οχημάτων είναι ο καθορισμός του συνόλου των διαδρομών, κάθε μία από τις οποίες αναλαμβάνει ένα μόνο όχημα το οποίο ξεκινάει και καταλήγει στη δική του αποθήκη, και πληρώντας όλους τους περιορισμούς θα ικανοποιούνται οι ανάγκες των πελατών ενω το συνολικό κόστος μεταφοράς θα ελαχιστοποιείται.

Το οδικό δίκτυο, μέσω του οποίου γίνονται οι μεταφορές, συνήθως αναπαριστάται από ένα γράφημα του οποίου τα τόξα ορίζουν τμήματα της διαδρομής ενώ οι κορυφές στις οποίες αυτά διασταυρώνονται αναπαριστούν τους πελάτες και τις αποθήκες του προβλήματος. Τα τόξα μπορεί να έχουν ή να μην έχουν προσανατολισμό αλλά αυτό εξαρτάται από το είδος του δρόμου τον οποίο αναπαριστούν (μονόδρομος ή διπλής κατεύθυνσης). Ακόμη χαρακτηρίζονται από ένα κόστος (μήκος του τόξου) το οποίο σχετίζεται με τον απαιτούμενο χρόνο διαδρομής [18].

Οι διαδρομές που εκτελούν τα οχήματα αρχίζουν και καταλήγουν σε μία ή περισσότερες αποθήκες. Κάθε αποθήκη χαρακτηρίζεται από τον αριθμό και το είδος των οχημάτων που τη χρησιμοποιούν καθώς και από τη χωρητικότητά της σε προϊόντα. Σε ορισμένες πραγματικές εφαρμογές δρομολόγησης οχημάτων, έχει γίνει εκ των προτέρων διαμοιρασμός των πελατών σε αποθήκες, και τα οχήματα πρέπει να επιστρέφουν στην αποθήκη εκκίνησης στο τέλος κάθε διαδρομής. Σε αυτές τις περίπτωσης το πρόβλημα μπορεί να διαιρεθεί σε επιμέρους ανεξάρτητα προβλήματα, κάθε ένα από τα οποία είναι συνδεδεμένο με διαφορετική αποθήκη [18].

Παρακάτω φαίνεται η λύση ενός βασικού παραδείγματος VRP, το οποίο όμως δε θα μπορούσε να αποδώσει στα περισσότερα αληθινά προβλήματα αφού παραλείπει πολλούς περιορισμούς και απαιτήσεις που θα συναντούσαμε στη πραγματικότητα, μέσω αναπαράστασης από ένα μη προσανατολισμένο γράφημα G=(V,A):



Εικόνα 2.1: Αναπαράσταση της λύσης ενός παραδείγματος δρομολόγησης οχημάτων

Έστω ότι i=0 είναι η κεντρική αποθήκη ή το κέντρο διανομής. Τότε τα i=2,....,n θα αναπαριστούν τους πελάτες του προβλήματος Όλα τα i=0,...n αποτελούν τους κόμβους του γραφήματος με *V= {0,……, n}* το σύνολο των κόμβων Θεωρούμε ότι κάθε πελάτης i έχει q_i ζήτηση ποσότητα προϊόντων και το κόστος μετάβασης από τον πελάτη i στον j ορίζετε ως c_{ij} και αναφέρεται στη χωρητικότητα του τόξου (i,j) με A το σύνολο των τόξων. Εάν η εταιρία διαθέτει K οχήματα που εκτελούν τις μεταφορές, η χωρητικότητα κάθε οχήματος θα είναι Q_k. Τέλος, σε κάθε όχημα θα αντιστοιχεί μια διαδρομή η οποία θα ξεκινάει και θα καταλήγει στο κέντρο διανομής [14].

Έτσι για την επίλυση πιο σύνθετων προβλημάτων VRP λαμβάνεται υπόψη ένα πλήθος από χαρακτηριστικά τα οποία μπορεί να έχουν οι πελάτες ή τα οχήματα. Παρακάτω παρουσιάζονται κάποια από αυτά:

Τα τυπικά χαρακτηριστικά των πελατών είναι [18]:

- Οι κορυφές του γραφήματος στις οποίες βρίσκονται οι πελάτες.
- Η ποσότητα των αγαθών, πιθανόν διαφορετικών ειδών, τα οποία πρέπει να παραδοθούν σε αυτούς ή να συλλεχθούν.
- Η χρονική περίοδος (time window) κατά την οποία μπορεί να πραγματοποιηθεί η εξυπηρέτηση του πελάτη.
- Το πλήθος των φορτώσεων ή εκφορτώσεων που πρέπει να πραγματοποιηθούν στη τοποθεσία που βρίσκεται ο πελάτης (συνήθως εξαρτώνται από το είδος των οχημάτων).
- Το σύνολο των διαθέσιμων οχημάτων που μπορούν να χρησιμοποιηθούν για να εξυπηρετηθεί ο πελάτης.

Τα τυπικά χαρακτηριστικά των οχημάτων είναι [18]:

- Η αποθήκη εκκίνησης του οχήματος και η πιθανότητα να τελειώσει την εξυπηρέτησή του σε κάποια άλλη αποθήκη.
- Η χωρητικότητα του οχήματος, η οποία εκφράζεται ως το μέγιστο βάρος ή ο όγκος που ένα όχημα μπορεί να μεταφέρει.
- Η πιθανή υποδιαίρεση του οχήματος σε τμήματα, κάθε ένα από τα οποία χαρακτηρίζεται από τη χωρητικότητα του και το είδος των προϊόντων που μεταφέρει.
- Το υποσύνολο των τόξων του γραφήματος τα οποία μπορούν να διασχίσουν τα οχήματα.
- Το συσχετιζόμενο με τη χρήση του οχήματος κόστος (ανά μονάδα απόστασης, ανά μονάδα χρόνου, ανά δρόμο, κ.λ.π.).

Στόχοι του προβλήματος [18]:

- Ελαχιστοποίηση του συνολικού κόστους μεταφοράς, το οποίο εξαρτάται από τη συνολική απόσταση που καλύπτουν τα οχήματα (ή το συνολικό χρόνο που σπαταλούν) και από τα προκαθορισμένα κόστη τα οποία αφορούν τη χρήση των οχημάτων και τους οδηγούς τους.
- Ελαχιστοποίηση του αριθμού των οχημάτων που χρησιμοποιούνται στην εξυπηρέτηση των πελατών.
- Εξισορρόπηση των διαδρομών λαμβάνοντας υπόψη τους χρόνους μετάβασης και τα μεταφερόμενα φορτία.
- Ελαχιστοποίηση των σφαλμάτων που συνδέονται με τη μερική εξυπηρέτηση πελατών.

2.1.1Το πρόβλημα του πλανόδιου πωλητή (Traveling Salesman Problem ή TSP)

Το πρόβλημα που αντιμετωπίζει ο πωλητής είναι το εξής: Ο πωλητής ξεκινάει από τη πόλη του και έχει στη κατοχή του μία λίστα από πόλεις τις οποίες πρέπει να επισκεφτεί. Επιπλέον γνωρίζει όλες τις αποστάσεις μεταξύ των συγκεκριμένων πόλεων και στόχος του είναι να τις επισκεφτεί όλες ακριβώς μία φορά, με σειρά η οποία να ελαχιστοποιεί τη συνολική απόσταση του ταξιδιού του, και μετά να επιστρέψει στη πόλη του. [14]
Το πρόβλημα του πλανόδιου πωλητή αναφέρεται αφού το μοντέλο του χρησιμοποιείται από τους περισσότερους αλγόριθμους του προβλήματος δρομολόγησης οχημάτων στο σημείο του καθορισμού της σειράς με την οποία θα επισκεφτεί κάθε όχημα τους πελάτες που έχει αναλάβει. Συγκεκριμένα στη CVRP, η οποία είναι η απλούστερη μορφή VRP και θα εξεταστεί αναλυτικότερα παρακάτω, εάν η χωρητικότητα των οχημάτων είναι αρκετά μεγάλη τότε η επίλυση του προβλήματος ταυτίζεται με την απλή επίλυση του TSP[4]. Άλλες εφαρμογές που βρίσκει είναι: η καλωδίωση ηλεκτρονικών υπολογιστών, η ακολουθία εργασιών, η ομαδοποίηση των δεδομένων ενός πίνακα κ.α. [14].

Βασική μορφοποίηση του προβλήματος δρομολόγησης οχημάτων:

Έστω [9] ότι τα x_{ijk} και y_{ijk} παίρνουν τις τιμές:

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

$$\mathbf{y_{ijk}} = \begin{cases} 1, \text{εάν ο πελάτης i δέχεται επίσκεψη από το όχημα k} \\ 0, \text{αλλιώς} \end{cases}$$

Η αντικειμενική συνάρτηση που πρέπει να ελαχιστοποιηθεί είναι:

$$\min \sum_{i,j} c_{ij} \sum_{k} x_{ijk} \qquad \textbf{(2.1)}$$

Υπό

$$\sum_{k} y_{ik} = \begin{cases} 1, & i=2,\ldots,n \\ m, & i=1 \end{cases} \qquad \textbf{(2.2)}$$

$$\sum_{i} q_i y_{ij} \leq Q_k, \qquad k=1,\ldots,m \qquad \textbf{(2.3)}$$

$$\sum_{i} x_{ijk} = \sum_{j} x_{ijk} = y_{ijk}, \qquad i=1,\ldots,n \qquad k=1,\ldots,m \qquad \textbf{(2.4)}$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1, \qquad \text{για όλα τα } S \subseteq \{2,\ldots,n\}, \quad k=1,\ldots,m \qquad \textbf{(2.5)}$$

$$y_{ijk} \in \{0,1\} \qquad i=1,\ldots,n \qquad k=1,\ldots,m \qquad \textbf{(2.6)}$$

$$x_{ijk} \in \{0,1\} \qquad i,j=1,\ldots,n \qquad k=1,\ldots,m \qquad \textbf{(2.7)}$$

Ο περιορισμός **(2.2)** αναφέρεται στο γεγονός ότι κάθε πελάτης αναλαμβάνεται από κάποιο όχημα ενώ την αποθήκη την επισκέπτονται όλα.

Ο περιορισμός **(2.3)** αναφέρεται στους περιορισμούς χωρητικότητας των οχημάτων.

Ο περιορισμός **(2.4)** διασφαλίζει ότι όταν ένα όχημα επισκέπτεται ένα πελάτη μετά αποχωρεί από τη τοποθεσία του.

Ο περιορισμός **(2.5)** διασφαλίζει ότι σε μία διαδρομή το όχημα δε θα εξυπηρετεί λιγότερους πελάτες από όσους του δίνεται η δυνατότητα να εξυπηρετήσει.

2.1.2 VRP με περιορισμένη χωρητικότητα (Capacitated VRP ή CVRP)

Πρόκειται για την απλούστερη κατηγορία VRP στην οποία όλοι οι πελάτες εξυπηρετούνται. Οι απαιτήσεις τους είναι προκαθορισμένες, γνωστές εκ των προτέρων και δεν υπάρχει η δυνατότητα διάσπασής τους. Τα οχήματα είναι όμοια και έχουν ως βάση τους μία κεντρική αποθήκη ενώ ο μόνος περιορισμός που τίθεται είναι αυτός της χωρητικότητάς τους. Στόχος του προβλήματος είναι η ελαχιστοποίηση του συνολικού κόστους και η εξυπηρέτηση όλων των πελατών.

Συγκεκριμένα η CVRP στοχεύει στην εύρεση ενός αριθμού K απλών κυκλικών διαδρομών οι οποίοι θα αποδίδουν το ελάχιστο κόστος. (ως άθροισμα των επιμέρους κοστών των τόξων που συμπεριλαμβάνονται σε αυτούς.) Ταυτόχρονα θα ισχύουν τα παρακάτω [18]:

- Κάθε κυκλική διαδρομή περνάει από τον κόμβο στον οποίο ορίζεται η αποθήκη.
- Κάθε κόμβος πελάτη δέχεται επίσκεψη από μία μόνο κυκλική διαδρομή.
- Το άθροισμα των φορτίων που απαιτούνται για παράδοση σε κάθε κόμβο δεν υπερβαίνει τη χωρητικότητα του οχήματος (ανά κυκλική διαδρομή).

Η μορφοποίηση του CVRP είναι αυτή που προαναφέρθηκε προηγουμένως σύμφωνα με τους Fisher και Jaikumar αφού όπως είπαμε αποτελεί την απλούστερη περίπτωση VRP.

Μία παραλλαγή CVRP είναι η VRP με περιορισμούς ως προς την απόσταση (Distance - Constrained VRP ή **DVRP**). Η κύρια διαφορά της από τη CVRP είναι ότι για κάθε διαδρομή ο περιορισμός που αφορούσε τη χωρητικότητα αντικαθίσταται από ένα περιορισμό για το μέγιστο μήκος (χρόνος). Συγκεκριμένα ορίζεται ένα μήκος για κάθε τόξο (έστω t_{ij}) και το συνολικό μήκος διαδρομής, δηλαδή το άθροισμα όλων των επιμέρους μηκών των τόξων τα οποία συμπεριλαμβάνει, δε πρέπει να ξεπερνάει το μέγιστο μήκος (έστω T). Εάν τα οχήματα είναι διαφορετικά τότε διαφοροποιούνται και τα μέγιστα μήκη (T_k, όπου το k αφορά το είδος οχήματος). Στόχος του εν λόγω προβλήματος είναι η ελαχιστοποίηση του συνολικού μήκους ή χρόνου διαδρομής [18].

2.1.3 VRP με χρονικά παράθυρα (VRP with time windows ή VRPTW)

Αποτελεί μία επέκταση του CVRP στην οποία επιβάλλονται περιορισμοί χωρητικότητας και κάθε πελάτης σχετίζεται με ένα χρονικό διάστημα $[a_i , b_i]$ το οποίο καλείται χρονικό παράθυρο.

Δεδομένα του προβλήματος είναι [18]:

- Η χρονική στιγμή κατά την οποία τα οχήματα εγκαταλείπουν την αποθήκη.
- Ο χρόνος ταξιδίου t_{ij} για κάθε τόξο (i , j) το οποίο ανήκει στο A (το σύνολο τόξων του γραφήματος).
- Ο χρόνος εξυπηρέτησης κάθε πελάτη s_i.

Η εξυπηρέτηση του πελάτη πρέπει να αρχίζει σε χρονική στιγμή που να συμπεριλαμβάνεται στο χρονικό παράθυρο του και το όχημα θα παραμένει στη τοποθεσία του πελάτη για διάστημα s_i. Έτσι εάν το όχημα φτάσει νωρίτερα από τη χρονική στιγμή a_i θα περιμένει για να ξεκινήσει την εξυπηρέτηση.

Το VRPTW στοχεύει στην εύρεση Κ αριθμού κυκλικών διαδρομών με το ελάχιστο δυνατό κόστος ενώ ταυτόχρονα θα ισχύουν [18]:

- Κάθε κυκλική διαδρομή περνάει από τον κόμβο στον οποίο ορίζεται η αποθήκη.
- Κάθε κόμβος πελάτη δέχεται επίσκεψη από μία μόνο κυκλική διαδρομή.
- Το άθροισμα των φορτίων που απαιτούνται για παράδοση σε κάθε κόμβο δεν υπερβαίνει τη χωρητικότητα του οχήματος (ανά κυκλική διαδρομή).
- Για κάθε πελάτη i η εξυπηρέτηση ξεκινάει μέσα στο χρονικό διάστημα $[a_i , b_i]$ και το όχημα παραμένει για χρονικό διάστημα s_i.

Παρατηρούμε ότι οι 3 πρώτες προϋποθέσεις συμπίπτουν με αυτές του CVRP ενώ προστίθεται μία επιπλέον προϋπόθεση η οποία αφορά τα χρονικά παράθυρα. Το γεγονός αυτό είναι λογικό αφού όπως προαναφέρθηκε το VRPTW είναι μία επέκταση του CVRP. Ακόμη θα μπορούσε κανείς να πει ότι αποτελεί μία γενίκευση του CVRP στη περίπτωση όπου το a_i ισούται με το μηδέν και το b_i με το άπειρο για κάθε i το οποίο ανήκει στο $V \setminus \{0\}$ (όπου V το σύνολο των κορυφών του γραφήματος).

2.1.4 VRP με δύο είδη πελατών (VRP with backhauls ή VRPB)

Το VRPB αποτελεί επίσης μία επέκταση του CVRP όπου οι πελάτες του γενικότερου συνόλου $V \setminus \{0\}$ χωρίζονται σε δύο υποσύνολα. Εάν υποθέσουμε ότι οι πελάτες του πρώτου υποσυνόλου (L) έχουν πλήθος n, τότε L= {1,....,n} και καθένας από αυτούς επιθυμεί να παραλάβει μία συγκεκριμένη ποσότητα προϊόντων. Αντίστοιχα αν για τους πελάτες της δεύτερης κατηγορίας υποθέσουμε ότι έχουν πλήθος m, τότε για το δεύτερο υποσύνολο θα ισχύει B= {n+1,....,n+m} και αντίθετα με τους πρώτους πελάτες, οι συγκεκριμένοι επιθυμούν να παραδώσουν στα οχήματα μια συγκεκριμένη ποσότητα προϊόντων. Η ποσότητα αυτή και στις δύο περιπτώσεις ορίζεται ως d_i για κάθε πελάτη i (φυσικά για την αποθήκη ισχύει $d_0=0$) και πρέπει να σημειωθεί ότι για τις διαδρομές που εξυπηρετούν και τα δύο είδη πελατών, οι πρώτοι θα έχουν πάντα προτεραιότητα από τους δεύτερους [18].

Το VRPB στοχεύει στην εύρεση Κ αριθμού κυκλικών διαδρομών με το ελάχιστο δυνατό κόστος ενώ ταυτόχρονα θα ισχύουν [18]:

- Κάθε κυκλική διαδρομή περνάει από τον κόμβο στον οποίο ορίζεται η αποθήκη.
- Κάθε κόμβος πελάτη δέχεται επίσκεψη από μία μόνο κυκλική διαδρομή.
- Οι συνολικές απαιτήσεις των πελατών (ξεχωριστά για κάθε κατηγορία) που περιλαμβάνονται στη κυκλική διαδρομή δε θα πρέπει να ξεπερνούν τη χωρητικότητα του οχήματος.
- Σε κάθε διαδρομή εξυπηρετούνται πρώτα οι πελάτες της πρώτης κατηγορίας και μετά της δεύτερης.

- Δεν επιτρέπονται διαδρομές οι οποίες περιλαμβάνουν πελάτες μόνο της δεύτερης κατηγορίας.

Παρατηρούμε λοιπόν ότι οι τρεις πρώτες προϋποθέσεις ταυτίζονται με αυτές της CVRP, με μία τροποποίηση στη τρίτη έτσι ώστε να προσαρμόζεται στα δεδομένα του VRPB. Ακόμη έχουν προστεθεί δύο επιπλέον προϋποθέσεις, ως επακόλουθο της επέκτασης του CVRP, οι οποίες σχετίζονται αποκλειστικά με τον εν λόγω πρόβλημα.

2.1.5 VRP με διανομή και παραλαβή προϊόντων (VRP with pickup and delivery ή VRPPD)

Στην απλή μορφή αυτού του προβλήματος, κάθε πελάτης i έχει συγκεκριμένη ποσότητα προϊόντων που θέλει να παραλάβει (d_i) και συγκεκριμένη ποσότητα όμοιων προϊόντων να παραδώσει (p_i). Αξίζει να σημειωθεί ότι η σειρά με την οποία εκτελείται η διαδικασία αυτή από το πελάτη είναι πρώτα παραλαβή και έπειτα παράδοση. Επομένως ο φόρτος του οχήματος κατά τη στιγμή της άφιξής του στο πελάτη θα προκύπτει ως το άθροισμα όλων των παραλαβών που έχουν προηγηθεί μαζί με την αρχική φόρτωσή του από την αποθήκη μείων τις παραδώσεις που έχει πραγματοποιήσει. Ακόμη, επειδή τα αγαθά όπως είπαμε είναι όμοια μπορεί να χρησιμοποιηθεί μόνο μία ποσότητα για το κάθε πελάτη d_i όπου $d_i = d_i - p_i$. Η ποσότητα αυτή θα είναι θετική εάν ο πελάτης παραλαμβάνει περισσότερα προϊόντα από αυτά που παραδίδει και αρνητική στην αντίθετη περίπτωση[21]. Τέλος καθορίζονται δύο κόμβοι O_i και D_i:

- Ο O_i αναφέρεται στο κόμβο προέλευσης των προϊόντων που πρέπει να διανεμηθούν στον πελάτη.
- Ο D_i αναφέρεται στο προορισμό των προϊόντων που έχουν περισυλλεχθεί από τον πελάτη.

Σε ορισμένες περιπτώσεις, οι δύο αυτοί κόμβοι μπορεί να ταυτίζονται μεταξύ τους.

Το VRPPD στοχεύει στην εύρεση K αριθμού κυκλικών διαδρομών με το ελάχιστο δυνατό κόστος ενώ ταυτόχρονα θα ισχύουν [21]:

- Κάθε κυκλική διαδρομή περνάει από τον κόμβο στον οποίο ορίζεται η αποθήκη.
- Κάθε κόμβος πελάτη δέχεται επίσκεψη από μία μόνο κυκλική διαδρομή.
- Η φόρτωση του οχήματος πρέπει να είναι μη αρνητική και να μη ξεπερνάει τη μέγιστη χωρητικότητα του οχήματος (για κάθε κυκλική διαδρομή).
- Για κάθε πελάτη i ισχύει ότι ο κόμβος O_i, αν δεν αναφέρεται στην αποθήκη, έχει προτεραιότητα εξυπηρέτησης από τον πελάτη i ο οποίος θα εξυπηρετηθεί μετά στην ίδια διαδρομή.

- Για κάθε πελάτη i ισχύει ότι ο κόμβος D_i, αν δεν αναφέρεται στην αποθήκη, εξυπηρετείται μετά από τον πελάτη i και στην ίδια διαδρομή.

Το VRPPD αποτελεί μία γενίκευση του CVRP στη περίπτωση που $O_i= D_i=0$ και $p_i=0$ για κάθε $i \in V$ [18].

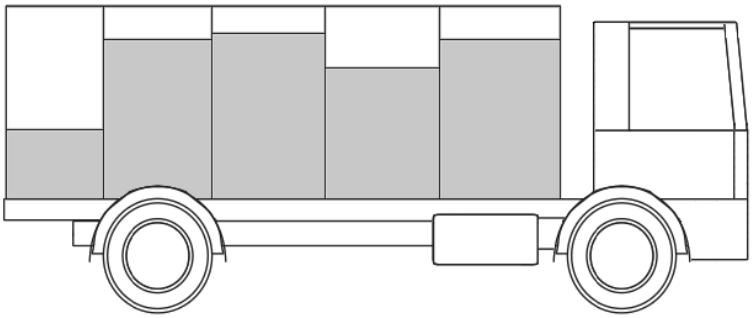
2.1.6 VRP με πολλαπλές αποθήκες (Multidepot Vehicle Routing ή MDVRP)

Στα προηγούμενα προβλήματα δρομολόγησης είχαμε μία κεντρική αποθήκη από την οποία όλα τα οχήματα προμηθεύονταν τα προϊόντα τα οποία διένειμαν και τελικά επέστρεφαν σε αυτή. Στο MDVRP χρησιμοποιούνται περισσότερες αποθήκες και η επίλυση του προβλήματος διαφοροποιείται κατά τους εξής τρόπους [21]:

- Οι αποθήκες έχουν το δικό τους πελατολόγιο και το πρόβλημα διαιρείται σε απλούστερα προβλήματα δρομολόγησης οχημάτων (VRP), προφανώς τόσα όσα και οι αποθήκες.
- Τα οχήματα μπορεί ενδιάμεσα της διαδρομής να εφοδιάζονται και τελικά να τερματίζουν σε περισσότερες από μία αποθήκες.

2.1.7 VRP με πολλαπλά τμήματα (multi-compartment vehicle routing problem ή MCVRP)

Το MCVRP μπορεί να εκφραστεί ως ένα γενικευμένο πρόβλημα VRP στη περίπτωση που τα οχήματα έχουν μόνο ένα τμήμα και κάθε πελάτης εξυπηρετείται από ένα μόνο όχημα [6]. Τα προβλήματα που αντιμετωπίζει όμως περιλαμβάνουν οχήματα τα οποία είναι όμοια μεταξύ τους αλλά ο αποθηκευτικός τους χώρος αποτελείται από διαφορετικά διαμερίσματα ενώ ακόμα υπάρχουν επιπλέον περιορισμοί για τα προϊόντα που εφοδιάζονται σε κάθε όχημα, οι οποίοι εμπίπτουν συνήθως από ασυμβατότητες μεταξύ των προϊόντων ενός τμήματος και μεταξύ των προϊόντων και των τμημάτων του οχήματος. Συγκεκριμένα, το MCVRP όπως και το απλό VRP περιλαμβάνει ένα σύνολο πελατών και μια κεντρική αποθήκη από την οποία τα οχήματα προμηθεύονται τα αγαθά που πρέπει να διανείμουν σε αυτούς ελαχιστοποιώντας το κόστος μεταφοράς. Υπάρχει όμως, όπως προαναφέρθηκε, διαφοροποίηση στους αποθηκευτικούς χώρους των οχημάτων οι οποίοι εκτός από τη συνολική χωρητικότητα που διαθέτουν, διαιρούνται σε ένα συγκεκριμένο αριθμό τμημάτων , κάθε ένα από τα οποία έχει μία συγκεκριμένη μέγιστη χωρητικότητα. Οι περιορισμοί ασυμβατότητας ορίζουν, ανάλογα με τα δεδομένα του προβλήματος, εάν επιτρέπεται η αποθήκευση διαφορετικού τύπου προϊόντων στο ίδιο τμήμα του οχήματος ή εάν έχει οριστεί εκ των προτέρων συγκεκριμένο τμήμα στο οποίο πρέπει να τοποθετηθεί το κάθε προϊόν. Τέλος αντίθετα με το VRP, κάθε πελάτης δύναται να εξυπηρετηθεί από πολλά οχήματα.



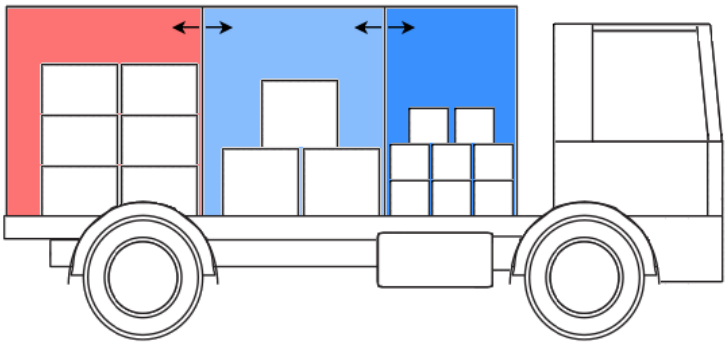
Εικόνα 2.2: Όχημα με προκαθορισμένα διαμερίσματα για διανομή πετρελαίου

Η μορφοποίηση του προβλήματος ως Mixed Integer Program προτάθηκε από τον Derigs [5].

2.1.8 VRP με πολλαπλά τμήματα μεταβλητών διαστάσεων (The multi-compartment vehicle routing problem with flexible compartment sizes ή MCVRP-FCS)

Το συγκεκριμένο πρόβλημα ανήκει στη κατηγορία των MCVRP με κάποιες διαφοροποιήσεις[12]:

- Οι διαστάσεις κάθε τμήματος δεν είναι προκαθορισμένες αφού μπορούν να αλλάξουν ξεχωριστά για κάθε όχημα ή διαδρομή.
- Τα τμήματα κάθε οχήματος είναι εκ των προτέρων χωρισμένα μεταξύ τους, για παράδειγμα με τοίχους, άρα μπορεί τα οχήματα να έχουν τμήματα διαφορετικού μεγέθους μεταξύ τους αλλά το κάθε όχημα έχει προκαθορισμένη δομή.
- Το πλήθος των τμημάτων, στα οποία διαιρείται ο αποθηκευτικός χώρος κάθε οχήματος, μπορεί να είναι ίδιο με τον αριθμό των διαφορετικών ειδών προϊόντων που υπάρχουν ή μικρότερο αφού για κάποια από τα είδη μπορεί να επιτρέπεται η ταυτόχρονη αποθήκευση τους στο ίδιο τμήμα.



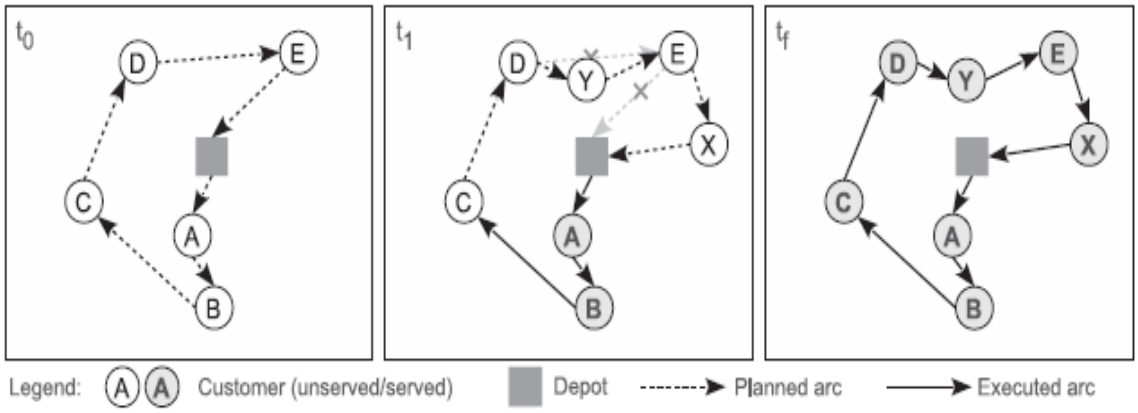
Εικόνα 2.3: Όχημα με μεταβλητά διαμερίσματα για διανομή τροφίμων

2.1.9 Προβλήματα δρομολόγησης οχημάτων με στοχαστικές και δυναμικές παραμέτρους.

Σε αντίθεση με το κλασσικό ορισμό των προβλημάτων δρομολόγησης οχημάτων, οι πραγματικές εφαρμογές συχνά περιλαμβάνουν δύο επιπλέον διαστάσεις: την εξέλιξη και τη ποιότητα των πληροφοριών. Για παράδειγμα εξέλιξη των πληροφοριών έχουμε στη περίπτωση που οι απαιτήσεις των πελατών αλλάζουν ή προστίθενται καινούριες ακόμη και όταν τα οχήματα έχουν ήδη ξεκινήσει την εξυπηρέτηση, με αποτέλεσμα ο σχεδιασμός της διαδρομής να χρειάζεται προσαρμογές. Η ποιότητα των πληροφοριών σχετίζεται με τη πιθανή αβεβαιότητα για τις τιμές των διαθέσιμων δεδομένων, όπως για παράδειγμα η ζήτηση του κάθε πελάτη από την οποία μπορεί να είναι γνωστό μόνο ένα εκτιμούμενο εύρος που τη περιλαμβάνει. Ακόμα, ανάλογα με το πρόβλημα και τη διαθέσιμη τεχνολογία, οι διαδρομές μπορεί να σχεδιάζονται στατικά (εκ των προτέρων) ή δυναμικά [15].

Στα **στατικά και καθορισμένα προβλήματα** (static and deterministic problems), όλα τα δεδομένα είναι γνωστά εκ των προτέρων και οι διαδρομές των οχημάτων δεν αλλάζουν όταν έχει ξεκινήσει η εκτέλεσή τους [15].

Στα **στατικά και στοχαστικά προβλήματα** (Static and stochastic problems) η γνώση ορισμένων δεδομένων, όπως για παράδειγμα η ζήτηση, περιορίζεται στις κατανομές των πιθανοτήτων τους και έτσι μοντελοποιούνται σαν στοχαστικές μεταβλητές [21]. Επιπλέον, οι διαδρομές σχεδιάζονται εκ των προτέρων και μόνο μικρές αλλαγές επιτρέπονται κατά την εκτέλεσή τους όπως για παράδειγμα η επιστροφή στην αποθήκη για ανεφοδιασμό. Αβεβαιότητα μπορεί να εμφανίζεται για οποιοδήποτε δεδομένο ωστόσο τα δημοφιλέστερα προβλήματα της κατηγορίας αυτής είναι: στοχαστικοί πελάτες, όπου η εξυπηρέτηση του πελάτη γίνεται με μια δεδομένη πιθανότητα, στοχαστικοί χρόνοι, στα οποία οι χρόνοι ταξιδίου ή εξυπηρέτησης μοντελοποιούνται από τυχαίες μεταβλητές [15] και τέλος στοχαστική ζήτηση όπου η ακριβής ζήτηση των πελατών γίνεται γνωστή μόνο όταν το όχημα φτάνει στη τοποθεσία του πελάτη [21].



Εικόνα 2.4: Παράδειγμα δυναμικής δρομολόγησης οχημάτων

Στα **δυναμικά και καθορισμένα προβλήματα (dynamic and deterministic problems),** όλα τα δεδομένα παρουσιάζουν αβεβαιότητα ενώ η γνωστοποίησή τους γίνεται δυναμικά κατά το σχεδιασμό ή την εκτέλεση των διαδρομών. Έτσι για το σχεδιασμό των διαδρομών απαιτείται συνεχής επικοινωνία με τους οδηγούς των οχημάτων, η οποία επιτυγχάνεται μέσα από τεχνολογική υποστήριξη (για παράδειγμα κινητά τηλέφωνα). Τέτοιου είδους προβλήματα συχνά αναφέρονται ως προβλήματα απευθείας σύνδεσης (online) ή πραγματικού χρόνου (real time) [15].

Τα **δυναμικά και στοχαστικά προβλήματα (dynamic and stochastic problems)** είναι όμοια με τα δυναμικά και καθορισμένα προβλήματα που προαναφέρθηκαν με τη διαφορά ότι εδώ παρέχονται στοχαστικές γνώσεις των δεδομένων οι οποίες αποκαλύπτονται όπως και πριν δυναμικά κατά το σχεδιασμό ή την εκτέλεση των διαδρομών [15].

2.1.10 Προγραμματισμός οχημάτων (Vehicle Scheduling Problem ή VSP)

Από τα παραπάνω έχει γίνει εμφανές ότι τα προβλήματα δρομολόγησης οχημάτων ασχολούνται κυρίως με παραμέτρους οι οποίες αφορούν τα χωρικά χαρακτηριστικά των ενεργειών που πραγματοποιούνται κατά τη διαδικασία της διανομής ή παραλαβής των προϊόντων από τα οχήματα. Τα προβλήματα προγραμματισμού οχημάτων λαμβάνουν υπόψη όσα προαναφέρθηκαν αλλά προσθέτουν κάποιους επιπλέον χρονικούς περιορισμούς, δίνοντας με αυτό το τρόπο έμφαση στους χρόνους με τους οποίους γίνονται οι εργασίες. Έτσι στον προγραμματισμό οχημάτων ο παράγοντας του χρόνου θεωρείται ότι εμπλέκεται σε κάθε δραστηριότητα και η εφικτότητα των ενεργειών που λαμβάνουν χώρα επηρεάζεται άμεσα από τη τήρηση των χρονικών περιορισμών που αυτός ορίζει. Οι βασικότεροι περιορισμοί του προβλήματος προγραμματισμού οχημάτων είναι οι ακόλουθοι [21]:

- Ο περιορισμός του συνολικού χρόνου που κάθε όχημα καταναλώνει από την αναχώρησή του από την αποθήκη μέχρι την επιστροφή του σε αυτή για ανεφοδιασμό.
- Ο περιορισμός που ορίζει συγκεκριμένου τύπου οχήματα για την εξυπηρέτηση ορισμένων τύπων προϊόντων.
- Η ύπαρξη πολλαπλών αποθηκών για τη στέγαση των οχημάτων.

Κεφάλαιο 3:Αλγόριθμοι βελτιστοποίησης προβλημάτων εφοδιαστικής αλυσίδας

3.1 Κλασσικοί ευρετικοί αλγόριθμοι (Classical heuristics)

Είναι αλγόριθμοι οι οποίοι πάντα αναζητούν μία καλύτερη λύση από τη ήδη υπάρχουσα μέχρις ότου να καταλήξουν σε μία βέλτιστη. Οι αλγόριθμοι αυτοί συνήθως χωρίζονται σε δύο κατηγορίες [13]: τους **κατασκευαστικούς** ευρετικούς αλγόριθμους (constructive heuristics) και τους αλγόριθμους **βελτίωσης** (improvement heuristics). Η αποτελεσματικότητα των ευρετικών αλγόριθμων συνήθως ελέγχεται συγκρίνοντας τα μεταξύ τους αποτελέσματά, με χρήση των παραδειγμάτων του Christofides [2] τα οποία χαρακτηρίζονται ως CMT test -bed και αποτελούνται από 14 παραδείγματα, τα οποία ανήκουν στο διάστημα από 51 κόμβους έως 200. Ακόμη χρησιμοποιούνται και παραδείγματα του Golden [11], με το χαρακτηρισμό GWKC test -bed αποτελούμενα από 20 παραδείγματα μεγέθους από 200 κόμβους έως 480.

3.1.1 Κατασκευαστικοί ευρετικοί αλγόριθμοι:

Ο πιο διαδεδομένος αλγόριθμος της κατηγορίας αυτής είναι ο **αλγόριθμος εξοικονόμησης** των Clarke και Wright [3]. Αρχικά κατασκευάζονται n εφικτές διαδρομές της μορφής (0,i,0) όπου i=1,2,....n, δηλαδή με εκκίνηση και επιστροφή στην αποθήκη. Στη συνέχεια πραγματοποιείται συγχώνευση διαδρομών που καταλήγουν στον κόμβο i με αυτές που ξεκινάνε από τον j αφαιρώντας τα τόξα (i,0) και (0,j) και εισάγοντας το τόξο (i,j) δεδομένου ότι ισχύει για τα κέρδη sij >0 όπου sij = ci0 + c0j – cij [13].

Η καλύτερη παραλλαγή του εν λόγω αλγόριθμου είναι η παράλληλη εκδοχή κατά την οποία η συγχώνευση η οποία αποδίδει το μεγαλύτερο κέρδος εφαρμόζεται σε κάθε επανάληψη έως ότου δεν εμφανίζεται κάποια άλλη επικερδής και ταυτόχρονα εφικτή συγχώνευση οπότε και ο αλγόριθμος τερματίζει [13].

Στη κατηγορία των κατασκευαστικών αλγόριθμων ανήκουν και οι ευρετικοί αλγόριθμοι πετάλων, η αρχική εκδοχή των οποίων περιγράφεται από τον **αλγόριθμο σαρώματος** των Gillett and Miller [10]. Πρόκειται για μία υποτυπώδη διαδικασία κατά την οποία δημιουργούνται διαδρομές χωρίς δυνατότητα να εμφανίζουν τομές μεταξύ τους.

Ο ευρετικός αλγόριθμος **αποσύνθεσης δύο φάσεων** των Fisher και Jaikumar [8] γνωστός ως ομαδοποίηση πρώτα-δρομολόγηση έπειτα βασίζεται στην εκτέλεση δύο βημάτων [13]:

- Τη κατασκευή ομάδων πελατών (clusters) μέσω του γενικευμένου προβλήματος ανάθεσης (GAP).
- Την επίλυση του προβλήματος πλανόδιου πωλητή (TSP) για κάθε μία ομάδα που έχει κατασκευαστεί στο πρώτο βήμα.

Αλγόριθμος απληστίας (Greedy Algorithm)

Ο αλγόριθμος απληστίας είναι ο απλούστερος τρόπος προσέγγισης της βέλτιστης λύσης για προβλήματα τα οποία επιλύονται μέσω μιας ακολουθίας βημάτων όπου σε κάθε στάδιο υπάρχει η δυνατότητα επιλογής ανάμεσα σε διαφορετικές εναλλακτικές. Με τη μέθοδο απληστίας, οι αποφάσεις λαμβάνονται σύμφωνα με τις διαθέσιμες πληροφορίες χωρίς να γίνεται περεταίρω έλεγχος άλλων λύσεων που πιθανόν να αποδίδουν καλύτερα. Για το λόγο αυτό οι αλγόριθμοι απληστίας πολλές φορές χαρακτηρίζονται ως μυωπικοί και παρά το γεγονός ότι είναι εύκολα εφαρμόσιμοι και κατανοητοί δεν ενδείκνυνται για τη λύση πολλών προβλημάτων [22].

Ένα παράδειγμα αλγόριθμου απληστίας είναι αυτό του πλησιέστερου γείτονα [21]. Σε αυτή την ευρετική διαδικασία, ο πωλητής ξεκινά από κάποια πόλη και έπειτα επισκέπτεται την πόλη που είναι πιο κοντά σε αυτή. Από εκεί επισκέπτεται την πλησιέστερη πόλη που δεν έχει ακόμη επισκεφτεί, μέχρις ότου όλες οι πόλεις να συμπεριλαμβάνονται στην διαδρομή του, και τότε επιστρέφει στην αρχική. Τα βασικά βήματα του αλγόριθμου είναι [21]:

Βήμα 1: Επιλέγεται οποιοσδήποτε κόμβος σαν ξεκίνημα της διαδρομής.

Βήμα 2: Βρίσκουμε τον πλησιέστερο κόμβο που δεν συμπεριλαμβάνεται στη διαδρομή και τον προσθέτουμε στο μονοπάτι.

Βήμα 3: Επαναλαμβάνουμε το Βήμα 2 μέχρις ότου όλοι οι κόμβοι ανήκουν στη διαδρομή.

3.1.2 Ευρετικοί αλγόριθμοι βελτίωσης (improvement heuristics)

Δύο είδη αλγόριθμων που βρίσκουν εφαρμογή στη κατηγορία αυτή είναι οι [10]: ευρετικοί εντός διαδρομών **(Intra -route heuristics)** και μεταξύ διαδρομών **(Inter -route heuristics**). Οι πρώτοι εφαρμόζουν κάποιο ευρετικό αλγόριθμο βελτίωσης διαδρομής της κατηγορίας TSP (π.χ. 2-opt ή 3-opt) σε κάθε διαδρομή ξεχωριστά, που έχει ευρεθεί από την επίλυση του VRP, και τις οποίες θέλουμε να βελτιστοποιήσουμε. Οι δεύτεροι χρησιμοποιούν τη μεταφορά κορυφών σε διαφορετικούς δρόμους της αρχικής λύσης του VRP [13].

3.2 Μεθευρετικοί αλγόριθμοι (Metaheuristics)

3.2.1 Γενικά για τους μεθευρετικούς αλγορίθμους

Οι αλγόριθμοι αυτοί χαρακτηρίζονται από διαφοροποίηση και εντατικοποίηση. Διαφοροποίηση επειδή παράγουν διαφορετικές μεταξύ τους λύσεις επιτρέποντας έτσι διεύρυνση του χώρου αναζήτησης, ενώ εντατικοποίηση επειδή εστιάζουν την αναζήτηση σε μία τοπική περιοχή γνωρίζοντας ότι η βέλτιστη λύση βρίσκεται εκεί. Κατά την επιλογή των τρεχουσών βέλτιστων λύσεων σε κάθε αναζήτηση θα πρέπει να υπάρχει σωστή ισορροπία μεταξύ των δύο αυτών χαρακτηριστικών έτσι ώστε ο αλγόριθμος να συγκλίνει γρηγορότερα στη τελική βέλτιστη λύση. Έτσι με την επιλογή κατάλληλων βέλτιστων λύσεων αυτές τελικά θα συγκλίνουν σε μία τελική βέλτιστη λύση ενώ με τη διαφοροποίηση επιτυγχάνεται αποφυγή των τοπικών ελαχίστων εξαιτίας της τυχαιότητας των λύσεων αφού λαμβάνονται υπόψη και λύσεις οι οποίες δεν αποδίδουν βελτίωση ακόμα και μη εφικτές ενδιάμεσες λύσεις [23].

Οι μεθευρετικοί αλγόριθμοι συνήθως προσεγγίζουν διαδικασίες οι οποίες κατά κάποιο τρόπο εφαρμόζονται και στη φύση. Οι εν λόγω διαδικασίες παρουσιάζονται παρακάτω [21]:

- Η εκτέλεση επαναλαμβανόμενων δοκιμών.
- Η χρήση ενός ή και παραπάνω πρακτόρων (νευρώνες, μυρμήγκια, μόρια).
- Οι πράκτορες λειτουργούν μέσω της μεταξύ τους συνεργασίας και ανταγωνισμού.
- Πραγματοποιούνται αυτό-τροποποιήσεις των ευρετικών παραμέτρων του προβλήματος και σε κάποιες περιπτώσεις αναπαράστασή του.

Τα χαρακτηριστικά που διέπουν τους μεθευρετικούς αλγόριθμους είναι τα εξής [21]:

- Μοντελοποιούν φαινόμενα τα οποία παρουσιάζονται και στη φύση.
- Μπορούν να μεταφερθούν εύκολα σε παράλληλη μορφή.
- Είναι προσαρμοστικοί αλγόριθμοι.

3.2.2 Άπληστη τυχαιοποιημένη προσαρμοστική αναζήτηση (Greedy randomized adaptive search procedure ή GRASP)

Η GRASP είναι μία επαναληπτική διαδικασία για την εύρεση μιας βέλτιστης λύσης η οποία αποτελείται από δύο στάδια: το στάδιο της κατασκευής και το στάδιο της τοπικής αναζήτησης. Η διαδικασία επαναλαμβάνεται μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού όπως ο μέγιστος αριθμός επαναλήψεων ή η εύρεση μιας επιθυμητής λύσης [7].

Στο στάδιο της κατασκευής, μία εφικτή λύση δημιουργείται προσθέτοντας επαναληπτικά ένα στοιχείο τη φορά. Δεδομένου ότι πρόκειται για έναν αλγόριθμο **απληστίας**, σε κάθε επανάληψη, η επιλογή του εν λόγο στοιχείου πραγματοποιείται κατατάσσοντας όλα τα στοιχεία σε μία λίστα υποψηφιότητας και στη συνέχεια μετράται (μυωπικά) το όφελος που αποδίδει η επιλογή κάθε στοιχείου. Ο αλγόριθμος είναι **προσαρμοστικός** αφού τα οφέλη όλων των στοιχείων ενημερώνονται σε κάθε επανάληψη της κατασκευαστικής φάσης έτσι ώστε να γίνονται αντιληπτές οι αλλαγές τις οποίες επέφερε η επιλογή του προηγούμενου στοιχείου. Η **τυχαιότητα** της διαδικασίας έγκειται στο γεγονός ότι η επιλογή κάθε στοιχείου της λίστας γίνεται τυχαία ανάμεσα σε αυτά που παρουσιάζουν τις υψηλότερες αποδόσεις χωρίς απαραίτητα να επιλέγεται αυτό με την υψηλότερη. Η λίστα με τους καλύτερους υποψήφιους χαρακτηρίζεται ως περιορισμένη λίστα υποψηφίων (restricted candidate list ή RCL)[7]. Παρακάτω παρουσιάζεται ένας ψευδοκώδικας για τη κατασκευαστική φάση του αλγόριθμου.

Διαδικασία Κατασκευή άπληστης τυχαιοποιημένης λύσης (Λύση)

1 Λύση={}

2 **Για** μη ολοκληρωμένη κατασκευή λύσης

3 **Φτιάξε RCL (RCL)**

4 **s=επέλεξε τυχαίο στοιχείο(RCL);**

5 **Λύση= Λύση υ{s};**

6 **Εφαρμογή συνάρτησης απληστίας (s);**

7 **Τέλος για**

8 Τέλος Κατασκευή άπληστης τυχαιοποιημένης λύσης

Η λύση που πρόκειται να κατασκευαστεί, αρχικοποιείται στη πρώτη γραμμή του ψευδοκώδικα. Το loop από τη δεύτερη ως την έβδομη γραμμή επαναλαμβάνεται μέχρι να κατασκευαστεί η λύση. Στη τρίτη γραμμή κατασκευάζεται η περιορισμένη λίστα υποψηφίων.

Στη τέταρτη γραμμή επιλέγεται τυχαία ένας υποψήφιος από τη λίστα ενώ στη πέμπτη γραμμή προστίθεται στη λύση. Η επίδραση του επιλεγμένου στοιχείου s στα οφέλη που σχετίζονται με κάθε στοιχείο, λαμβάνεται υπόψη στην έκτη γραμμή όπου εφαρμόζεται η συνάρτηση απληστίας [7].

Οι λύσεις που δημιουργούνται κατά τη κατασκευαστική φάση της GRASP δεν είναι πάντα βέλτιστες, έτσι όπως προαναφέρθηκε ακολουθεί μία δεύτερη φάση τοπικής αναζήτησης με στόχο τη βελτίωση των λύσεων που έχουν παραχθεί στη πρώτη φάση της διαδικασίας.

3.2.3 Αλγόριθμοι τοπικής αναζήτησης

Οι αλγόριθμοι τοπικής αναζήτησης (local search algorithms) ξεκινάνε από μία αρχική τυχαία λύση και συνεχίζουν τη διαδικασία αναζήτησης μεταβάλλοντας την διαδοχικά. Μία μετακίνηση εφαρμόζεται στην ήδη υπάρχουσα λύση για να παραχθεί μία καινούρια από την γειτονιά της. Η απόφαση για την πραγματοποίηση της εν λόγο κίνησης εξαρτάται από την ποιότητα των παραγόμενων λύσεων. Σε περίπτωση θετικής απόφασης τότε η υπάρχουσα λύση θα αντικατασταθεί από κάποια γειτονική η οποία θα χρησιμοποιηθεί ως αφετηρία για τις επακόλουθες δοκιμές, διαφορετικά η αναζήτηση συνεχίζεται με την υπάρχουσα λύση. Στους κλασικούς αλγορίθμους τοπικής αναζήτησης η απόφαση είναι θετική όταν η νέα λύση είναι καλύτερη από την υπάρχουσα. Η όλη διαδικασία συνεχίζεται μέχρι την ικανοποίηση κάποιου κριτηρίου τερματισμού το οποίο συνήθως αφορά ένα συνολικό αριθμό επαναλήψεων ή έναν αριθμό επαναλήψεων που δεν εμφανίζουν βελτίωση στη λύση [13].

Πλανόδιος πωλητής - μέθοδος k-opt
Στο πρόβλημα του πλανόδιου πωλητή ένας δημοφιλής αλγόριθμος αναζήτησης είναι ο k-opt (για k=2 ή k=3) στον οποίο κάνοντας k-opt κινήσεις, μπορούν να επιτευχθούν γειτονικές λύσεις με διαγραφή k ακμών από τη τρέχουσα διαδρομή και δημιουργία μιας βελτιωμένης διαδρομής επανασυνδέοντας τα μονοπάτια που προκύπτουν με k νέες ακμές [19].

Η διαδικασία της μεθόδου 2-opt είναι η εξής[21]:

Βήμα 1 Έστω T η τρέχουσα διαδρομή

Βήμα 2 Για κάθε κόμβο i του τρέχον γραφήματος εξετάζουμε όλες τις 2-opt κινήσεις που μπορούν να πραγματοποιηθούν από τον i και i+1 μέσα στη T. Εάν με αυτό το τρόπο παρατηρηθεί βελτίωση στο κόστος της διαδρομής, τότε επιλέγεται η 2-opt κίνηση που αποδίδει καλύτερα και η διαδρομή προσαρμόζεται κατάλληλα.

Βήμα 3 Εάν δεν εντοπίζεται περεταίρω βελτίωση, σταματάμε.

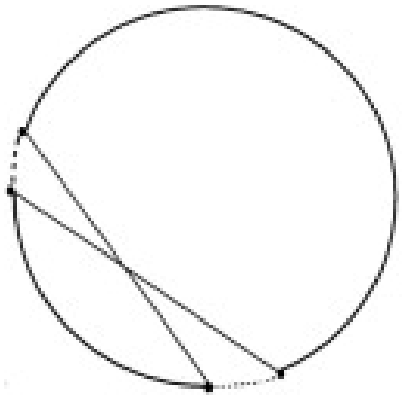
Αντίστοιχα για το 3-opt η διαδικασία είναι η ακόλουθη [21]:

Βήμα 1 Έστω T η τρέχουσα διαδρομή

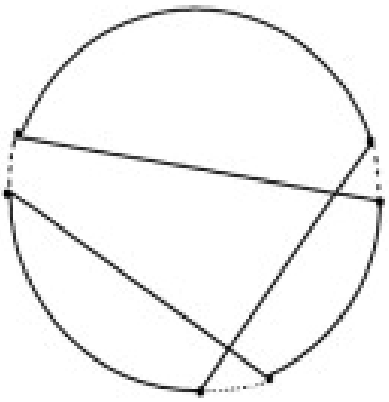
Βήμα 2 Για κάθε κόμβο i του τρέχον γραφήματος υπολογίζουμε ένα σύνολο από κόμβους N(i).

Βήμα 3 Για κάθε κόμβο i εξετάζουμε όλες τις 3-opt κινήσεις τέτοιες ώστε να διαγράφουν από τρεις πλευρές και ταυτόχρονα κάθε μία από αυτές να έχει μια πλευρά στο N(i) . Εάν με αυτό το τρόπο παρατηρηθεί βελτίωση στο κόστος της διαδρομής, τότε επιλέγεται η 3-opt κίνηση που αποδίδει καλύτερα και η διαδρομή προσαρμόζεται κατάλληλα.

Βήμα 4 Εάν δεν εντοπίζεται περεταίρω βελτίωση, σταματάμε.



Εικόνα 3.1: Παράδειγμα 2-opt



Εικόνα3.2: Παράδειγμα 3-opt

Πλανόδιος πωλητής-Εισχώρηση κόμβων και ακμών

Σε αυτή τη περίπτωση, ένας κόμβος αφαιρείται από τη θέση του και επανατοποθετείται σε άλλο σημείο τέτοιο ώστε το κόστος διαδρομής να μειώνεται. Η διαδικασία μπορεί να περιγραφτεί από τα εξής βήματα [21]:

Βήμα 1 Έστω T η τρέχουσα διαδρομή

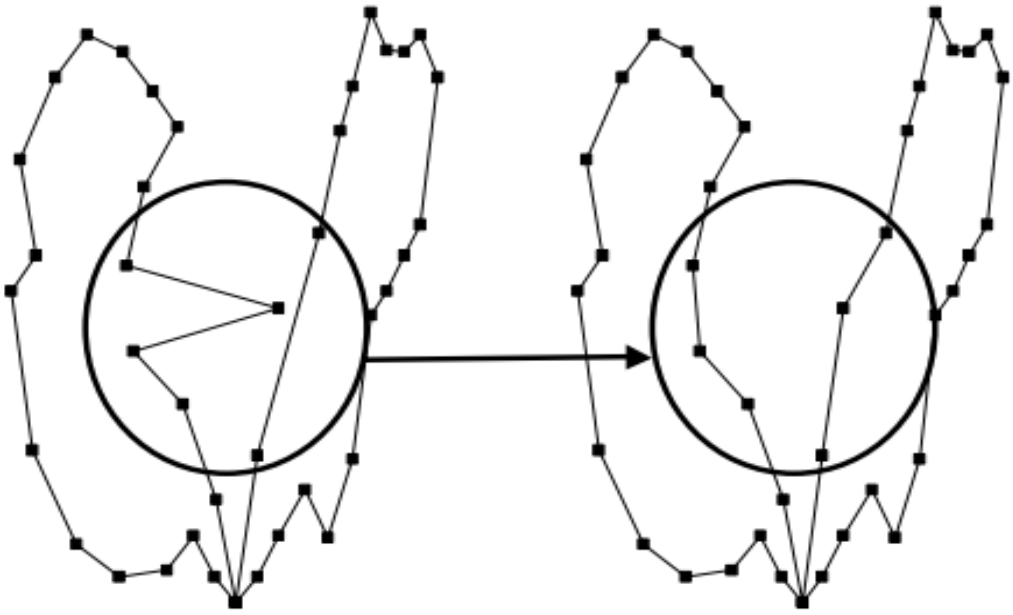
Βήμα 2 Για κάθε κόμβο i του τρέχον γραφήματος εξετάζουμε όλες τις πιθανές θέσεις που θα μπορούσε να εισέλθει ο κόμβος i μέσα στη T. Επιλέγουμε να τον τοποθετήσουμε στη θέση εκείνη που μειώνει το κόστος περισσότερο και η διαδρομή προσαρμόζεται κατάλληλα.

Βήμα 3 Εάν δεν εντοπίζεται περεταίρω βελτίωση, σταματάμε.

Με παρόμοιο τρόπο μπορεί να εφαρμοστεί εισχώρηση ακμών αντί κόμβων με τη διαφορά ότι στο δεύτερο βήμα της παραπάνω διαδικασίας για κάθε κόμβο i εξετάζουμε όλες τις θέσεις που θα μπορούσαν να εισέλθουν οι i και i+1 ακμές και επιλέγονται αυτές που ελαχιστοποιούν το κόστος.

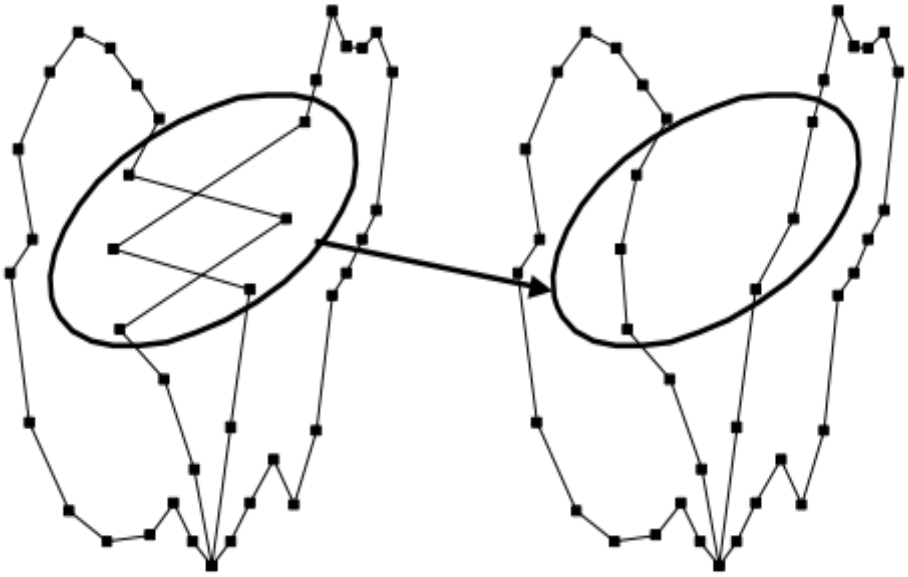
Ο Waters [20], πρότεινε τέσσερεις διαφορετικές διαδικασίες τοπικής αναζήτησης που μπορούν να εφαρμοστούν μεταξύ δύο ή περισσότερων διαδρομών όπως παρουσιάζονται παρακάτω [21]:

- Την 1-0 επανατοποθέτηση (1-0 relocate) όπου πραγματοποιείται διαγραφή ενός πελάτη από μία διαδρομή επανατοποθέτηση του σε μία άλλη με καλύτερο κόστος.



Εικόνα 3.3: 1-0 επανατοποθέτηση

- Την 1-1 ανταλλαγή (1-1 exchange) στην οποία γίνεται ταυτόχρονη ανταλλαγή δύο πελατών που βρίσκονται σε ίδιες ή διαφορετικές διαδρομές.



Εικόνα 3.4: 1-1 ανταλλαγή πελατών διαφορετικών διαδρομών

- Την 1-2 ανταλλαγή (1-2 exchange) όπου ένας πελάτης που βρίσκεται σε κάποιον κύκλο ανταλλάσσεται με δύο πελάτες που βρίσκονται σε διαφορετικό κύκλο.
- Την 2-2 ανταλλαγή (2-2 exchange) η οποία πρόκειται για μια συνδυασμένη ανταλλαγή δύο γειτονικών πελατών ενός κύκλου με δύο γειτονικούς πελάτες διαφορετικού κύκλου.

3.2.4 Έρευνα πληθυσμού (Population search)

Οι **γενετικοί αλγόριθμοι(genetic algorithms)**, αναφέρονται στη μετατροπή ενός πληθυσμού λύσεων μέσω μιας διαδικασίας διασταύρωσης. Αρχικά γίνεται επιλογή δύο <<γονέων>> από τον πληθυσμό οι οποίοι συνδυάζονται ώστε να παραχθούν μία ή δύο νέες λύσεις (γόνοι). Στη συνέχεια οι γόνοι αυτοί μετατρέπονται (συνήθως τυχαία) και αντικαθιστούν τα χειρότερα στοιχεία του πληθυσμού. Για παράδειγμα, ο Prins [16] μετατρέπει τη λύση ενός προβλήματος VRP σε TSP αφαιρώντας τα όρια διαδρομών και ανακατασκευάζει τη λύση του VRP στο τέλος της διαδικασίας. Ακόμη στη κατηγορία αυτή ανήκουν και οι **μιμητικοί αλγόριθμοι (memetic algorithms)** στη περίπτωση της μετατροπής των γόνων μέσω συγκεκριμένων ευρετικών αλγόριθμων βελτίωσης. Άλλη μια γνωστή διαδικασία είναι αυτή της **προσαρμοστικής μνήμης (adaptive memory procedure)** η οποία έχει προταθεί από τους Rochat και Taillard [17]. Πρώτα εκτελείται ένας αλγόριθμος περιορισμένης αναζήτησης (tabu search algorithm) και στη συνέχεια γίνεται προσπάθεια για μεταβελτιστωποίηση των καλύτερων λύσεων που έχουν

παραχθεί από τη πρώτη φάση μέσω μηχανισμών διασταύρωσής τους και τοπικών αναζητήσεων [13].

3.2.5 Μηχανισμοί εκμάθησης (Learning mechanisms)

Οι μηχανισμοί εκμάθησης περιλαμβάνουν **νευρωνικά δίκτυα** (neural networks) και **βελτιστοποίηση αποικίας μυρμηγκιών** (ant colony optimization). Οι προσπάθειες που έγιναν για εφαρμογή των νευρωτικών δικτύων στο VRP ήταν ανεπιτυχείς και η έρευνα στο τομέα αυτό τελικά εγκαταλείφτηκε. Οι ευρετικοί αλγόριθμοι βελτιστοποίησης αποικίας μυρμηγκιών προσπαθούν να μιμηθούν τη συμπεριφορά των μυρμηγκιών τα οποία ανιχνεύουν τα μονοπάτια που περιέχουν φερομόνη και τα ενισχύουν με τη δική τους φερομόνη. Με αυτό το τρόπο επιτυγχάνεται ο εντοπισμός των συντομότερων διαδρομών στα οποία η φερομόνη συσσωρεύεται γρηγορότερα. Στους μεθευρετικούς αλγόριθμους, η φερομόνη αντιπροσωπεύει τη μνήμη του συστήματος και ανταποκρίνεται στις ακμές που εμφανίζονται συχνότερα στις συμφέρουσες λύσεις. Έτσι ο αλγόριθμος απομνημονεύει τις καλές ακμές και είναι πιο πιθανό να τις συμπεριλάβει στη τελική λύση [13].

Κεφάλαιο 4: Περιγραφή και επίλυση προβλημάτων

4.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο περιγράφεται ο τρόπος με τον οποίο λειτουργούν και στη συνέχεια υλοποιούνται οι μέθοδοι GRASP και πλησιέστερος γείτονας μέσα από το προγραμματιστικό περιβάλλον MATLAB. Ακόμα περιγράφεται και η μέθοδος τοπικής αναζήτησης 1-1 exchange που χρησιμοποιήθηκε για να ελεγχθεί εάν οι λύσεις που κατασκευάστηκαν με τους πρώτους αλγόριθμους μπορούν να βελτιωθούν περαιτέρω.

Αρχικά θα ξεκινήσουμε παραθέτοντας τα δεδομένα που είχαμε στη διάθεσή μας για τα προβλήματα που χρησιμοποιήθηκαν.

4.2 Δεδομένα Προβλήματος

Ως δεδομένα για την επίλυση του προβλήματος μας δόθηκαν σε μορφή αρχείου excel οι ακόλουθες πληροφορίες:

Για τους πελάτες:

- Το πλήθος των πελατών που πρέπει να εξυπηρετηθούν. Πρόκειται για το συνολικό αριθμό των κόμβων-πελατών του προβλήματος, χωρίς φυσικά να συμπεριλαμβάνεται σε αυτόν η αποθήκη, ο οποίος αποθηκεύεται στη μεταβλητή ''num''.
- Οι συντεταγμένες του κάθε πελάτη. Δηλαδή η θέση των πελατών στον χάρτη η οποία δίνεται σε δύο διαστάσεις x και y και εκχωρείται σε δύο μεταβλητές ''X'' και ''Y'' αντίστοιχα.
- Η απαίτηση σε μονάδες προϊόντων του κάθε πελάτη. Κάθε πελάτης θεωρείται ότι εξυπηρετείται με τη παράδοση σε αυτόν ενός συγκεκριμένου αριθμού μονάδων προϊόντων ο οποίος θεωρείται γνωστός εξαρχής και αποθηκεύεται στη μεταβλητή ''zhthsh''.

Πρέπει να σημειωθεί ότι οι κόμβοι είναι αριθμημένοι έτσι ώστε ο κόμβος 1 να αντιστοιχεί στη αποθήκη (προφανώς η ζήτηση θα είναι μηδέν για τον συγκεκριμένο).

Για το όχημα:

- Η χωρητικότητα του οχήματος σε μονάδες προϊόντων. Δηλαδή η ποσότητα προϊόντων που το όχημα δύναται να μεταφέρει χωρίς να χρειαστεί να επιστρέψει στην αποθήκη για ανεφοδιασμό και η οποία αποθηκεύεται στη μεταβλητή ''capacity''.
- Ο μέγιστος χρόνος διαδρομής. Είναι ο μέγιστος χρόνος κατά τον οποίο ένα όχημα επιτρέπεται να εκτελεί μια διαδρομή πριν επιστρέψει στην αποθήκη και ξεκινήσει την επόμενη. Ο χρόνος αυτός αποθηκεύεται στη μεταβλητή ''maxtime''.

Ακόμη έχουμε ως δεδομένο και το χρόνο που το όχημα χρειάζεται για τη παράδοση των προϊόντων σε κάθε πελάτη ο όποιος εκχωρείται στη μεταβλητή ''time''.

Όπως προαναφέρθηκε, οι συντεταγμένες των κόμβων X και Y είναι γνωστές. Για την επίλυση του προβλήματος όμως χρειάζεται να ξέρουμε τις μεταξύ τους αποστάσεις. Για το λόγο αυτό κατασκευάζουμε ένα πίνακα C διαστάσεων (num+1) x (num+1), τα στοιχεία του οποίου είναι η ευκλείδεια απόσταση ανάμεσα σε όλους τους κόμβους (αποθήκη και πελάτες) και υπολογίζονται ως εξής:

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

Ο πίνακας που προκύπτει είναι συμμετρικός, αφού η απόσταση από έναν κόμβο i σε έναν άλλο κόμβο j προφανώς δε θα μεταβάλλεται από τον j στον i. Επιπλέον η διαγώνιος του πίνακα θα αποτελείται από μηδενικά στοιχεία αφού για τις συγκεκριμένες θέσεις ισχύει i=j.

Τελικά τα στοιχεία του πίνακα C, θεωρούμε ότι αντιστοιχούν στο κόστος μετακίνησης από έναν κόμβο σε κάποιον άλλον το οποίο εκφράζεται σε χρονικές μονάδες, εξισώνοντας την απόσταση μεταξύ των κόμβων με το χρόνο που χρειάζεται κάποιο όχημα για να τη διασχίσει.

4.3 Αλγόριθμος του πλησιέστερου γείτονα

Στον αλγόριθμο αυτό ένα όχημα ξεκινάει από την αποθήκη και επισκέπτεται τον πελάτη που είναι πλησιέστερος σε αυτήν. Στη συνέχεια πηγαίνει στον πλησιέστερο πελάτη από τη θέση που βρίσκεται ο οποίος δεν έχει εξυπηρετηθεί ακόμα. Η διαδικασία επαναλαμβάνεται μέχρι το όχημα να επισκεφτεί όλους τους πελάτες και τελικά να επιστρέψει στην αποθήκη από όπου ξεκίνησε. Ο αλγόριθμός ελέγχει σε κάθε βήμα αν το όχημα μπορεί να εξυπηρετήσει έναν νέο πελάτη χωρίς να παραβιάσει τους περιορισμούς χρόνου και χωρητικότητας. Αν οι περιορισμοί παραβιάζονται τότε το όχημα επιστρέφει στην αποθήκη και ένα νέο όχημα ξεκινά.

4.3.1 Υλοποίηση της μεθόδου του πλησιέστερου γείτονα

Για την υλοποίηση της μεθόδου, αρχικά γίνεται έλεγχος της απόστασης από i=1 προς όλα τα j=2,3...num+1, δηλαδή όλους τους πελάτες του προβλήματος, και τελικά επιλέγεται η ελάχιστη. Έπειτα γίνεται έλεγχος των περιορισμών μέγιστης χωρητικότητας και χρόνου, εάν υπάρχει συμφωνία τότε θεωρούμε εφικτή την εξυπηρέτηση του συγκεκριμένου πελάτη στη τρέχουσα διαδρομή και η μεταβλητή ''pelates'' η οποία εκφράζει το σύνολο των πελατών του προβλήματος που δεν έχουν εξυπηρετηθεί μειώνεται κατά ένα. Ακόμα ο εν λόγω κόμβος προστίθεται στο διάνυσμα want το οποίο αναπαριστά την επιθυμητή διαδρομή, η στήλη του πίνακα C στην οποία ανήκει απειρίζεται έτσι ώστε να μην επιλεχθεί ξανά σε επόμενη επανάληψη της πρώτης φάσης του προβλήματος και τελικά το i παίρνει τη τιμή του

συγκεκριμένου πελάτη προκειμένου η επόμενη επανάληψη της διαδικασίας αναζήτησης πλησιέστερου γείτονα να ξεκινάει από τη θέση που βρίσκεται. Επιπλέον, το κόστος μετακίνησης από τη προηγούμενη θέση του οχήματος μέχρι τη θέση που βρίσκεται ο πελάτης προστίθεται στη μεταβλητή ''nearestcost'' η οποία εκφράζει το συνολικό κόστος μετακίνησης του προβλήματος. Σε περίπτωση μη ικανοποίησης των περιορισμών, το κόστος μετακίνησης από τον προηγούμενο πελάτη έως την αποθήκη προστίθεται στο συνολικό κόστος διαδρομής ''nearestcost'', το όχημα επιστρέφει στην αποθήκη (το i παίρνει τη τιμή 1) και από εκεί επαναλαμβάνεται η διαδικασία από την αρχή. Ο αλγόριθμος τερματίζει όταν η μεταβλητή ''pelates'' γίνει ίση με το μηδέν.

Όπως προαναφέρθηκε, η εξυπηρέτηση του πελάτη που έχει ευρεθεί στη πρώτη φάση της μεθόδου ως ο πλησιέστερος από τη θέση του οχήματος, θεωρείται εφικτή στη τρέχουσα διαδρομή μόνο όταν πληρούνται οι περιορισμοί μέγιστου χρόνου και χωρητικότητας.

Συγκεκριμένα για το χρονικό περιορισμό, κατασκευάζεται ένας πίνακας t ως εξής:

t(d)=t(d-1)+C(want1,want2)+time

όπου:

want1: η τρέχουσα θέση του οχήματος.

want2: η θέση που έχει επιλεχθεί για να μεταβεί το όχημα.

Έτσι θέτοντας t(1)=0, στο t(2) θα αποθηκεύεται ο χρόνος που δαπανά το όχημα για να μεταφερθεί από την αποθήκη στον πρώτο πελάτη που έχει επιλεχθεί για εξυπηρέτηση συν το χρόνο εξυπηρέτησης , στο t(3) η τιμή του t(2) συν το χρόνο από τον πρώτο πελάτη στο δεύτερο συν το χρόνο εξυπηρέτησης κ.ο.κ.

Στη συνέχεια πραγματοποιείται έλεγχος του περιορισμού που ορίζει ότι η τιμή του t(d) συν το χρόνο που απαιτείται για να επιστρέψει το όχημα στην αποθήκη δε πρέπει να ξεπερνά το μέγιστο χρόνο διαδρομής: t(d)+C(want2,1)<=maxtime.

Πρέπει να σημειωθεί ότι κάθε φορά που ο περιορισμός δεν ικανοποιείται οπότε και το όχημα επιστρέφει στην αποθήκη το t(d) γίνεται μηδέν.

Ακόμη πρέπει να γίνεται έλεγχος έτσι ώστε αθροιστικά ο αριθμός των προϊόντων που πρέπει να διανείμει το όχημα στους πελάτες που έχουν επιλεχθεί στην ίδια διαδρομή να μη ξεπερνάει τη χωρητικότητά του.

Προκειμένου να επιτευχθεί αυτό, κατασκευάζεται ένας πίνακας w ως εξής:

w(d)=w(d-1)+zhthsh(want2)

Έτσι για w(1)=0, το w(2) θα εκφράζει τη ποσότητα προϊόντων που απαιτεί να παραλάβει ο πρώτος πελάτης, το w(3) τη ποσότητα προϊόντων που απαιτούν ο πρώτος και ο δεύτερος πελάτης μαζί κ.ο.κ.

Έπειτα, ακολουθεί έλεγχος της συνθήκης: capacity>=w(d), εάν αυτή ικανοποιείται τότε ο πελάτης που έχει επιλεχθεί στη τρέχουσα επανάληψη εξυπηρετείται άμεσα από το όχημα αλλιώς το όχημα επιστρέφει στην αποθήκη για ανεφοδιασμό, το t(d) γίνεται ίσο με το μηδέν και η εξυπηρέτηση συνεχίζεται για τον πελάτη που βρίσκεται πλησιέστερα σε αυτήν και δεν έχει δεχθεί επίσκεψη από το όχημα.

Ακολουθεί ένα παράδειγμα εφαρμογής της μεθόδου με παραμέτρους:

αριθμός πελατών (num): 5

χωρητικότητα οχήματος (capacity): 100

μέγιστος χρόνος διαδρομής (maxtime): 75

πίνακας αποστάσεων-κόστους (C):

	1	2	3	4	5	6
1	0	13,8924	21,0238	32,5576	17,2047	14,1421
2	13,8924	0	12,3693	19,2094	31,0644	22,2036
3	21,0238	12,3693	0	15,2971	37,0135	21,0238
4	32,5576	19,2094	15,2971	0	49,679	36,0555
5	17,2047	31,0644	37,0135	49,679	0	20,3961
6	14,1421	22,2036	21,0238	36,0555	20,3961	0

ζήτηση πελατών (zhthsh):

1	0
2	7
3	30
4	16
5	9
6	21

χρόνος παράδοσης (time): 5

1η επανάληψη:

Το όχημα ξεκινάει από την αποθήκη (1) και μεταβαίνει στο πλησιέστερο σε αυτή πελάτη που όπως φαίνεται από το πίνακα αποστάσεων είναι ο (2).

Το διάνυσμα διαδρομής γίνεται: want =1 2

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(2)=zhthsh(2)=7

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(2)= C(1,2)+time=18,8924

το συνολικό κόστος διαδρομών είναι: nearestcost= C(1,2)=13,8924

2η επανάληψη:

Το όχημα πηγαίνει στο πελάτη (3) αφού είναι ο πλησιέστερος στο πελάτη (2) που βρισκόταν το όχημα πριν.

Το διάνυσμα διαδρομής γίνεται: want =1 2 3

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(3)=w(2)+zhthsh(3)=37

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(3)= t(2)+C(2,3)+time=36,2618

Το συνολικό κόστος διαδρομών είναι: nearestcost= C(1,2)+C(2,3)=26,2618

3η επανάληψη:

Από το πελάτη (3), ο πλησιέστερος πελάτης που δεν έχει δεχθεί ακόμα επίσκεψη είναι ο (4), όμως εάν από τον (3) το όχημα μεταβεί στον (4) και έπειτα επιστρέψει στην αποθήκη, ο συνολικός χρόνος διαδρομής θα υπερβεί το μέγιστο επιτρεπτό χρόνο που έχει τεθεί στους περιορισμούς. Έτσι επιλέγεται το όχημα από τον πελάτη (3) να επιστρέψει στην αποθήκη (1).

Το διάνυσμα διαδρομής γίνεται: want =1 2 3 1

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(4)=0

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(4)= 0

Το συνολικό κόστος διαδρομών είναι: nearestcost= C(1,2)+C(2,3)+C(3,1)=47,2856

4η επανάληψη:

Από την αποθήκη (1), ο πλησιέστερος σε αυτήν πελάτης που δεν έχει εξυπηρετηθεί ακόμα είναι ο (6) στον οποίο και μεταβαίνει το όχημα.

Το διάνυσμα διαδρομής γίνεται: want =1 2 3 1 6

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(5)=w(4)+zhthsh(6)=21

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(5)= t(4)+C(1,6)+time=19,1421

Το συνολικό κόστος διαδρομών είναι:

nearestcost= C(1,2)+C(2,3)+C(3,1)+C(1,6)=61,4277

5η επανάληψη:

Οι πελάτες που δεν έχουν εξυπηρετηθεί ακόμα είναι ο (4) και ο (5). Το όχημα θα μεταβεί στο πελάτη (5) αφού είναι πλησιέστερος στον (6) που βρισκόταν πριν.

Το διάνυσμα διαδρομής γίνεται: want =1 2 3 1 6 5

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(6)=w(5)+zhthsh(5)=30

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(6)= t(5)+C(6,5)+time=44,5382

Το συνολικό κόστος διαδρομών είναι:

nearestcost= C(1,2)+C(2,3)+C(3,1)+C(1,6)+C(6,5)=81,8238

6η επανάληψη:

Ο μόνος πελάτης που δεν έχει εξυπηρετηθεί ακόμα είναι ο (4), όμως εάν από τον πελάτη (5) το όχημα μεταβεί στον (4) και έπειτα επιστρέψει στην αποθήκη, ο συνολικός χρόνος διαδρομής θα υπερβεί το μέγιστο επιτρεπτό χρόνο που έχει τεθεί στους περιορισμούς. Γι' αυτό το λόγο το όχημα επιστρέφει στην αποθήκη (1).

Το διάνυσμα διαδρομής γίνεται: want =1 2 3 1 6 5 1

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(7)=0

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(7)= 0

Το συνολικό κόστος διαδρομών είναι:

nearestcost= C(1,2)+C(2,3)+C(3,1)+C(1,6)+C(6,5)+C(5,1)=99,0285

$7^{η}$ επανάληψη

Το όχημα μπορεί πλέον να μεταβεί στον πελάτη (4) που είναι και ο μόνος που δεν έχει εξυπηρετηθεί.

Το διάνυσμα διαδρομής γίνεται: want =1 2 3 1 6 5 1 4

Η συνολική ζήτηση τρέχουσας διαδρομής είναι: w(8)=w(7)+zhthsh(4)=16

Ο συνολικός χρόνος τρέχουσας διαδρομής είναι: t(8)= t(7)+C(1,4)+time=37,5576

Το συνολικό κόστος διαδρομών είναι:

nearestcost= C(1,2)+C(2,3)+C(3,1)+C(1,6)+C(6,5)+C(5,1)+C(1,4)=131,5861

Τέλος αφού έχουν εξυπηρετηθεί όλοι οι πελάτες του προβλήματος, το όχημα επιστρέφει πίσω στην αποθήκη με το τελικό διάνυσμα διαδρομής να έχει τη μορφή:

want =1 2 3 1 6 5 1 4 1

ενώ το αντίστοιχο συνολικό κόστος διαδρομών θα είναι:

nearestcost= C(1,2)+C(2,3)+C(3,1)+C(1,6)+C(6,5)+C(5,1)+C(1,4)+C(4,1)=164,1437

4.4 Αλγόριθμος Άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης

Η συγκεκριμένη μέθοδος, επαναλαμβάνει τη παραπάνω διαδικασία με τη διαφορά ότι στη φάση της επιλογής του επόμενου πελάτη για εξυπηρέτηση, δεν επιλέγεται απαραίτητα εκείνος που βρίσκεται πλησιέστερα στο όχημα. Συγκεκριμένα η επιλογή γίνεται τυχαία ανάμεσα σε ένα αριθμό υποψήφιων πελατών οι οποίοι απέχουν λιγότερο από τη θέση του οχήματος σε σχέση με τους υπόλοιπους. Όσο μεγαλώνει ο αριθμός των υποψήφιων πελατών, τόσο αυξάνεται και η τυχαιότητα με την οποία επιλέγονται οι πελάτες ενώ εάν επαναλάβουμε πολλές φορές τη διαδικασία, οι διαδρομές που θα παράγονται κάθε φορά θα παρουσιάζουν μεγαλύτερη ποικιλομορφία σε σχέση με αυτές που θα είχαμε για μικρό αριθμό υποψηφίων, οι

οποίες όπως είναι λογικό θα παρουσιάζουν περισσότερες ομοιότητες μεταξύ τους. Μετά την επιλογή του πελάτη, ακολουθεί όπως και πριν έλεγχος των περιορισμών του προβλήματος, εάν υπάρχει συμφωνία τότε το όχημα μεταβιβάζεται στον πελάτη που έχει επιλεχθεί αλλιώς επιστρέφει στην αποθήκη. Σε κάθε περίπτωση, ακολουθεί επανάληψη της διαδικασίας εύρεσης επόμενου πελάτη για εξυπηρέτηση.

4.4.1 Υλοποίηση του αλγόριθμου Άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης

Όπως προαναφέρθηκε, η φάση στην οποία ελέγχονται οι περιορισμοί και λαμβάνεται η απόφαση εάν το όχημα μπορεί να συνεχίσει τη μετακίνηση του στη τρέχουσα διαδρομή ή πρέπει να επιστρέψει στην αποθήκη και να ξεκινήσει καινούρια, είναι όμοιο με αυτό που αναλύσαμε για τη μέθοδο του πλησιέστερου γείτονα. Έτσι μένει να αναλυθεί ο τρόπος που επιτυγχάνεται η τυχαιοποιημένη επιλογή πελατών μέσα από τη λίστα υποψηφιότητας κατά τη πρώτη φάση της μεθόδου.

Αρχικά, επιλέγεται ο αριθμός που επιθυμούμε να έχει η λίστα υποψηφίων και αποθηκεύεται στη μεταβλητή ''arithmos''. Για παράδειγμα έστω ότι θέλουμε η επιλογή του πελάτη που θα προταθεί για εξυπηρέτηση να γίνεται κάθε φορά τυχαία ανάμεσα στους πέντε πλησιέστερους από τη θέση του οχήματος. Ο αριθμός αυτός επιλέγεται πριν αρχίσει οποιαδήποτε επαναληπτική διαδικασία, γι' αυτό και δεν μπορεί να αλλάξει εφόσον έχει ξεκινήσει το όχημα την εξυπηρέτηση. Στη συνέχεια, πριν από κάθε διαδικασία εύρεσης επόμενου πελάτη για εξυπηρέτηση, γίνεται έλεγχος εάν οι πελάτες που δεν έχουν εξυπηρετηθεί έως τώρα, είναι περισσότεροι ή ίσοι σε αριθμό με τη μεταβλητή ''arithmos'' που ορίσαμε προηγουμένως. Εάν ο έλεγχος είναι αληθής, τότε ο πίνακας του κόστους αποθηκεύεται προσωρινά σε ένα νέο πίνακα Β. Έπειτα ακολουθεί κανονικά η διαδικασία εύρεσης πλησιέστερου πελάτη από τη θέση του οχήματος όπως στον αλγόριθμο του πλησιέστερου γείτονα. Όταν τελειώσει η διαδικασία, ο πλησιέστερος από όλους τους πελάτες του προβλήματος έχει αποθηκευτεί στο πρώτο κελί ενός πίνακα ''thesi'' που έχουμε κατασκευάσει ο οποίος αναπαριστά τη λίστα υποψηφίων. Στη συνέχεια η στήλη του πίνακα Β στην οποία αυτός αντιστοιχεί απειρίζεται έτσι ώστε όταν επαναλάβουμε τη διαδικασία εύρεσης πλησιέστερου πελάτη να επιλεχθεί ο δεύτερος πλησιέστερος πελάτης, στην επόμενη επανάληψη ο τρίτος κ.ο.κ. Οι επαναλήψεις θα τερματίσουν όταν ο πίνακας ''thesi'' έχει γεμίσει με πλήθος στοιχείων ίσο με τη τιμή της μεταβλητής ''arithmos'' που είχε επιλεχθεί προηγουμένως. Τέλος και αφού έχουμε κατασκευάσει την επιθυμητή λίστα υποψηφίων, με την εντολή randsample(thesi,1) θα επιλεχθεί τυχαία ένα στοιχείο της λίστας για το οποίο στη συνέχεια θα πραγματοποιηθεί έλεγχος εάν είναι εφικτή η μετάβασης του οχήματος σε αυτό, σύμφωνα με τους περιορισμούς μέγιστου χρόνου και χωρητικότητας ακριβώς όπως και στον αλγόριθμο του πλησιέστερου γείτονα. Στη περίπτωση που οι πελάτες οι οποίοι δεν έχουν εξυπηρετηθεί ακόμα είναι μικρότεροι σε αριθμό από τη μεταβλητή ''arithmos'', η εύρεση πλησιέστερου πελάτη πραγματοποιείται μόνο μία φορά, δηλαδή μέχρι να ευρεθεί ο πλησιέστερος στη θέση του

οχήματος ο οποίος και επιλέγεται, χωρίς να κατασκευάζεται λίστα υποψηφιότητας και να τυχαιοποιείται η επιλογή του πελάτη.

4.5 1-1 Ανταλλαγή

Όπως έχουμε ήδη αναφέρει, μετά τη δημιουργία της αρχικής εφικτής λύσης από κάποιον κατασκευαστικό αλγόριθμο, συνίσταται η επιπλέον εφαρμογή άλλης κατάλληλης μεθόδου για τη βελτίωση της λύσης. Πολλές φορές για το σκοπό αυτό χρησιμοποιούνται αλγόριθμοι τοπικής αναζήτησης όπως και στη συγκεκριμένη περίπτωση που γίνεται χρήση της 1-1 ανταλλαγής. Έτσι αφού έχουμε κατασκευάσει αρχικές λύσεις, στη πρώτη περίπτωση με τον αλγόριθμο του πλησιέστερου γείτονα, και στη δεύτερη με τον αλγόριθμο της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης, εφαρμόζουμε τον αλγόριθμο 1-1 ανταλλαγή για να εξετάσουμε εάν είναι δυνατή η βελτίωση των λύσεων που έχουν παραχθεί. Η διαδικασία αποτελείται από τρεις φάσεις με τις οποίες τελικά επιτυγχάνεται η τυχαία ανταλλαγή δύο στοιχείων του διανύσματος want που έχει κατασκευαστεί από προηγούμενη μέθοδο, χωρίς να παραβιάζονται οι περιορισμοί του προβλήματος.

Με τη χρήση των μεθόδων του πλησιέστερου γείτονα και της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης έχουν παραχθεί λύσεις οι οποίες περιλαμβάνουν έναν αριθμό οχημάτων και επισκέψεων στην αποθήκη, προκειμένου να υπάρχει συμφωνία με τους περιορισμούς του προβλήματος. Όμως εφόσον πρόκειται να εφαρμόσουμε ανταλλαγή πελατών, υπάρχει το ενδεχόμενο να παρουσιαστούν διαφοροποιήσεις στις διαδρομές από και προς την αποθήκη. Για το λόγο αυτό στη πρώτη φάση υλοποίησης της 1-1 ανταλλαγής, θέλουμε να ''διαγράψουμε'' τις επισκέψεις που έχουν πραγματοποιηθεί στην αποθήκη από το διάνυσμα της αρχικής λύσης, προκειμένου στη συνέχεια και εφόσον έχει πραγματοποιηθεί η επιθυμητή ανταλλαγή πελατών, να επανεξετάσουμε εκ νέου τους περιορισμούς.

Έτσι χρησιμοποιείται μία επαναληπτική διαδικασία κατά την οποία η σειρά επίσκεψης πελατών που έχει οριστεί στο διάνυσμα want της αρχικής λύσης εκχωρείται σε ένα διάνυσμα want3. Ταυτόχρονα πραγματοποιείται έλεγχος έτσι ώστε όταν εμφανίζεται ο αριθμός 1, έστω για παράδειγμα στη θέση i=6 του διανύσματος want, που δηλώνει επίσκεψη του οχήματος στην αποθήκη να παραλείπεται και στη θέση του να εκχωρείται το στοιχείο i+1=7, δηλαδή ο επόμενος πελάτης του προβλήματος. Πρέπει να σημειωθεί ότι αυτό συμβαίνει μόνο για ενδιάμεσες επισκέψεις του οχήματος στην αποθήκη, αφού κάτι τέτοιο δεν απαιτείται για το πρώτο και το τελευταίο στοιχείο του διανύσματος τα οποία είναι βέβαιο ότι θα παραμείνουν 1(ένα όχημα πάντα θα ξεκινάει από την αποθήκη και αφού έχουν εξυπηρετηθεί όλοι οι πελάτες κάποιο όχημα θα επιστρέφει σε αυτή.) Έστω για παράδειγμα ότι έχουμε το διάνυσμα:

want =1 2 3 1 6 5 1 4 1

το οποίο έχει προκύψει από την εφαρμογή του αλγόριθμου του πλησιέστερου γείτονα. Μετά το τέλος της διαδικασίας που περιγράψαμε παραπάνω θα έχει κατασκευαστεί ένα νέο διάνυσμα: want3 =1 2 3 6 5 4 1.

Στη συνέχεια θέλουμε να πραγματοποιήσουμε την ανταλλαγή δύο τυχαίων στοιχείων του διανύσματος want3. Έτσι στη δεύτερη φάση της διαδικασίας με την εντολή randsample(want3,1) επιλέγεται ένα τυχαίο στοιχείο του διανύσματος want3 το οποίο εκχωρείται σε μία μεταβλητή l και με χρήση της ίδιας εντολής ένα άλλο τυχαίο στοιχείο του διανύσματος θα εκχωρείται σε μια μεταβλητή r. Ακόμη πραγματοποιείται κατάλληλος έλεγχος που μας εγγυάται ότι τα στοιχεία l και r που έχουν επιλεχθεί είναι διαφορετικά μεταξύ τους και διάφορα του 1. Έπειτα ακολουθεί μια επαναληπτική διαδικασία για την εισαγωγή των στοιχείων του want3 σε ένα νέο διάνυσμα want5 κατά την οποία πραγματοποιείται κατάλληλος έλεγχος έτσι ώστε όταν έρχεται η σειρά για την εισαγωγή του στοιχείου l στη θέση του να εκχωρείται το στοιχείο r και αντίστοιχα στη θέση του στοιχείου r να εκχωρείται το στοιχείο l. Εάν για παράδειγμα έχουμε want3 =1 2 3 6 5 4 1, l=3 και r=5 τότε το νέο διάνυσμα θα έχει τη μορφή: want5 = 1 2 5 6 3 4 1.

Τέλος απομένει η τρίτη φάση της διαδικασίας στην οποία γίνεται έλεγχος των περιορισμών και επιστροφή των οχημάτων στην αποθήκη όταν κρίνεται απαραίτητο. Το τελικό διάνυσμα το οποίο περιλαμβάνει τις απαραίτητες επισκέψεις στην αποθήκη θα είναι το want4 ενώ ακόμη υπολογίζεται το συνολικό κόστος διαδρομής μετά την ανταλλαγή των πελατών το οποίο εκχωρείται στη μεταβλητή count2.

Η παραπάνω διαδικασία πραγματοποιείται προκειμένου να ελεγχθεί η περαιτέρω βελτίωση τόσο της λύσης που έχει παραχθεί από τον αλγόριθμο του πλησιέστερου γείτονα όσο και των λύσεων που παράγονται από τον αλγόριθμο της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης.

Στη πρώτη περίπτωση για τον αλγόριθμο του πλησιέστερου γείτονα έχουμε δημιουργήσει μία αρχική λύση want με ένα κόστος διαδρομής nearestcost. Πριν την έναρξη της διαδικασίας ανταλλαγής κόμβων θα εισάγουμε το διάνυσμα want σε ένα νέο διάνυσμα bestroute και τη μεταβλητή nearestcost σε μια μεταβλητή best. Στη συνέχεια θα εφαρμόσουμε τη μέθοδο 1-1 ανταλλαγή και στο τέλος της διαδικασίας θα ελέγξουμε εάν η λύση που βρήκαμε είναι καλύτερη από την ήδη υπάρχουσα. Εάν κάτι τέτοιο ισχύει, τότε η μεταβλητή best θα πάρει τη τιμή count2, δηλαδή του κόστους που βρέθηκε μετά την ανταλλαγή των πελατών και αντίστοιχα το διάνυσμα bestroute θα αντικατασταθεί με το want4. Η διαδικασία ανταλλαγής κόμβων του αρχικού διανύσματος want πραγματοποιείται για έναν επιθυμητό αριθμό επαναλήψεων και κάθε φορά ελέγχεται εάν η λύση που παράχθηκε είναι αρκετά συμφέρουσα ώστε να αντικαταστήσει τη προηγούμενη βέλτιστη λύση που είχε βρεθεί.

Στη περίπτωση της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης υπάρχει η διαφορά ότι παράγουμε περισσότερες από μία αρχικές λύσεις, δηλαδή και η grasp πραγματοποιείται για έναν επιθυμητό αριθμό επαναλήψεων. Έτσι τη πρώτη φορά η αρχική εφικτή λύση που κατασκευάζεται αποθηκεύεται ως βέλτιστη. Στη συνέχεια πραγματοποιείται η 1-1 ανταλλαγή πελατών και όπως προηγουμένως εάν η λύση που προκύπτει θεωρείται πιο συμφέρουσα αντικαθιστά την αρχική. Όμως η 1-1 ανταλλαγή επαναλαμβάνεται και αυτή για έναν αριθμό επαναλήψεων οπότε ενδέχεται πάλι η βέλτιστη λύση να αλλάξει. Έτσι η δεύτερη αρχική λύση που θα παραχθεί στην επόμενη επανάληψη του αλγόριθμου grasp ενδέχεται να συγκριθεί με κάποια από τις λύσεις της 1-1 ανταλλαγής εάν αυτή είχε επιλεχθεί να αντικαταστήσει τη πρώτη αρχική λύση. Η διαδικασία θα τερματίσει όταν έχει ολοκληρωθεί ο επιθυμητός αριθμός επαναλήψεων του αλγορίθμου grasp.

4.6 Παρουσίαση του κώδικα σε μορφή ψευδοκώδικα

Πλησιέστερος γείτονας

Αρχικοποίηση

Ανάγνωση δεδομένων

Κατασκευή πίνακα κόστους

Δημιουργία αρχική λύσης με τη μέθοδο του πλησιέστερου γείτονα(NN_solution)

Υπολογισμός κόστους αρχικής λύσης(cost)

Best_solution= NN_solution

Best_cost=cost

1-1 exchange

Όσο δεν έχει ολοκληρωθεί ο μέγιστος αριθμός επαναλήψεων

local_solution=NN_solution

Άλλαξε τη local_solution με τη μέθοδο 1-1 exchange

Υπολόγισε το κόστος της νέας local_solution(local_cost)

Αν η νέα λύση βελτιώνει το κόστος

Όρισε Best_solution=local_solution

Όρισε Best_cost=local_cost

Τέλος_αν

Τέλος_όσο

Επέστρεψε Best_solution και Best_cost

GRASP

Αρχικοποίηση

Ανάγνωση δεδομένων

Κατασκευή πίνακα κόστους

Best_cost=inf

Όσο δεν έχει ολοκληρωθεί ο μέγιστος αριθμός επαναλήψεων

Δημιουργία αρχική λύσης με τη μέθοδο GRASP(GRASP_solution)

Υπολογισμός κόστους αρχικής λύσης(GRASP_cost)

Αν η αρχική λύση βελτιώνει το κόστος

Best_solution= GRASP_solution

Best_cost= GRASP_cost

Τέλος_αν

1-1 exchange

Όσο δεν έχει ολοκληρωθεί ο μέγιστος αριθμός επαναλήψεων

local_solution= GRASP_solution

Άλλαξε τη local_solution με τη μέθοδο 1-1 exchange

Υπολόγισε το κόστος της νέας local_solution(local_cost)

Αν η νέα λύση βελτιώνει το κόστος

Όρισε Best_solution=local_solution

Όρισε Best_cost=local_cost

Τέλος_αν

Τέλος_όσο

Τέλος_όσο

Επέστρεψε Best_solution και Best_cost

Κεφάλαιο 5: Υπολογιστικά αποτελέσματα

5.1 Περιγραφή και αναπαράσταση αποτελεσμάτων

Αφού κωδικοποιήσαμε τις μεθόδους όπως περιγράφτηκε στο προηγούμενο κεφάλαιο στο προγραμματιστικό περιβάλλον matlab, ακολούθησε εφαρμογή του εν λόγο κώδικα για ένα πλήθος παραδειγμάτων και επαναλήψεων προκειμένου να διασφαλίσουμε την εγκυρότητα των συμπερασμάτων μας. Τα 14 προβλήματα αναφοράς που χρησιμοποιήθηκαν έχουν προταθεί από τον Christoforides [2] ενώ παρουσιάζουν μεταξύ τους διαφοροποιήσεις όσων αφορά το πλήθος των πελατών, τη ζήτηση κάθε πελάτη, τους χρόνους εξυπηρέτησης και τους περιορισμούς που αφορούν τη χωρητικότητα των οχημάτων και το μέγιστο χρόνο που κάθε ένα από αυτά σπαταλά στην εξυπηρέτηση. Πιο συγκεκριμένα για κάθε ένα παράδειγμα, ο αριθμός επαναλήψεων της μεθόδου grasp διατηρήθηκε σταθερός στις 100 επαναλήψεις με μέγεθος λίστας υποψηφίων ίσο με 2, ενώ για την 1-1 exchange, τόσο για το πρόβλημα του πλησιέστερου γείτονα όσο και τη grasp, πραγματοποιήθηκαν 10 δοκιμές στις 50, 100, 200, 500 και 1000 επαναλήψεις.

Στη συνέχεια παρουσιάζονται τα κόστη όπως προέκυψαν για κάθε παράδειγμα.

- **Παράδειγμα 1**

  αριθμός πελατών: 50

  χωρητικότητα οχήματος: 160

  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος

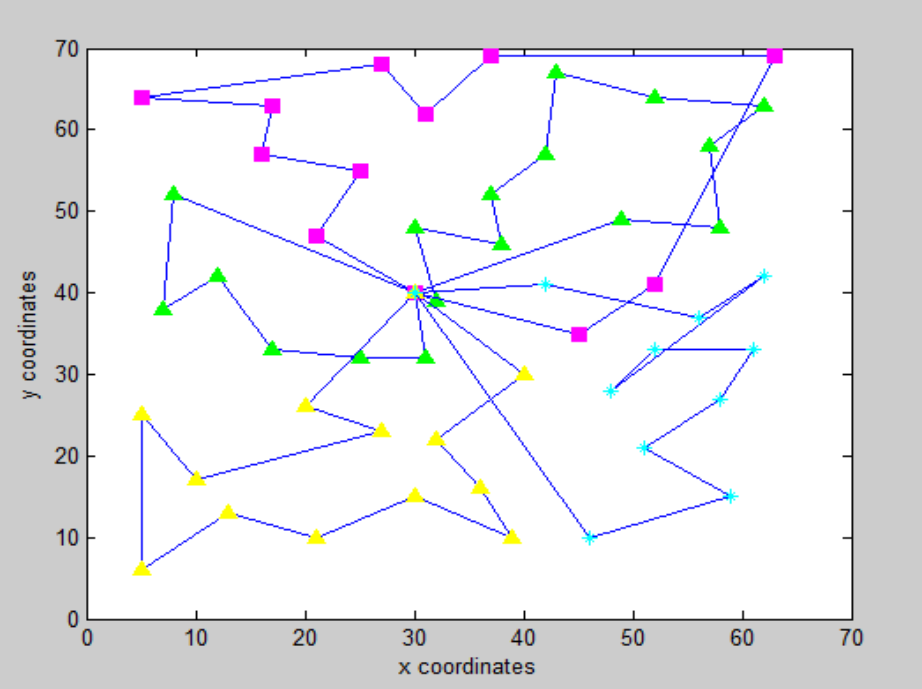
  χρόνος εξυπηρέτησης: 0

PARADEIGMA 1										
epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	710,3382	689,6395	705,334	688,538	710,3382	696,3	688,3176	652,93	688,3176	673,683
2	709,0496	686,9476	688,3176	696,305	688,3176	688,3	688,3176	678,457	688,3176	704,837
3	710,3382	701,9171	710,3382	689,177	708,7467	671,22	688,3176	656,102	694,5937	685,599
4	710,3382	693,4302	709,7167	702,046	705,334	681,78	710,3382	677,796	694,5937	649,05
5	710,3382	691,7375	710,3382	708,513	705,334	670,02	680,4368	676,519	680,4368	699,284
6	710,3382	696,0948	694,5937	699,66	708,7467	684,01	694,5937	682,112	688,3176	653,17
7	710,3382	663,5736	680,4368	688,629	707,8632	686,6	704,1077	682,652	704,1077	664,78
8	710,3382	684,9099	705,334	683,524	704,1077	706,18	710,3382	682,284	680,4368	660,661
9	704,1077	699,0319	710,3382	716,26	694,5937	706,22	708,7467	692,266	704,1077	656,127
10	709,0496	687,288	709,7167	708,183	710,3382	695,94	694,5937	677,207	704,1077	659,334
mesos oros	709,45743	689,457	702,44641	698,083	704,372	688,66	696,8108	675,832	692,73369	670,652

Πίνακας 5.1: Υπολογιστικά αποτελέσματα για το παράδειγμα 1

Στο συγκεκριμένο παράδειγμα, τα αποτελέσματα που παράγονται με τη μέθοδο grasp είναι στη πλειοψηφία τους καλύτερα από αυτά που επιστρέφει η μέθοδος του πλησιέστερου γείτονα. Συγκεκριμένα, το χαμηλότερο κατά μέσο όρο κόστος εμφανίζεται για τη μέθοδο grasp στις 1000 επαναλήψεις 1-1 exchage με τιμή 670,652, ενώ για τις ίδιες επαναλήψεις έχουμε και το καλύτερο μέσο κόστος για τον αλγόριθμο του πλησιέστερου γείτονα με τιμή 692,73369.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 1000 επαναλήψεις 1-1 exchange και 4η εφαρμογή του κώδικα.



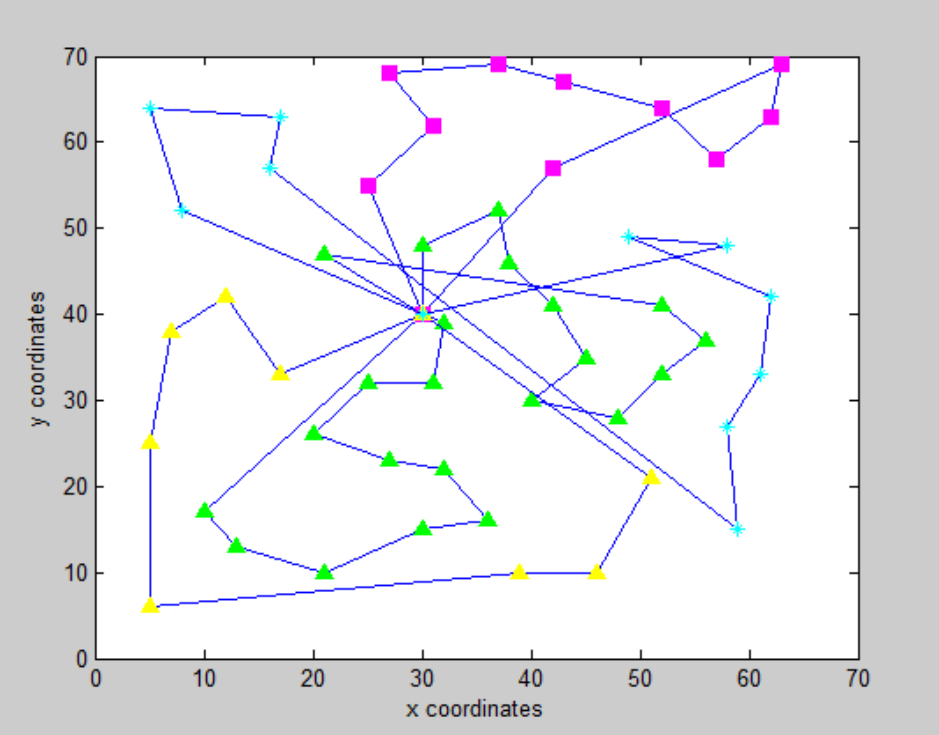
Γράφημα 5.1: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 1

Συνολικό κόστος: 649,0497

Διαδρομή:

1 47 28 33 2 23 29 4 36 21 30 3 1 13 48 19 15 26 25 1 7 49 24
8 44 27 9 32 37 17 39 1 6 38 16 46 45 43 20 41 14 42 18 5 1 34
40 11 31 35 10 50 22 51 12 1

Το μικρότερο κόστος για τη μέθοδο πλησιέστερου γείτονα εμφανίζεται στις 100 επαναλήψεις 1-1 exchange και $7^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.2: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 1

Συνολικό κόστος: 680,4368

Διαδρομή:

1 47 13 48 5 18 38 16 45 43 20 42 1 28 2 33 12 39 6 50 10 51 17
7 1 49 9 27 32 29 4 21 36 37 23 1 19 15 26 14 41 46 34 11 1 30
3 22 35 31 40 24 8 44 25 1

- **Παράδειγμα 2**

  αριθμός πελατών: 75

  χωρητικότητα οχήματος: 140

  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος

  χρόνος εξυπηρέτησης: 0

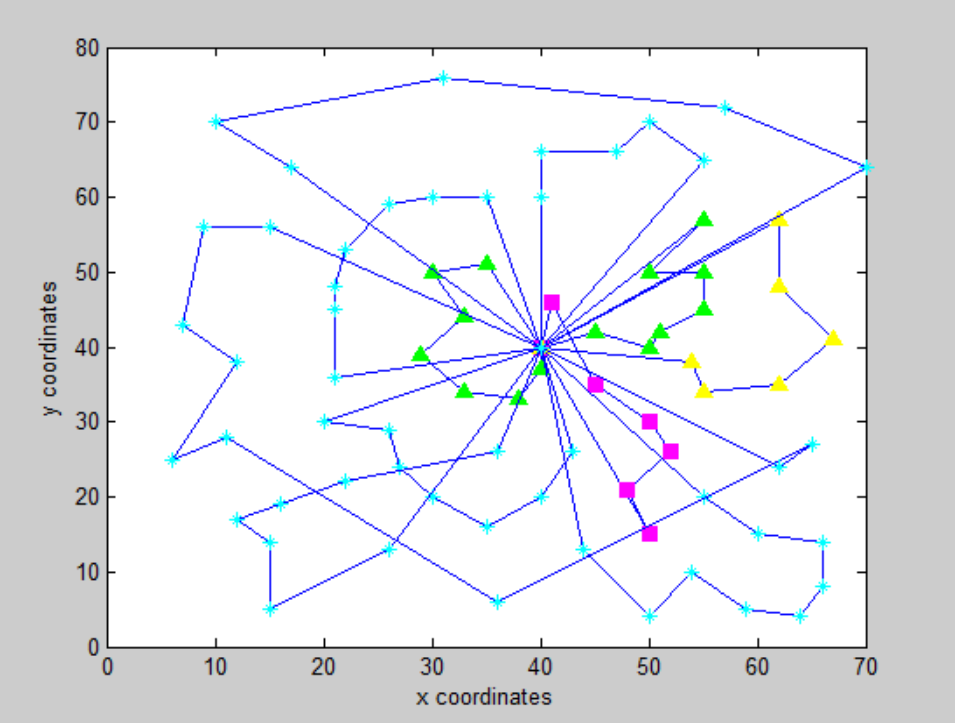
PARADEIGMA 2

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1000,7	1064,7	1000,7	1067,8	1000,7	1078	1000,7	1079,3	1000,7	1083,9
2	1000,7	1077	1000,7	1078,9	1000,7	1070	1000,7	1050,1	1000,7	1062,9
3	1000,7	1076,5	1000,7	1079,3	1000,7	1022,3	1000,7	1069,8	1000,7	1082,4
4	1000,7	1054,1	1000,7	1106	1000,7	1082,4	1000,7	1081,4	1000,7	1077,8
5	1000,7	1059,3	1000,7	1067,6	1000,7	1088,6	1000,7	1058,5	1000,7	1042,1
6	1000,7	1049,2	1000,7	1061,1	1000,7	1080,3	1000,7	1043,4	1000,7	1038,6
7	1000,7	1099,2	1000,7	1040,5	1000,7	1094,6	1000,7	1079,9	1000,7	1063,4
8	1000,7	1041,5	1000,7	1064,1	1000,7	1070,8	1000,7	1062,8	1000,7	1090,4
9	1000,7	1069,9	1000,7	1076	1000,7	1094,3	1000,7	1052,9	1000,7	1018,9
10	1000,7	1059	1000,7	1070	1000,7	1051,7	1000,7	1037	1000,7	1091
mesos oros	1000,7	1065,04	1000,7	1071,13	1000,7	1073,3	1000,7	1061,51	1000,7	1065,14

Πίνακας 5.2: Υπολογιστικά αποτελέσματα για το παράδειγμα 2

Στο παράδειγμα αυτό παρατηρούμε ότι η αρχική λύση της μεθόδου του πλησιέστερου γείτονα παρουσιάζει τη καλύτερη τιμή 1000,7, αφού δεν επιτυγχάνεται η εύρεση κάποιας μικρότερης τιμής για καμία από τις επαναλήψεις 1-1 exchange της ίδιας μεθόδου αλλά ούτε και με τη μέθοδο grasp. Συγκεκριμένα ο αλγόριθμος grasp εμφανίζει το μικρότερο μέσο κόστος για τις 500 επαναλήψεις 1-1 exchange με τιμή 1061,51, μεγαλύτερη δηλαδή από το 1000,7 που επιστρέφει ο πλησιέστερος γείτονας.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου του πλησιέστερου γείτονα και πρόκειται για την αρχική λύση.

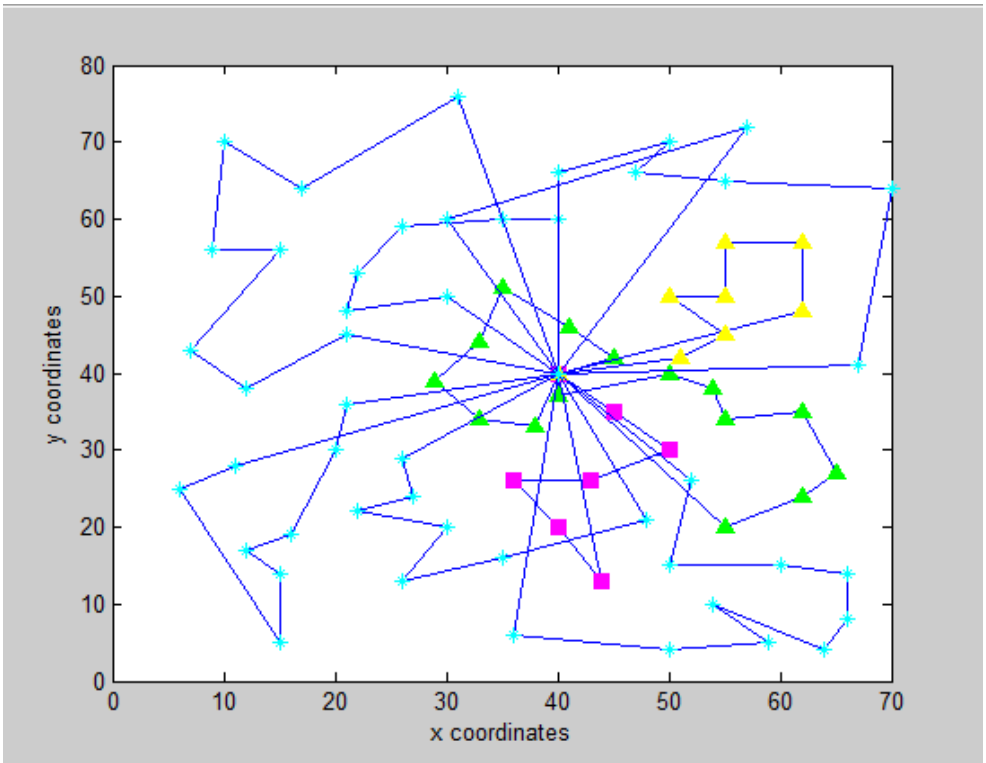


Γράφημα 5.3: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδειγμα 2

Συνολικό κόστος: 1000,7

Διαδρομή: 1 76 69 7 52 18 41 13 1 68 35 47 9 36 8 54 1 27 5 46 30
49 48 1 53 28 14 55 20 15 1 31 75 29 63 74 34 64 1 3 2 44 42 43
65 23 1 17 4 45 33 10 40 73 1 59 11 39 66 12 1 6 38 21 71 61 72
37 70 22 1 16 58 62 24 57 50 25 19 51 1 26 56 32 67 60 1

Το μικρότερο κόστος για τη μέθοδο grasp εμφανίζεται στις 1000 επαναλήψεις 1-1 exchange και $9^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.4: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο grasp για το παράδειγμα 2

Συνολικό κόστος: 1018,9

Διαδρομή:

1 68 27 13 18 52 7 69 1 76 35 53 28 14 58 16 6 1 5 46 31 3 75
22 1 47 9 8 36 54 15 20 1 34 74 2 63 23 29 49 1 30 48 38 21 71
61 37 72 70 62 1 41 45 33 10 73 59 1 4 50 25 51 19 56 26 32 1 17
64 44 42 43 65 57 24 1 11 66 39 12 60 55 1 67 40 1

- **Παράδειγμα 3**

  αριθμός πελατών: 100

  χωρητικότητα οχήματος: 200

  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος

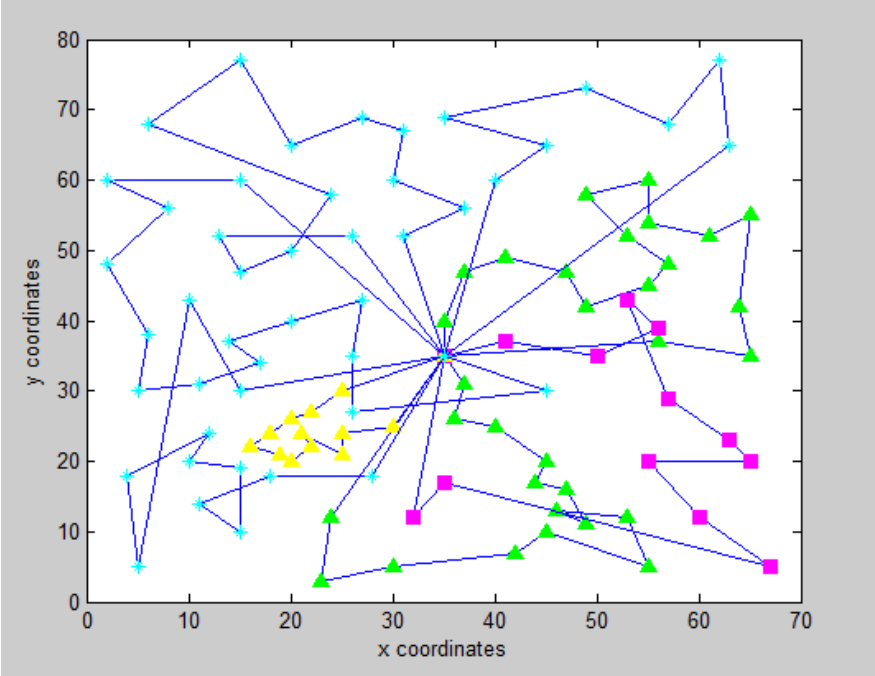
  χρόνος εξυπηρέτησης: 0

PARADEIGMA 3										
epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1096,1	1096,6	1096,1	1110,6	1096,1	1093,5	1096,1	1091,7	1096,1	1105,7
2	1096,1	1092,4	1096,1	1128	1096,1	1112,2	1096,1	1054,3	1096,1	1087,6
3	1096,1	1108	1096,1	1116,7	1096,1	1138,5	1096,1	1082,9	1096,1	1076,4
4	1096,1	1103,3	1096,1	1108,3	1096,1	1085,4	1096,1	1102,9	1096,1	1079,8
5	1096,1	1133,6	1096,1	1113,8	1096,1	1117,3	1096,1	1072,6	1096,1	1118,1
6	1096,1	1094,5	1096,1	1062,9	1096,1	1140,8	1096,1	1119,6	1096,1	1128,8
7	1096,1	1108,9	1096,1	1106,3	1096,1	1102,8	1096,1	1122,1	1096,1	1133,4
8	1096,1	1146,4	1096,1	1125,1	1096,1	1120,3	1096,1	1139,2	1096,1	1084,5
9	1096,1	1080,2	1096,1	1118,8	1096,1	1075,3	1096,1	1089,7	1096,1	1094
10	1096,1	1125,2	1096,1	1077,6	1096,1	1047	1096,1	1079,2	1096,1	1060,5
mesos oros	1096,1	1108,91	1096,1	1106,81	1096,1	1103,3	1096,1	1095,42	1096,1	1096,88

Πίνακας 5.3: Υπολογιστικά αποτελέσματα για το παράδειγμα 3

Όπως προηγουμένως, γίνεται αντιληπτό ότι και για το παράδειγμα 3 το καλύτερο μέσο κόστος είναι ίσο με την αρχική λύση του πλησιέστερου γείτονα με τιμή αυτή τη φορά 1096,1 ενώ για τη μέθοδο grasp το μικρότερο μέσο κόστος εμφανίζεται στις 1000 επαναλήψεις 1-1 exchange με τιμή 1096,88. Παρόλα αυτά παρατηρείται πως με τη μέθοδο grasp έχει επιτευχθεί αρκετές φορές κόστος διαδρομής μικρότερο από την αρχική λύση του πλησιέστερου γείτονα που προαναφέρθηκε όπως για παράδειγμα στην 6η εφαρμογή του κώδικα για 100 επαναλήψεις 1-1 exchange με τιμή 1062,9.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 200 επαναλήψεις 1-1 exchange και $10^{η}$ εφαρμογή του κώδικα.



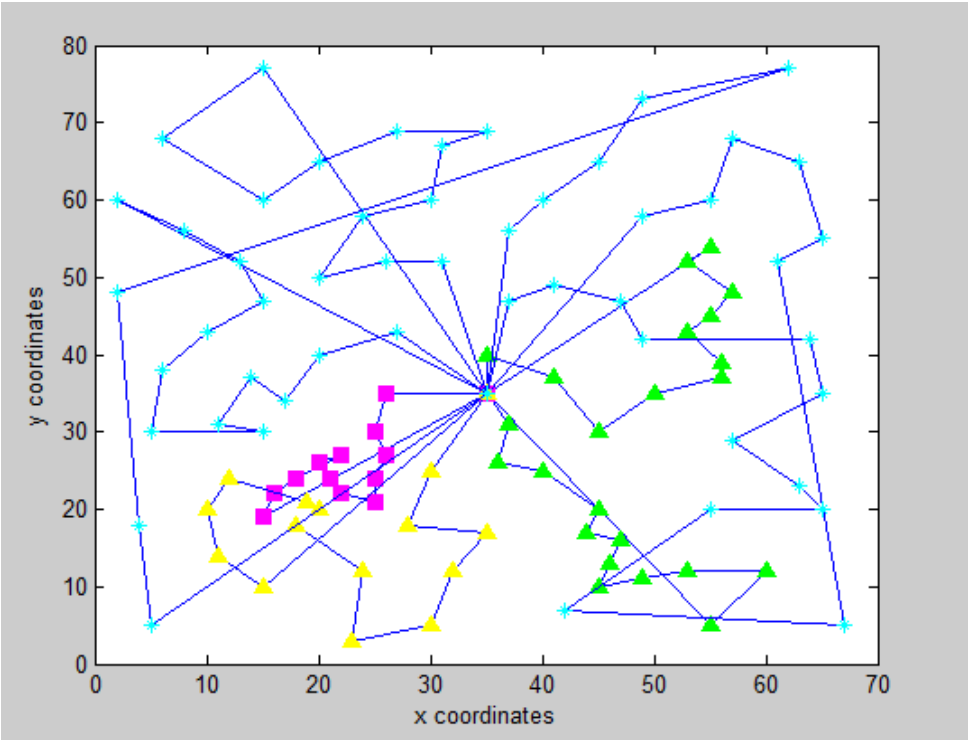
Γράφημα 5.5: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 3

Συνολικό κόστος: 1047

Διαδρομή:

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

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα είναι η αρχική του λύση.



Γράφημα 5.6: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 3

Συνολικό κόστος: 1096,1

Διαδρομή:

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

- **Παράδειγμα 4**

  αριθμός πελατών: 150

  χωρητικότητα οχήματος: 200

  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος

  χρόνος εξυπηρέτησης: 0

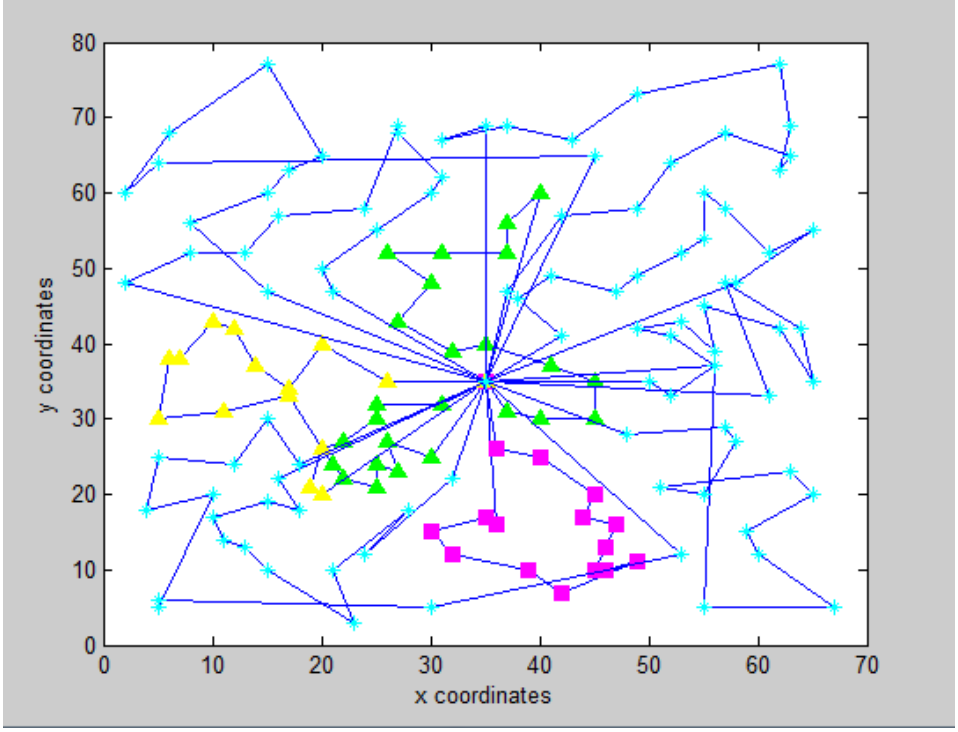
PARADEIGMA 4

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1367,7	1457,4	1367,7	1443,9	1367,7	1454	1367,7	1477,9	1367,7	1442,7
2	1367,7	1408,9	1367,7	1449	1367,7	1428,3	1367,7	1465,9	1367,7	1435,2
3	1367,7	1395,6	1367,7	1432,3	1367,7	1400,1	1367,7	1459,1	1367,7	1412,4
4	1367,7	1465,2	1367,7	1446,9	1367,7	1445,8	1367,7	1463,3	1367,7	1473,7
5	1367,7	1464,4	1367,7	1463,5	1367,7	1393,2	1367,7	1386,5	1367,7	1448,7
6	1367,7	1436,7	1367,7	1428,7	1367,7	1438,2	1367,7	1443,7	1367,7	1459,7
7	1367,7	1430,6	1367,7	1427,1	1367,7	1465,4	1367,7	1426,9	1367,7	1449,1
8	1367,7	1416,3	1367,7	1448,3	1367,7	1448,3	1367,7	1420,8	1367,7	1446,6
9	1367,7	1483,4	1367,7	1435,5	1367,7	1439,3	1367,7	1440,2	1367,7	1429,1
10	1367,7	1478,2	1367,7	1444,6	1367,7	1470,4	1367,7	1460,9	1367,7	1362,4
mesos oros	1367,7	1443,67	1367,7	1441,98	1367,7	1438,3	1367,7	1444,52	1367,7	1435,96

Πίνακας 5.4: Υπολογιστικά αποτελέσματα για το παράδειγμα 4

Και στο 4^{ο} παράδειγμα καθώς φαίνεται, η αρχική λύση της μεθόδου του πλησιέστερου γείτονα με τιμή 1367,7 είναι μικρότερη από τα μέσα κόστη των 10 επαναλήψεων της grasp από τα οποία το χαμηλότερο ισούται με 1435,96 για τις 1000 επαναλήψεις 1-1 exchange.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 1000 επαναλήψεις 1-1 exchange και 10^n εφαρμογή του κώδικα.



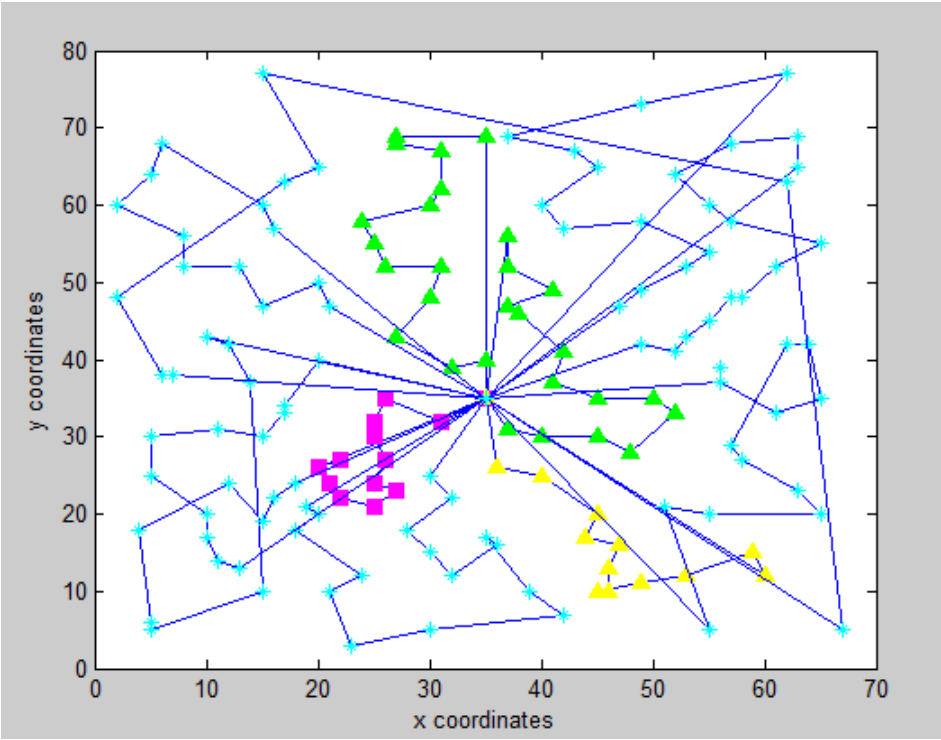
Γράφημα 5.7: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 4

Συνολικό κόστος: 1362,4

Διαδρομή:

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

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα είναι η αρχική του λύση.



Γράφημα 5.8: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 4

Συνολικό κόστος: 1367,7

Διαδρομή:

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

- **Παράδειγμα 5**

  αριθμός πελατών: 199

  χωρητικότητα οχήματος: 200

  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος

  χρόνος εξυπηρέτησης: 0

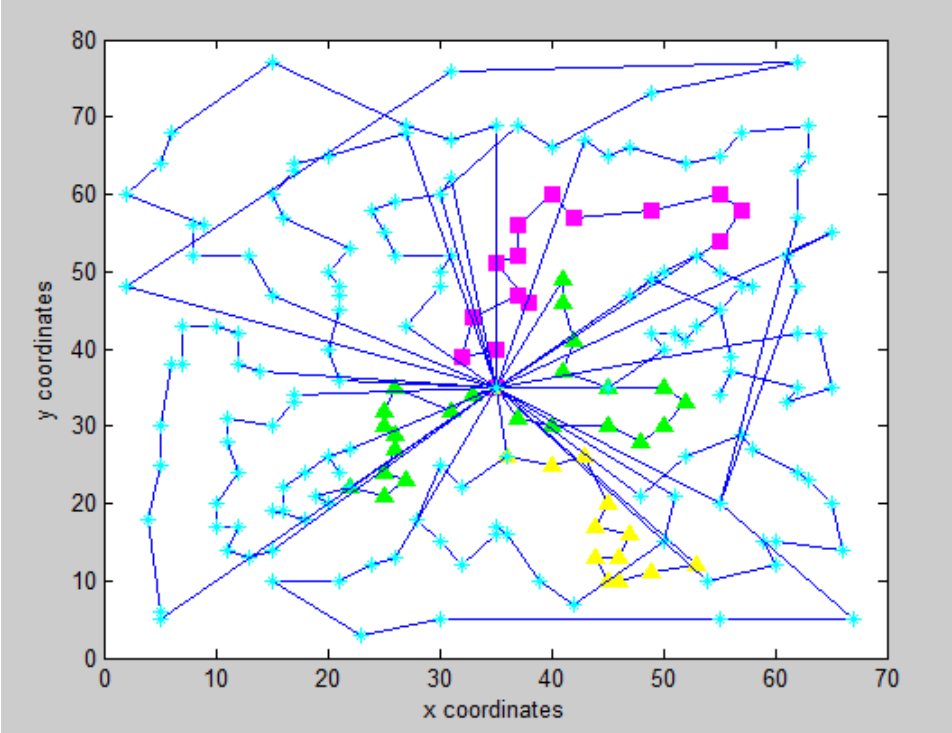
PARADEIGMA 5

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1646,7	1751,8	1646,7	1775,6	1646,7	1792,6	1646,7	1782,6	1646,7	1741,3
2	1646,7	1780,3	1646,7	1766,6	1646,7	1811,9	1646,7	1784,7	1646,7	1790,2
3	1646,7	1772,4	1646,7	1770,2	1646,7	1780,6	1646,7	1759,3	1646,7	1784,7
4	1646,7	1756	1646,7	1745	1646,7	1774,1	1646,7	1743,4	1646,7	1782,5
5	1646,7	1757,5	1646,7	1793,2	1646,7	1782,5	1646,7	1719,7	1646,7	1761,5
6	1646,7	1759,9	1646,7	1765,4	1646,7	1739,3	1646,7	1824,1	1646,7	1824,6
7	1646,7	1804,5	1646,7	1719,9	1646,7	1766	1646,7	1786,2	1646,7	1795,9
8	1646,7	1795,7	1646,7	1733,3	1646,7	1812,2	1646,7	1774,9	1646,7	1760,6
9	1646,7	1786,8	1646,7	1790,2	1646,7	1808,9	1646,7	1791,5	1646,7	1753,2
10	1646,7	1798,3	1646,7	1826,8	1646,7	1787,1	1646,7	1779,2	1646,7	1764,2
mesos oros	1646,7	1776,32	1646,7	1768,62	1646,7	1785,5	1646,7	1774,56	1646,7	1775,87

Πίνακας 5.5: Υπολογιστικά αποτελέσματα για το παράδειγμα 5

Και για το παράδειγμα 5 παρατηρούμε ότι το καλύτερο κόστος είναι το 1646,7, η τιμή δηλαδή της αρχικής λύσης του πλησιέστερου γείτονα. Για τη μέθοδο grasp, το χαμηλότερο μέσο κόστος εμφανίζεται στις 100 επαναλήψεις 1-1 exchange με τιμή 1768,62.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου του πλησιέστερου γείτονα και πρόκειται για την αρχική του λύση.



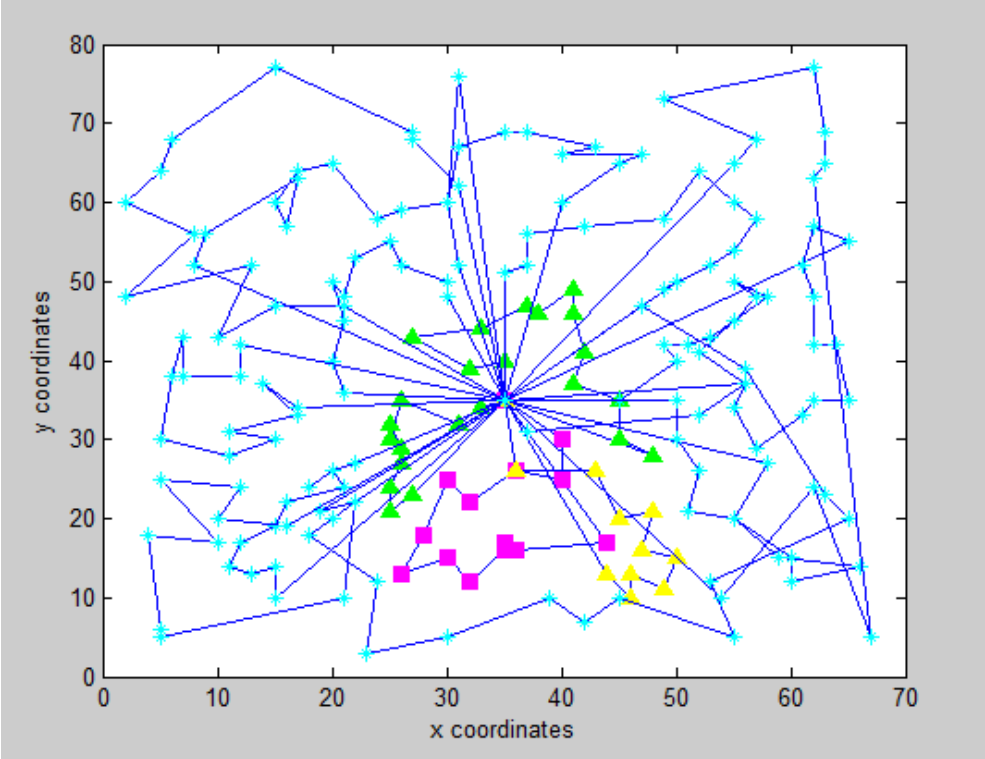
Γράφημα 5.9: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 5

Συνολικό κόστος: 1646,7

Διαδρομή:

1 157 113 90 148 7 184 95 96 118 98 93 1 54 106 27 150 196 110 13 139
29 112 177 2 1 28 147 168 70 133 163 102 71 31 123 52 10 121 82 1 59
41 181 22 74 73 75 172 23 134 76 57 1 153 138 14 88 145 58 179 3 116
146 42 198 111 1 155 185 77 197 117 78 4 80 130 186 34 158 1 53 128 191
32 89 149 63 160 11 109 1 167 19 154 107 195 8 183 124 20 108 176 12 127
1 97 100 60 105 94 86 194 92 101 38 99 152 1 51 103 159 69 81 178 151
164 135 25 30 122 1 61 119 6 85 174 62 17 142 192 45 120 193 1 199 180
55 131 166 56 26 171 188 140 40 187 1 84 200 115 9 175 126 46 18 114 87
141 39 1 83 49 125 48 169 37 144 50 65 64 91 33 1 173 43 143 15 44 16
24 68 5 170 79 35 1 156 165 136 36 137 72 162 104 189 21 129 1 190 132
161 67 66 182 47 1

Το μικρότερο κόστος για τη μέθοδο grasp εμφανίζεται στις 500 επαναλήψεις 1-1 exchange και $5^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.10: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο grasp για το παράδείγμα 5

Συνολικό κόστος: 1719,7

Διαδρομή:

1 157 113 90 7 148 184 95 96 98 118 1 28 147 53 168 70 133 2 177 112
29 139 27 150 1 106 41 59 138 14 88 173 145 58 179 3 116 74 1 153 181
22 199 73 198 76 75 134 172 1 155 185 77 197 78 4 80 186 130 159 117 81
1 128 191 89 149 183 195 8 154 19 167 1 13 196 180 111 156 188 40 171 140
5 131 1 54 110 151 178 55 135 164 25 30 122 170 79 165 35 1 163 102 71
123 52 104 10 121 82 34 103 158 1 51 69 56 26 57 24 23 42 146 16 44
43 101 38 93 99 1 32 11 160 63 12 176 20 124 108 169 125 1 97 100 94
105 60 86 92 192 45 120 193 15 1 61 84 119 85 6 174 18 46 175 126 200
115 1 107 83 9 49 47 48 37 144 50 65 64 127 109 1 194 17 62 114 142
87 141 39 143 152 1 31 21 189 161 129 132 33 91 190 182 1 187 166 68 136
36 137 66 67 72 162 1

- **Παράδειγμα 6**
  αριθμός πελατών: 50
  χωρητικότητα οχήματος: 160
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 200
  χρόνος εξυπηρέτησης: 10

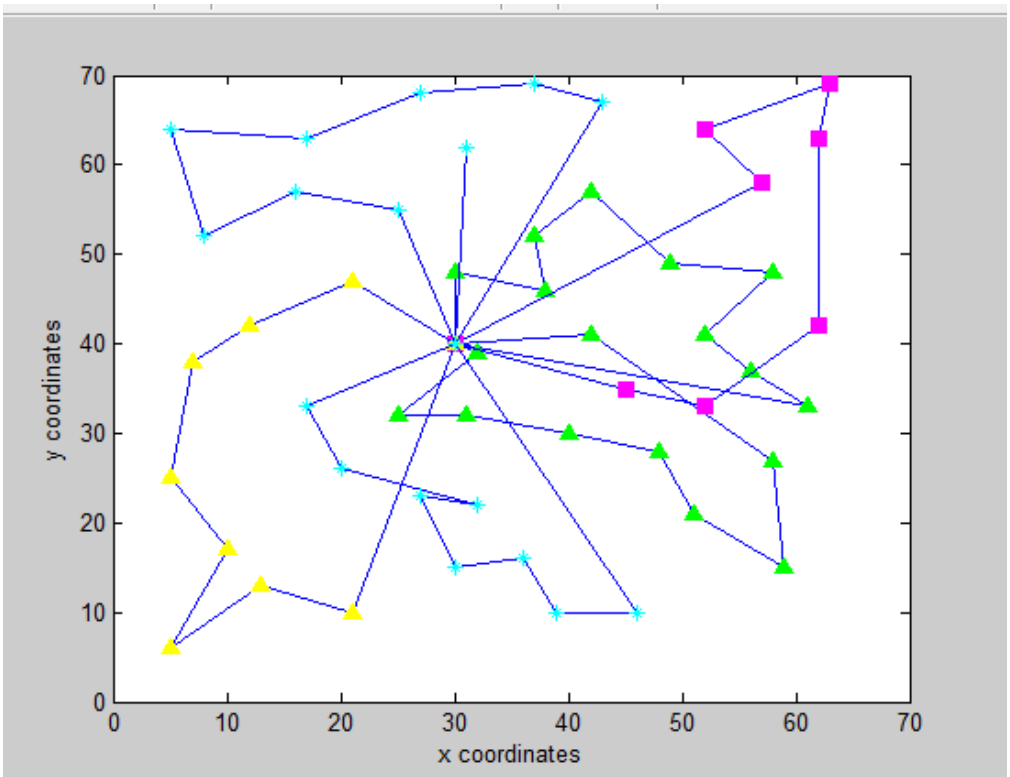
PARADEIGMA 6

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	784,502	726,1043	784,502	739,93	784,502	712,73	784,502	729,81	782,3456	712,99
2	784,502	699,0687	784,502	736,023	784,502	721,4	772,6388	716,316	772,6388	707,767
3	784,502	754,8265	784,502	736,589	784,502	742,97	784,502	733,85	772,6388	715,23
4	784,502	740,9549	782,3456	710,835	784,502	720,88	772,6388	738,746	784,502	728,629
5	784,502	698,7464	784,502	738,589	782,3456	714,17	772,6388	724,46	772,6388	726,792
6	784,502	726,5827	784,502	727,554	784,502	725,46	784,502	721,949	784,502	692,849
7	784,502	745,0688	784,502	747,747	784,502	717,42	784,502	726,391	784,502	717,376
8	784,502	727,7183	784,502	725,179	772,6388	735,81	784,502	733,443	772,6388	755,755
9	784,502	760,5881	784,502	733,364	784,502	743,27	772,6388	707,397	784,502	704,674
10	784,502	751,1243	772,6388	738,486	784,502	718,12	784,502	753,322	772,6388	720,211
mesos oros	784,502	733,0783	783,10004	733,43	783,1	725,22	779,75672	728,569	778,35476	718,227

Πίνακας 5.6: Υπολογιστικά αποτελέσματα για το παράδειγμα 6

Στο συγκεκριμένο παράδειγμα, η μέθοδος grasp είναι αυτή που επιστρέφει περισσότερο συμφέρουσες λύσεις σε σχέση με αυτή του πλησιέστερου γείτονα. Συγκεκριμένα, το χαμηλότερο μέσο κόστος εμφανίζεται στις 1000 επαναλήψεις 1-1 exchange της μεθόδου grasp με τιμή 718,227, ενώ στις ίδιες επαναλήψεις εμφανίζεται και η χαμηλότερη μέση τιμή της μεθόδου του πλησιέστερου γείτονα η οποία ισούται με τον αριθμό 778,35476.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 1000 επαναλήψεις 1-1 exchange και 6η εφαρμογή του κώδικα.



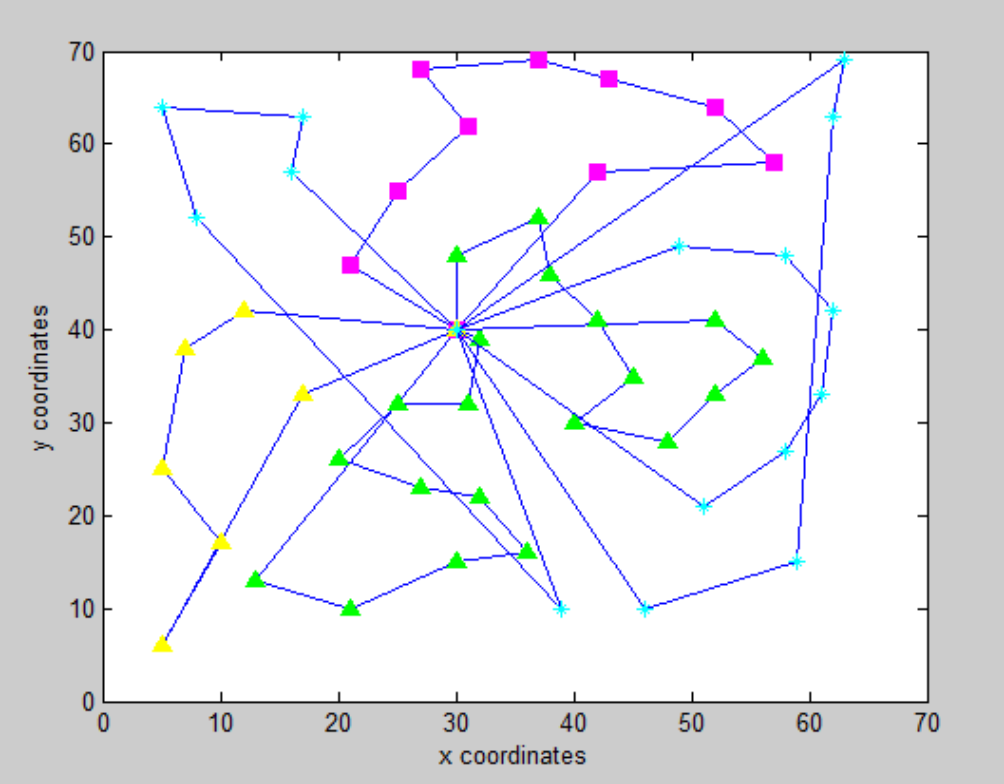
Γράφημα 5.11: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 6.

Συνολικό κόστος: 692,8492

Διαδρομή:

1 28 33 2 23 3 30 17 51 35 1 47 48 13 6 50 11 40 31 12 1 39 10
22 36 37 4 21 1 7 15 26 14 42 41 20 43 1 49 24 25 44 8 27 32 29
1 19 5 38 18 45 16 46 34 1 9 1

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα εμφανίζεται για τις 100, 200, 500 και 1000 επαναλήψεις της 1-1 exchange.



Γράφημα 5.12: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 6

Συνολικό κόστος: 772,6388

Διαδρομή:

1 47 13 48 5 18 38 16 45 43 20 1 28 2 33 12 39 6 50 10 51 17 1
7 49 9 27 32 29 4 21 23 1 15 26 14 42 41 19 1 3 30 22 35 31 11
1 24 8 44 25 46 1 34 40 36 37 1

- **Παράδειγμα 7**
  αριθμός πελατών: 75
  χωρητικότητα οχήματος: 140
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 160
  χρόνος εξυπηρέτησης: 10

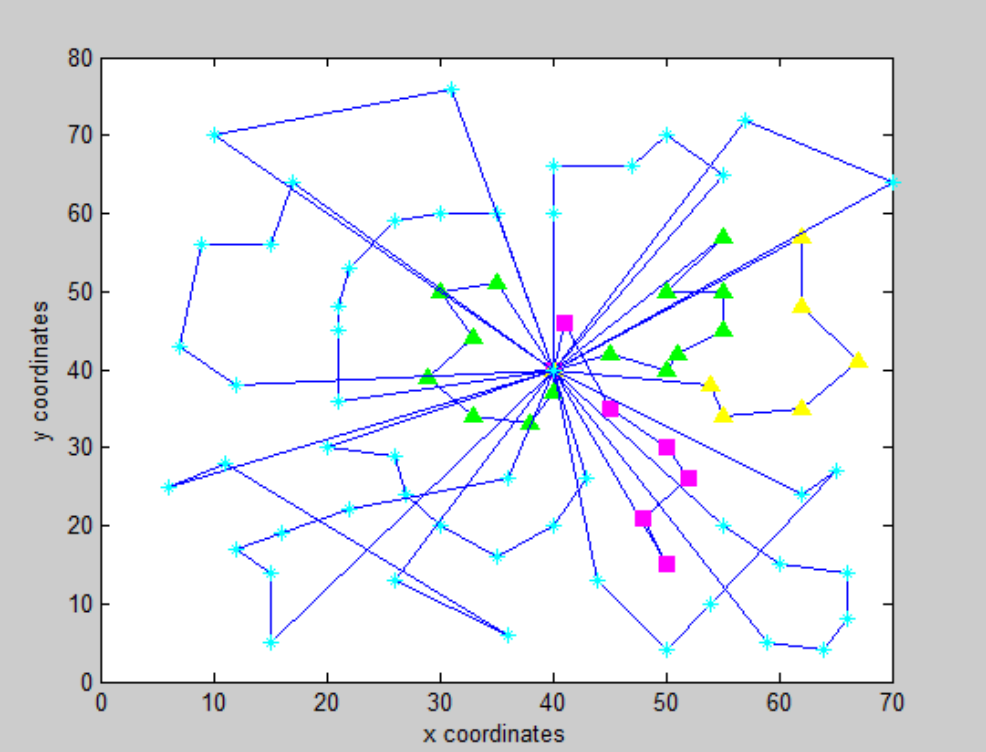
PARADEIGMA 7

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1192,2	1187,3	1192,2	1211,1	1192,2	1216,8	1175,8	1226,1	1175,8	1182,9
2	1189,7	1179,4	1169,3	1200,2	1184	1174,7	1169,3	1199,5	1175,8	1192,9
3	1192,2	1201,7	1192,2	1202,4	1192,2	1223,5	1185,7	1205,3	1175,8	1205,6
4	1192,2	1224,9	1184	1172,8	1192,2	1205,9	1169,3	1199,3	1179,3	1207,1
5	1192,2	1225,1	1192,2	1199,2	1184	1226,5	1189,1	1211	1169,3	1189,2
6	1192,2	1177,9	1192,2	1208,6	1184	1210,8	1179,3	1193,3	1169,3	1210,5
7	1192,2	1192,9	1189,1	1184,1	1192,2	1213,4	1175,8	1215,5	1169,3	1221,8
8	1189,1	1200,6	1192,2	1229,7	1185,7	1223	1169,3	1171	1169,3	1212,4
9	1192,2	1199,7	1192,2	1210,6	1175,8	1212,6	1169,3	1215,5	1169,3	1201,2
10	1192,2	1191,2	1189,1	1211,7	1189,7	1215,7	1190,3	1204,6	1179,9	1218,1
mesos oros	1191,64	1198,07	1188,47	1203,04	1187,2	1212,3	1177,32	1204,11	1173,31	1204,17

Πίνακας 5.7: Υπολογιστικά αποτελέσματα για το παράδειγμα 7

Στο παράδειγμα αυτό παρατηρείται ότι οι λύσεις που επιστρέφει ο αλγόριθμος του πλησιέστερου γείτονα είναι στη πλειοψηφία τους καλύτερες από αυτές της grasp. Συγκεκριμένα το μικρότερο μέσο κόστος εμφανίζει ο πλησιέστερος γείτονας στις 1000 επαναλήψεις 1-1 exchange με τιμή 1173,31 και ακολουθεί η grasp με καλύτερο μέσο κόστος 1198,07 στις 50 επαναλήψεις. Παρόλα αυτά, αξίζει να σημειωθεί ότι σε κάποιες επαναλήψεις, όπως για παράδειγμα στη 4η δοκιμή εφαρμογής του κώδικα και 100 επαναλήψεις 1-1 exchange, η grasp εμφανίζει μικρότερο κόστος διαδρομής.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου του πλησιέστερου γείτονα στις 100 επαναλήψεις 1-1 exchange και $2^{η}$ εφαρμογή του κώδικα.



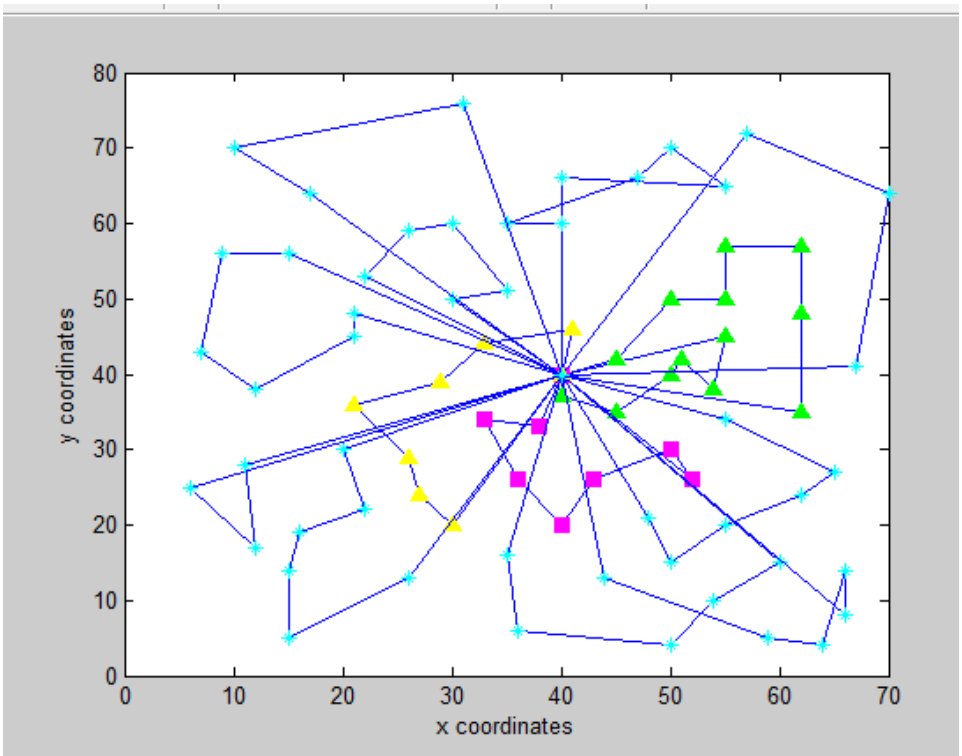
Γράφημα 5.13: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδειγμα 7

Συνολικό κόστος: 1169,3

Διαδρομή:

1 76 69 7 52 18 41 13 1 68 35 47 9 36 8 54 1 27 5 46 30 49 48
1 53 28 14 55 20 15 1 31 75 29 63 74 34 64 1 3 2 44 42 43 65 1
17 4 45 33 10 40 73 1 59 11 39 66 12 1 6 38 21 71 61 72 1 16 58
37 70 22 1 50 25 19 51 26 1 23 62 24 57 1 67 60 1 56 32 1

Το μικρότερο κόστος για τη μέθοδο grasp εμφανίζεται στις 500 επαναλήψεις 1-1 exchange και $8^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.14: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο grasp για το παράδειγμα 7

Συνολικό κόστος: 1171

Διαδρομή:

1 76 5 35 47 53 9 1 68 8 36 54 15 20 14 1 69 7 3 75 31 46 30 1
27 18 52 17 34 74 63 1 41 13 40 10 33 1 28 58 16 6 48 49 1 59 73
39 66 12 11 1 45 4 50 25 19 51 1 64 2 44 43 65 23 1 55 60 67 1
29 62 70 37 38 1 24 42 57 1 22 72 61 21 71 1 32 56 26 1

- **Παράδειγμα 8**
  αριθμός πελατών: 100
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 230
  χρόνος εξυπηρέτησης: 10

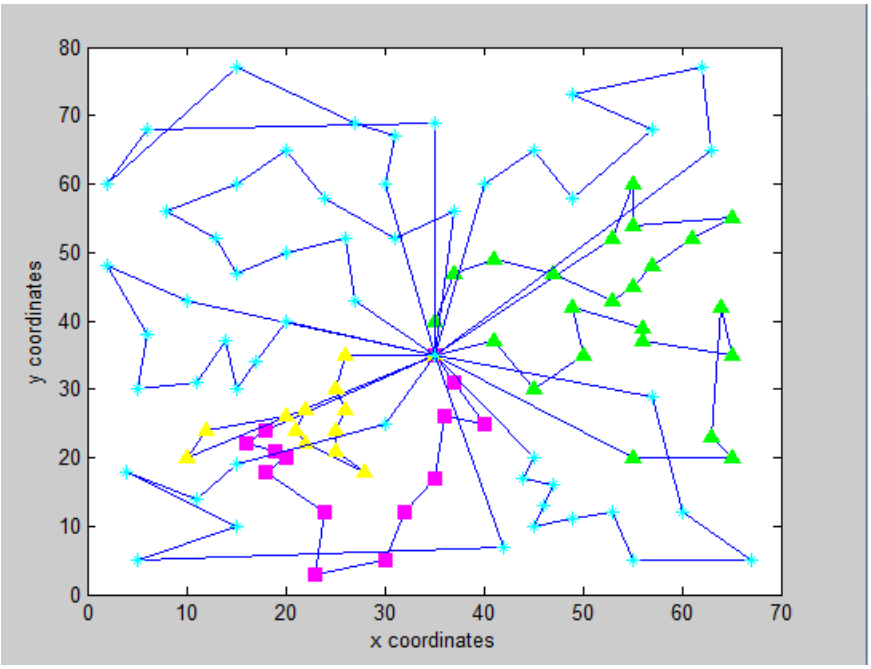
PARADEIGMA 8

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1260,6	1211,2	1260,6	1219,8	1254,2	1137	1255,7	1149,4	1227,1	1193
2	1260,6	1235,6	1260,6	1193,8	1259,2	1170,4	1259	1215,9	1249,8	1164,1
3	1260,6	1232,7	1249,8	1140,3	1260,6	1141,2	1258,1	1151,9	1255,1	1214,5
4	1260,6	1107,1	1260,1	1234,7	1260,6	1179,1	1260,3	1181,5	1255,3	1176,2
5	1260,6	1193,7	1260,6	1150,5	1250,5	1190	1250,5	1207,6	1255,3	1187,4
6	1260,6	1190,7	1260,6	1170,8	1260,3	1201,1	1259	1218,3	1257	1192,4
7	1260,6	1196	1260,6	1212	1260,6	1130,8	1250,5	1165,8	1250,3	1177,9
8	1260,6	1187,6	1260,6	1228	1250,5	1108,9	1255,3	1234,8	1249,8	1188,4
9	1260,6	1195,3	1260,6	1216,1	1257,6	1183,9	1258,8	1142,8	1255,3	1167,7
10	1260,6	1158,2	1260,1	1181	1260,6	1182,5	1260,6	1190,2	1250,3	1184,6
mesos oros	1260,6	1190,81	1259,42	1194,7	1257,47	1162,5	1256,78	1185,82	1250,53	1184,62

Πίνακας 5.8: Υπολογιστικά αποτελέσματα για το παράδειγμα 8

Στο 8^{ο} παράδειγμα βλέπουμε ότι η μέθοδος grasp αποδίδει καλύτερα σε σχέση με τον αλγόριθμό του πλησιέστερου γείτονα. Συγκεκριμένα, εμφανίζει μικρότερο μέσο κόστος, ίσο με 1162,5, στις 200 επαναλήψεις 1-1 exchange ενώ ο πλησιέστερος γείτονας στις 1000, με τιμή 1250,53.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 50 επαναλήψεις 1-1 exchange και $4^{η}$ εφαρμογή του κώδικα.



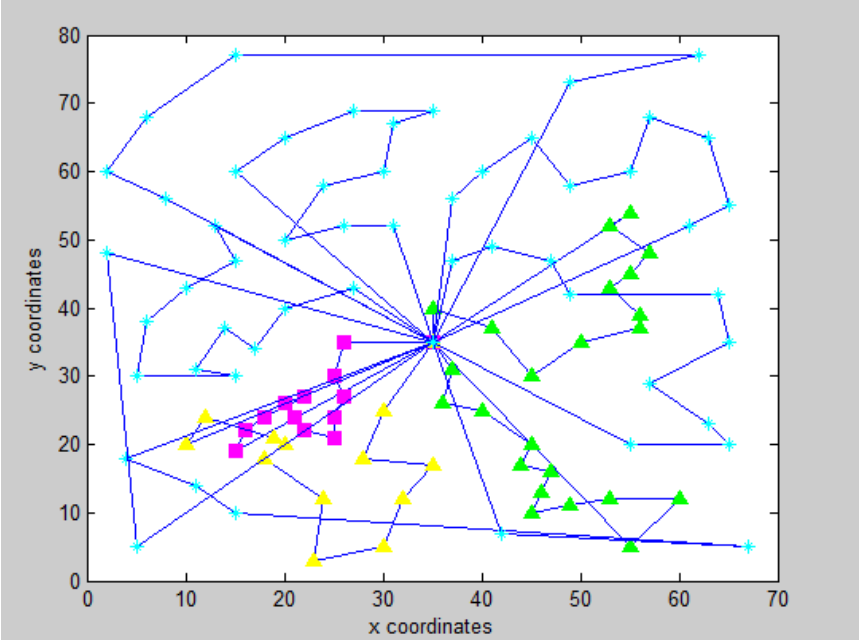
Γράφημα 5.15: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 8

Συνολικό κόστος: 1107,1

Διαδρομή:

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

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα εμφανίζεται για τις 1000 επαναλήψεις της 1-1 exchange και $1^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.16: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 8

Συνολικό κόστος: 1227,1

Διαδρομή:

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

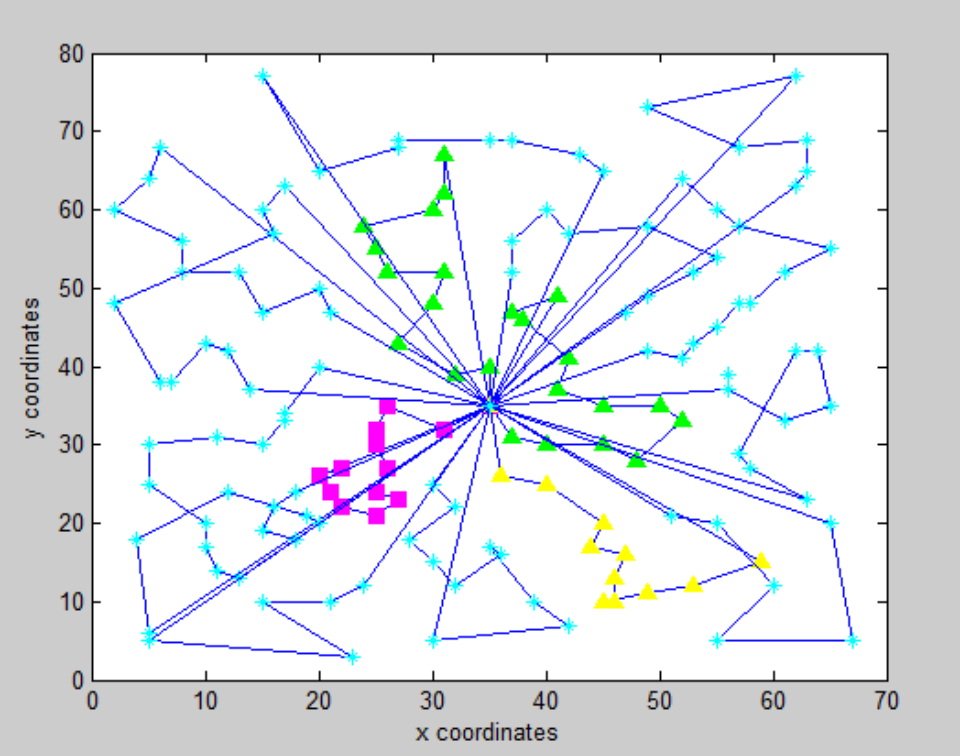
- **Παράδειγμα 9**
  αριθμός πελατών: 150
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 200
  χρόνος εξυπηρέτησης: 10

PARADEIGMA 9										
epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1485,4	1662,9	1485,4	1648,5	1485,4	1673,5	1485,4	1630,9	1485,4	1663,5
2	1485,4	1647,5	1485,4	1640,9	1485,4	1662,5	1485,4	1674,9	1485,4	1669,9
3	1485,4	1639,8	1485,4	1622	1485,4	1671,4	1485,4	1645,5	1485,4	1658,9
4	1485,4	1641,9	1485,4	1652,9	1485,4	1627,8	1485,4	1664,4	1485,4	1641,4
5	1485,4	1659,1	1485,4	1671,3	1485,4	1654,5	1485,4	1612,9	1485,4	1609,4
6	1485,4	1660,7	1485,4	1614,6	1485,4	1603,2	1485,4	1677,3	1485,4	1667
7	1485,4	1662,5	1485,4	1656,5	1485,4	1686	1485,4	1637,5	1485,4	1642
8	1485,4	1631,2	1485,4	1638,9	1485,4	1647,3	1485,4	1658,5	1485,4	1634,4
9	1485,4	1629,7	1485,4	1626,4	1485,4	1637,3	1485,4	1623,5	1485,4	1666,9
10	1485,4	1645,6	1485,4	1707,3	1485,4	1672,3	1485,4	1654,7	1485,4	1653,5
mesos oros	1485,4	1648,09	1485,4	1647,93	1485,4	1653,6	1485,4	1648,01	1485,4	1650,69

Πίνακας 5.9: Υπολογιστικά αποτελέσματα για το παράδειγμα 9

Από τον παραπάνω πίνακα γίνεται αντιληπτό, ότι για συγκεκριμένο παράδειγμα η αρχική λύση του πλησιέστερου γείτονα αποδίδει τη καλύτερη τιμή. Έτσι στον εν λόγω αλγόριθμο δε παρουσιάζονται βελτιώσεις με την ανταλλαγή πελατών της μεθόδου 1-1 exchage, επομένως και το μέσο κόστος διατηρεί σταθερά την αρχική τιμή 1485,4 για όλες τις επαναλήψεις. Η grasp φαίνεται να αποδίδει καλύτερα στις 100 επαναλήψεις 1-1 exchange όπου παρουσιάζει το ελάχιστο, σε σχέση με τις άλλες επαναλήψεις, μέσο κόστος 1647,93.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου του πλησιέστερου γείτονα και πρόκειται για την αρχική λύση.



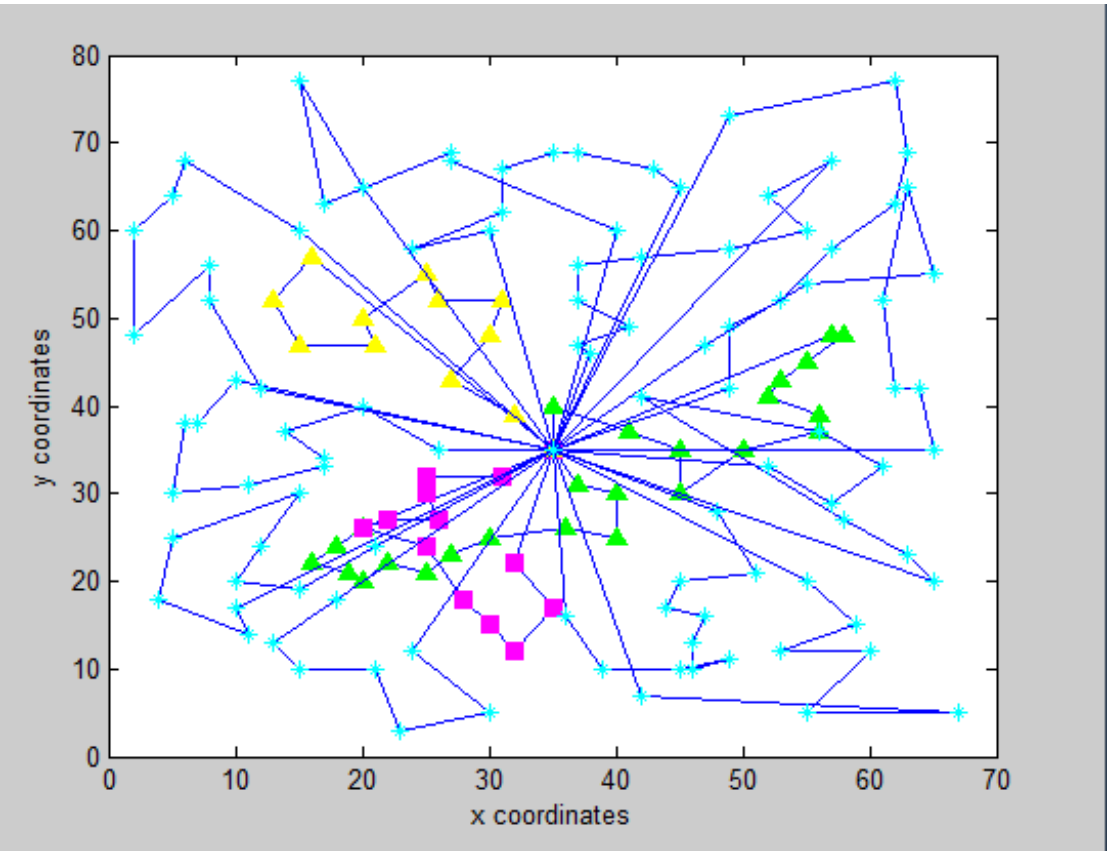
Γράφημα 5.17: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 9

Συνολικό κόστος: 1485,4

Διαδρομή:

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

Το μικρότερο κόστος για τη μέθοδο grasp εμφανίζεται στις 200 επαναλήψεις 1-1 exchange και $6^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.18: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο grasp για το παράδειγμα 9

Συνολικό κόστος: 1603,2

Διαδρομή:

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

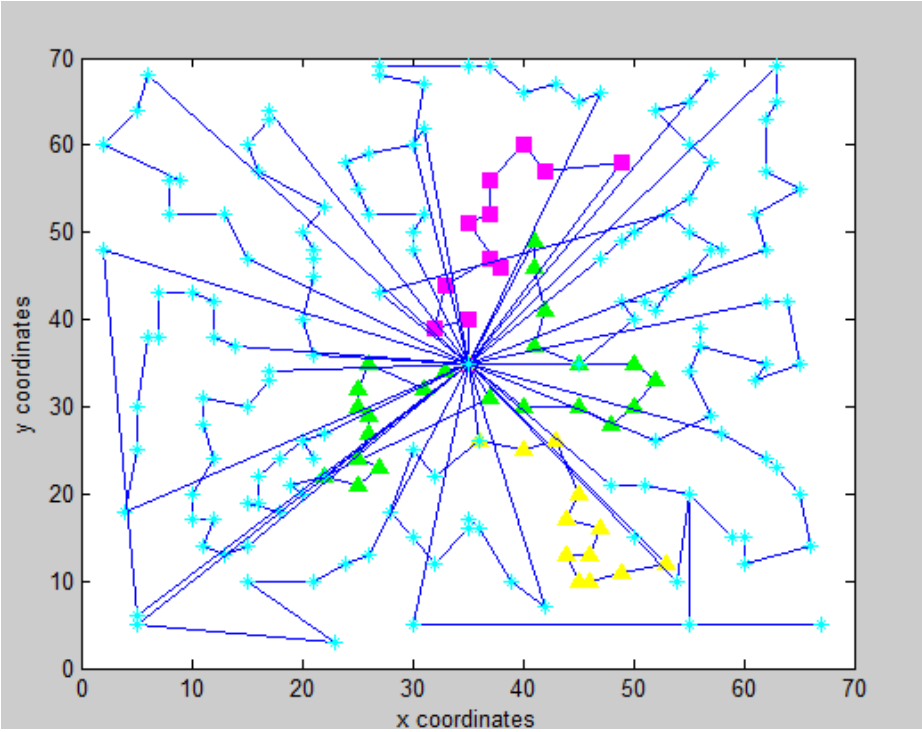
- **Παράδειγμα 10**
  αριθμός πελατών: 199
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 200
  χρόνος εξυπηρέτησης: 10

PARADEIGMA 10										
epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1872,2	1981,6	1872,2	1938,2	1872,2	1938,8	1872,2	2034,7	1872,2	2023,1
2	1872,2	1940,9	1872,2	2004,4	1872,2	1956,5	1872,2	1976,2	1872,2	1929,2
3	1872,2	1961,1	1872,2	1984,7	1872,2	1961,5	1872,2	1976,5	1858,6	1979,3
4	1872,2	1941,4	1872,2	2002,9	1872,2	1933,7	1872,2	1996,7	1872,2	1989,3
5	1872,2	1969,3	1858,6	1897,8	1872,2	2029,9	1872,2	1968,3	1872,2	1963
6	1872,2	2017,1	1872,2	1987,8	1872,2	1972,8	1872,2	1956,7	1872,2	1983,3
7	1872,2	1990	1872,2	1922,3	1872,2	1912,1	1872,2	2014,9	1872,2	1901,4
8	1872,2	1997,6	1872,2	1991,6	1872,2	1920,3	1872,2	1973,1	1872,2	1980,8
9	1872,2	1932,7	1872,2	2002,9	1872,2	2026,4	1872,2	2028,6	1872,2	1942,6
10	1872,2	2014,9	1872,2	2007,4	1872,2	2009,2	1872,2	1935	1872,2	1993,9
mesos oros	1872,2	1974,66	1870,84	1974	1872,2	1966,1	1872,2	1986,07	1870,84	1968,59

Πίνακας 5.10: Υπολογιστικά αποτελέσματα για το παράδειγμα 10

Στο παράδειγμα αυτό διαπιστώνουμε, ότι ο αλγόριθμος του πλησιέστερου γείτονα επιστρέφει καλύτερα αποτελέσματα από τη grasp. Συγκεκριμένα το χαμηλότερο μέσο κόστος παρουσιάζεται για τον πλησιέστερο γείτονα με τιμή 1870,84 στις 100 και 1000 επαναλήψεις 1-1 exchange ενώ η grasp στις 200 επαναλήψεις εμφανίζει το δικό της καλύτερο μέσο κόστος με τιμή 1966,1.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου του πλησιέστερου γείτονα, στις 100 επαναλήψεις 1-1 exchange και 5^{η} εφαρμογή του κώδικα.



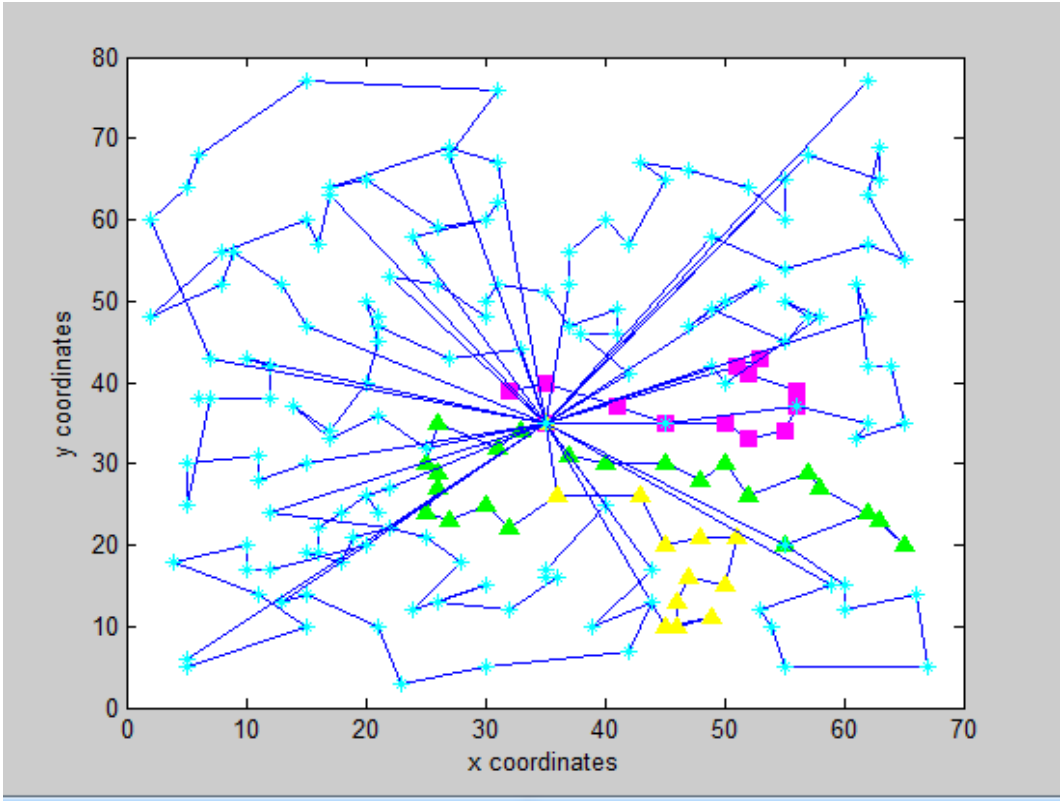
Γράφημα 5.19: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδειγμα 10

Συνολικό κόστος: 1858,6

Διαδρομή:

1 157 113 90 148 7 184 95 96 118 98 93 54 1 106 27 150 196 110 13 139
29 112 177 2 1 28 147 168 70 133 163 102 71 31 123 52 1 59 41 181 22
74 73 75 172 23 134 76 57 1 153 138 14 88 145 58 179 3 116 146 42 1
155 185 77 197 117 78 4 80 130 186 34 53 1 128 191 32 89 149 63 160 11
109 1 167 19 154 107 195 8 183 124 20 108 176 1 97 100 60 105 94 86 194
92 101 38 99 152 1 51 103 158 82 121 10 104 162 72 1 61 119 6 85 174
62 17 142 192 45 120 193 1 199 111 5 140 188 40 171 26 56 166 131 1 180
55 178 81 69 151 164 135 25 30 122 1 84 200 115 9 175 126 46 18 114 87
1 159 170 79 35 165 136 36 137 1 83 49 125 48 169 37 144 50 1 173 43
143 15 44 39 1 198 187 156 24 68 16 1 190 91 127 64 33 132 161 129 21
189 1 12 65 182 67 66 1 47 141 1

Το μικρότερο κόστος για τη μέθοδο grasp εμφανίζεται για τις 100 επαναλήψεις της 1-1 exchange και $5^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.20: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο grasp για το παράδειγμα 10

Συνολικό κόστος: 1897,8

Διαδρομή:

1 54 106 27 150 196 180 55 131 56 26 166 156 1 157 113 90 7 184 95 96
118 14 138 59 1 147 28 29 139 13 110 178 81 69 117 78 197 1 153 181 22
199 111 198 73 75 134 76 23 1 112 133 177 2 70 163 32 191 128 89 183 1
168 53 107 195 8 154 19 61 84 119 167 148 1 41 3 179 116 58 173 145 43
88 98 93 1 97 100 60 105 94 86 194 92 101 99 152 62 1 77 185 4 130
186 80 159 103 51 158 34 1 155 151 164 135 25 30 122 79 170 1 74 146 172
42 16 44 143 193 120 1 102 71 31 123 21 129 189 104 10 162 1 38 192 142
17 87 45 15 39 141 1 6 174 85 18 114 126 46 200 115 9 1 83 49 169 125
47 48 20 124 108 1 149 63 11 109 190 160 12 176 64 91 1 5 188 40 171
68 24 187 57 140 1 52 82 165 35 136 137 36 72 1 175 37 144 50 65 182
127 1 121 67 132 161 33 1 66 1

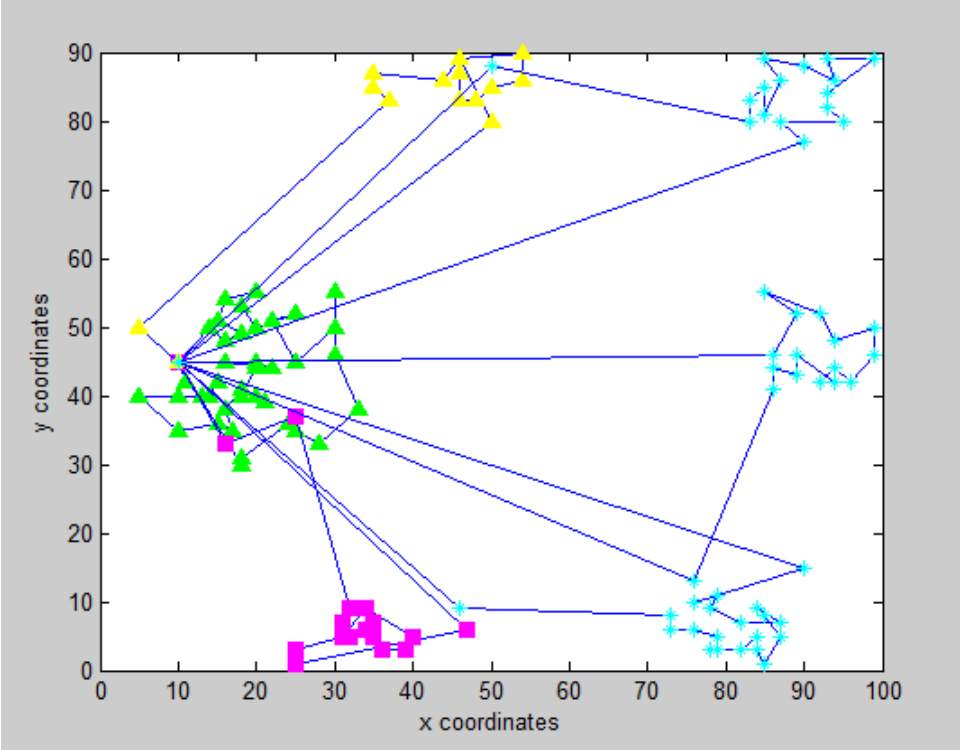
- **Παράδειγμα 11**
  αριθμός πελατών: 120
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος
  χρόνος εξυπηρέτησης: 0

PARADEIGMA 11										
epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1340,1	1300,5	1340,1	1294,6	1340,1	1310,6	1340,1	1314,7	1340,1	1218,6
2	1340,1	1349,2	1340,1	1290,1	1340,1	1327,6	1340,1	1346,2	1340,1	1317,6
3	1340,1	1342,2	1340,1	1348,7	1340,1	1312,4	1340,1	1317,7	1340,1	1319,9
4	1340,1	1369,7	1340,1	1299,7	1340,1	1288,6	1340,1	1337,4	1340,1	1347,9
5	1340,1	1336,6	1340,1	1275,7	1340,1	1336,8	1340,1	1361,9	1340,1	1284,3
6	1340,1	1355,7	1340,1	1286,1	1340,1	1370,8	1340,1	1331,9	1340,1	1352,4
7	1340,1	1372,1	1340,1	1361,1	1340,1	1335,2	1340,1	1329,4	1340,1	1353,7
8	1340,1	1377,5	1340,1	1273,6	1340,1	1302,8	1340,1	1349,8	1340,1	1336,7
9	1340,1	1315,5	1340,1	1294,2	1340,1	1308,4	1340,1	1310,3	1340,1	1350,5
10	1340,1	1322,2	1340,1	1305,1	1340,1	1297,5	1340,1	1322,3	1340,1	1299
mesos oros	1340,1	1344,12	1340,1	1302,89	1340,1	1319,1	1340,1	1332,16	1340,1	1318,06

Πίνακας 5.11: Υπολογιστικά αποτελέσματα για το παράδειγμα 11

Για το παράδειγμα 11 παρατηρούμε ότι τα αποτελέσματα που επιστρέφει η grasp, συμφέρουν στη πλειοψηφία τους περισσότερο από αυτά του πλησιέστερου γείτονα. Όσων αφορά τα μέσα κόστη, το χαμηλότερο εμφανίζεται για τη grasp στις 100 επαναλήψεις 1-1 exchange με τιμή 1302,89, ενώ ο πλησιέστερος γείτονας φαίνεται να διατηρεί σε όλες τις επαναλήψεις την αρχική λύση της μεθόδου με τιμή 1340,1.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 1000 επαναλήψεις 1-1 exchange και 1η εφαρμογή του κώδικα.



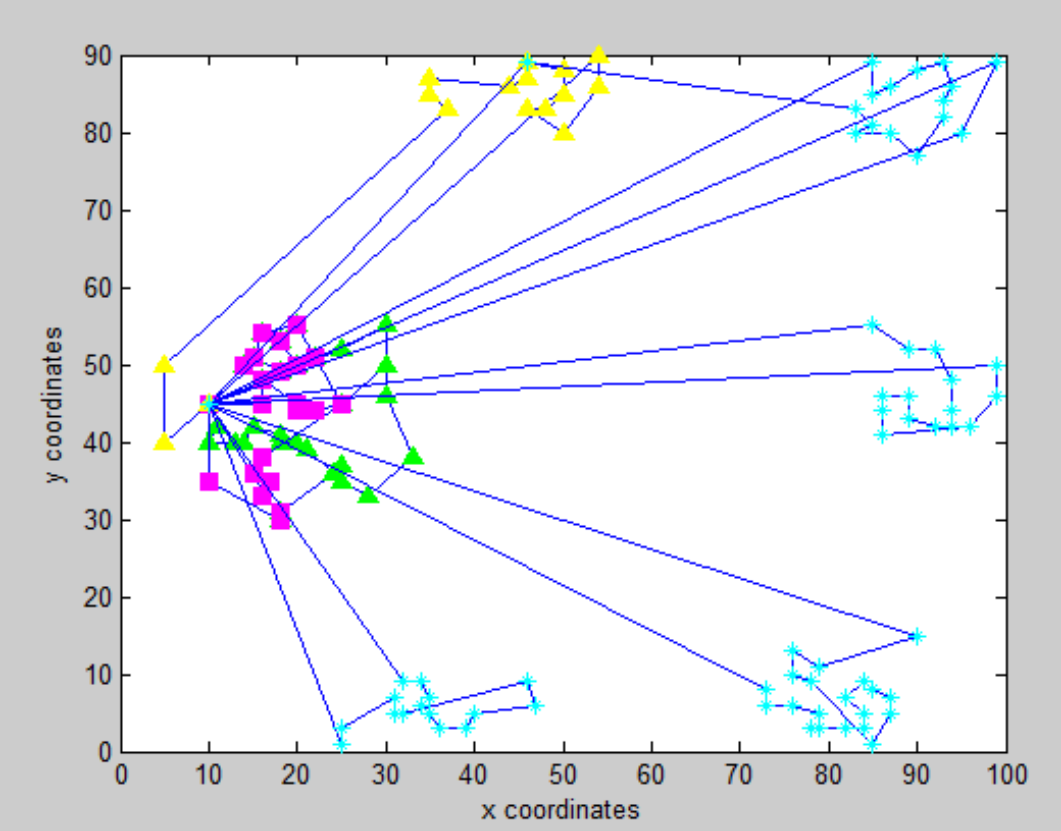
Γράφημα 5.21: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδειγμα 11

Συνολικό κόστος: 1218,6

Διαδρομή:

1 89 112 88 87 83 120 82 113 85 86 93 92 91 90 94 97 95 96 1 106 103
102 107 108 104 105 100 117 101 98 111 99 116 110 109 119 19 114 84 1 118
115 7 6 5 11 12 10 16 15 14 8 4 3 2 13 1 121 68 70 71 72 73 75
74 77 78 80 81 76 69 1 79 53 54 56 55 59 57 61 65 64 67 63 62 66 58
60 1 9 18 17 20 26 25 23 28 32 31 34 35 36 33 37 29 24 27 21 30 1
22 38 39 42 43 47 45 48 50 51 52 49 46 41 44 40 1

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα ισούται με την αρχική του λύση.



Γράφημα 5.22: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 11

Συνολικό κόστος: 1340,1

Διαδρομή:

1 89 83 112 87 88 93 90 92 91 19 115 119 109 110 116 111 99 117 1 96
103 102 100 101 104 105 108 107 106 97 94 95 98 86 113 85 118 114 84 82 1
120 121 68 70 71 72 75 73 79 78 77 74 69 80 81 1 7 8 10 11 12 16
15 14 13 9 5 4 6 3 2 1 76 54 55 53 58 60 62 63 65 64 61 59 56
57 1 18 17 20 26 25 23 28 31 32 29 33 36 37 35 34 24 21 22 27 30 1
41 44 46 49 48 47 45 42 43 40 39 38 50 51 52 1 66 67 1

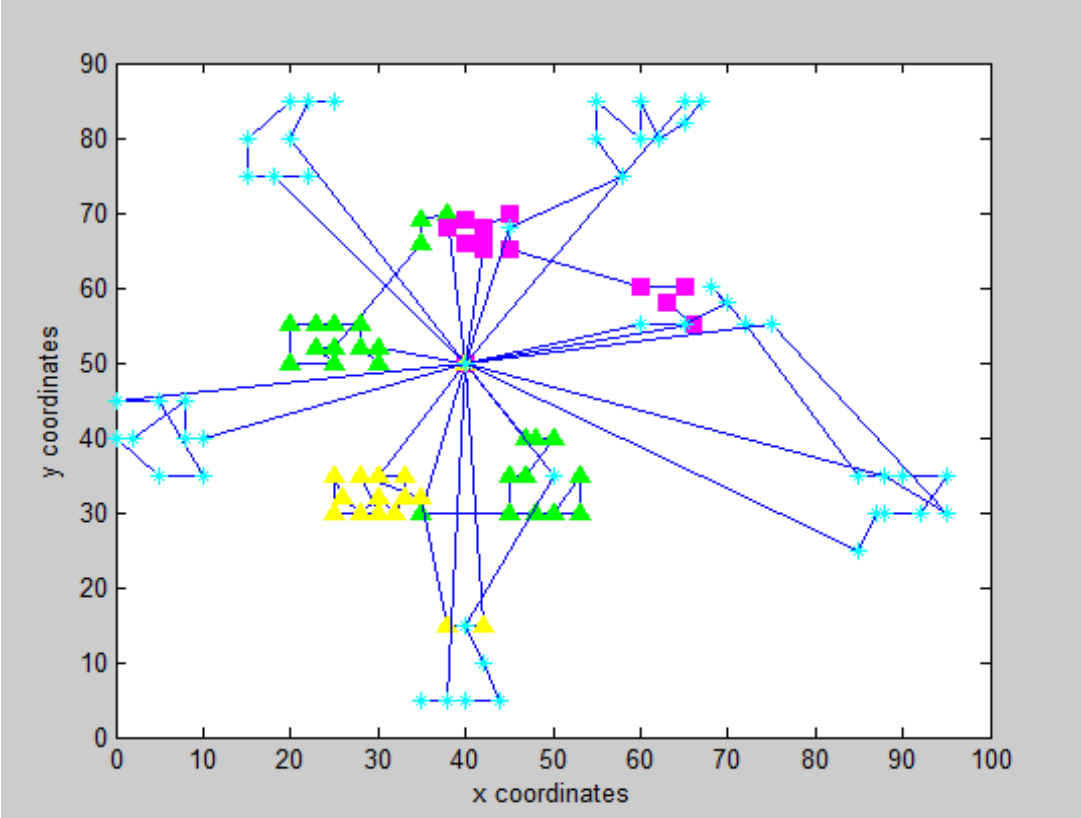
- **Παράδειγμα 12**
  αριθμός πελατών: 100
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: απεριόριστος
  χρόνος εξυπηρέτησης: 0

PARADEIGMA 12										
epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1146,4	1184,4	1146,4	1172,8	1146,4	1190,4	1146,4	1180,2	1146,4	1148,7
2	1146,4	1211,3	1146,4	1100	1146,4	1229,5	1146,4	1199,5	1146,4	1163,7
3	1146,4	1185	1146,4	1209,6	1146,4	1217,5	1146,4	1166,6	1146,4	1206,7
4	1146,4	1190,8	1146,4	1111,5	1146,4	1172,4	1146,4	1112,7	1146,4	1085,9
5	1146,4	1188,5	1146,4	1165,4	1146,4	1196,7	1146,4	1179,2	1146,4	1170,2
6	1146,4	1186,7	1146,4	1235,4	1146,4	1191,4	1146,4	1189,5	1146,4	1198,5
7	1146,4	1198,6	1146,4	1181,7	1146,4	1165,1	1146,4	1182	1146,4	1145,4
8	1146,4	1177,6	1146,4	1117,1	1146,4	1205,7	1146,4	1187,9	1146,4	1125,8
9	1146,4	1126,1	1146,4	1201,6	1146,4	1112,9	1146,4	1189,3	1146,4	1182,3
10	1146,4	1209,4	1146,4	1117,9	1146,4	1179,3	1146,4	1161,7	1146,4	1172,8
mesos oros	1146,4	1185,84	1146,4	1161,3	1146,4	1186,1	1146,4	1174,86	1146,4	1160

Πίνακας 5.12: Υπολογιστικά αποτελέσματα για το παράδειγμα 12

Στο συγκεκριμένο παράδειγμα, το καλύτερο μέσο κόστος είναι ίσο με την αρχική λύση του πλησιέστερου γείτονα με τιμή 1146,4 ενώ για τη μέθοδο grasp το καλύτερο μέσο κόστος εμφανίζεται στις 1000 επαναλήψεις 1-1 exchange με τιμή 1160. Παρόλα αυτά παρατηρείται πως με τη μέθοδο grasp έχει επιτευχθεί αρκετές φορές κόστος διαδρομής μικρότερο από την αρχική λύση του πλησιέστερου γείτονα που προαναφέρθηκε όπως για παράδειγμα στην 4η εφαρμογή του κώδικα για 100 επαναλήψεις 1-1 exchange με τιμή 1111,5.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 1000 επαναλήψεις 1-1 exchange και 4^{η} εφαρμογή του κώδικα.



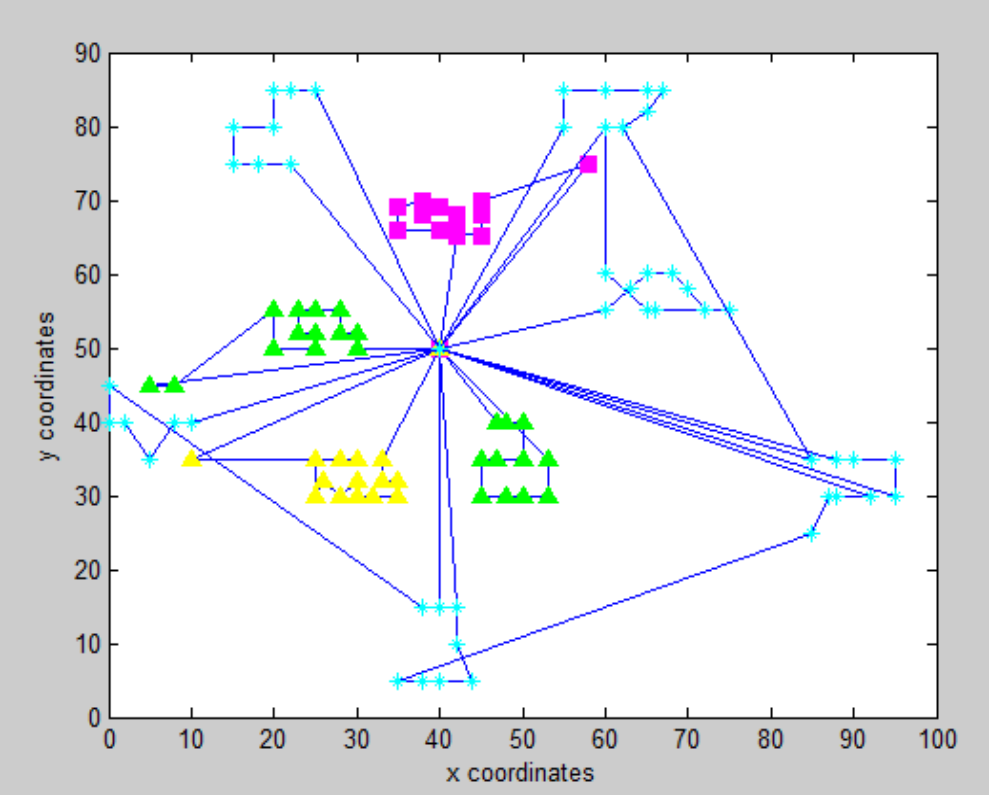
Γράφημα 5.23: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδείγμα 12

Συνολικό κόστος: 1085,9

Διαδρομή:

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

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα ισούται με την αρχική του λύση.



Γράφημα 5.24: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 12

Συνολικό κόστος: 1146,4

Διαδρομή:

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

- **Παράδειγμα 13**
  αριθμός πελατών: 120
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 720
  χρόνος εξυπηρέτησης: 50

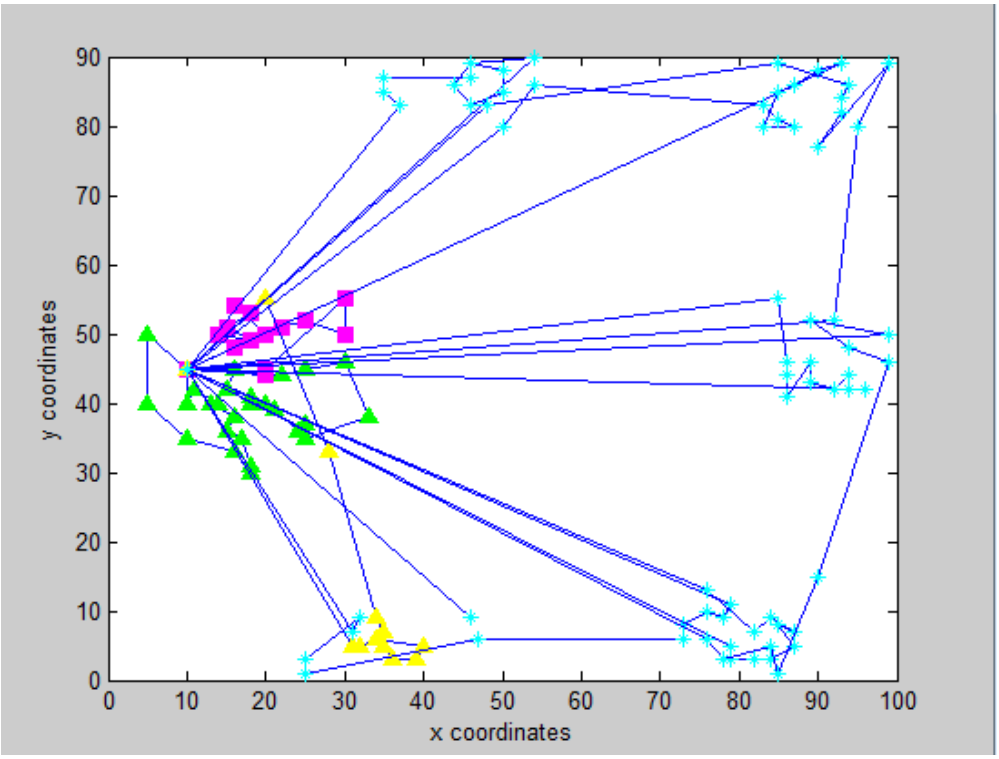
PARADEIGMA 13

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1913,7	1908,4	1912,7	1914,9	1912	1942,7	1907,8	1881,4	1911,1	1886,6
2	1913,4	1932,8	1913,7	1909,7	1911,2	1905,4	1912,1	1883,9	1912,2	1914,8
3	1913,7	1871,5	1913,7	1920,8	1908,7	1906,7	1911,9	1880,5	1909,7	1908,8
4	1913,7	1914	1913,7	1895,3	1913,7	1917,1	1909,7	1913,9	1902,2	1880,6
5	1913,7	1896,1	1913,7	1925,3	1913,6	1891,7	1899,4	1893,3	1908	1909,6
6	1909,7	1897,8	1913,7	1895,5	1913,7	1903,4	1913	1887,4	1909,7	1878,7
7	1913,7	1929,8	1913,7	1895,1	1911,1	1908,8	1912,1	1900,1	1899,4	1883,1
8	1913,7	1903,6	1913,7	1903,1	1908	1871,6	1908	1903,6	1907,8	1909,2
9	1913,7	1942,9	1913,3	1923,5	1912,6	1910,6	1912,7	1908,6	1907,8	1898,8
10	1913,7	1924,7	1912,1	1851,7	1913,7	1911,5	1910,6	1892,2	1910,6	1889,8
mesos oros	1913,27	1912,16	1913,4	1903,49	1911,83	1907	1909,73	1894,49	1907,85	1896

Πίνακας 5.13: Υπολογιστικά αποτελέσματα για το παράδειγμα 13

Από το παραπάνω πίνακα γίνεται αντιληπτό ότι στο συγκεκριμένο παράδειγμα η grasp αποδίδει καλύτερα. Συγκεκριμένα, το καλύτερο μέσο κόστος εμφανίζεται στις 500 επαναλήψεις 1-1 exchange της μεθόδου grasp με τιμή 1894,49 ενώ το καλύτερο μέσο κόστος του πλησιέστερου γείτονα ισούται με 1907,85 και παρουσιάζεται στις 1000 επαναλήψεις 1-1 exchange.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 100 επαναλήψεις 1-1 exchange και $10^{η}$ εφαρμογή του κώδικα.



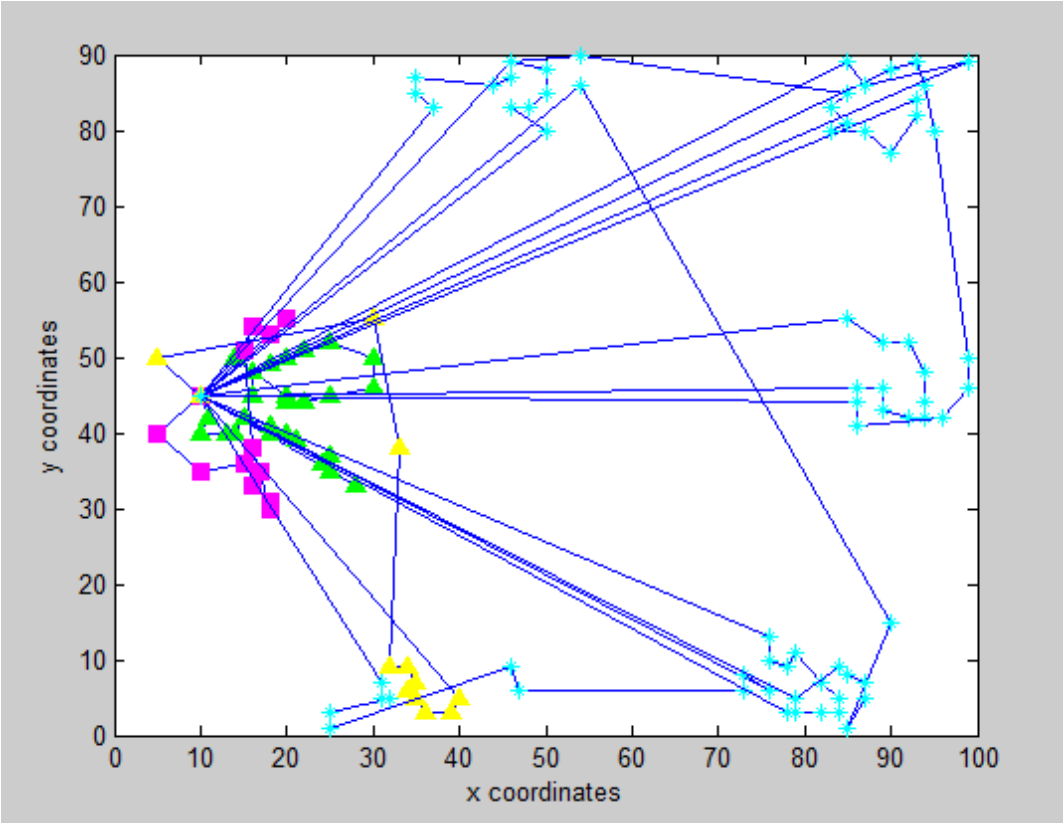
Γράφημα 5.25: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδειγμα 13

Συνολικό κόστος: 1851,7

Διαδρομή:

1 83 89 87 112 86 113 85 114 84 118 82 120 121 1 96 88 93 92 90 91 19
115 119 110 116 98 95 1 103 106 102 107 105 108 100 101 117 111 99 97 94
1 104 109 8 10 12 16 15 14 11 5 4 1 6 7 3 2 13 17 18 21 24 27 1
68 70 71 75 73 72 74 78 79 76 81 1 69 80 54 55 58 53 56 59 61 64 1
77 57 65 63 62 60 67 66 46 1 22 29 33 37 36 35 31 28 25 26 1 41 40
39 38 43 42 45 48 47 50 1 20 23 32 34 30 51 49 44 52 1 9 1

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα παρουσιάζεται στις 1000 επαναλήψεις 1-1 exchange και $7^{η}$ εφαρμογή του κώδικα.



Γράφημα 5.26: Γραφική αναπαράσταση καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 13

Συνολικό κόστος: 1899,4

Διαδρομή:

1 89 83 112 87 88 93 90 92 91 19 115 119 109 1 96 103 102 100 101 117
111 116 98 95 94 97 106 1 120 82 113 85 118 114 84 86 107 108 105 104 1
121 99 110 7 8 10 11 12 16 15 14 1 6 4 5 3 2 9 13 17 18 20 1
68 70 71 72 75 73 79 78 77 74 69 1 76 81 56 54 55 53 58 60 62 63 1
22 21 24 27 29 32 31 28 25 23 1 41 44 46 49 48 47 45 42 43 40 1 39
38 50 51 52 66 65 64 61 1 26 33 36 37 35 34 30 80 1 57 59 67 1

- **Παράδειγμα 14**
  αριθμός πελατών: 100
  χωρητικότητα οχήματος: 200
  μέγιστος επιτρεπτός χρόνος των οχημάτων στην εξυπηρέτηση: 1040
  χρόνος εξυπηρέτησης: 90

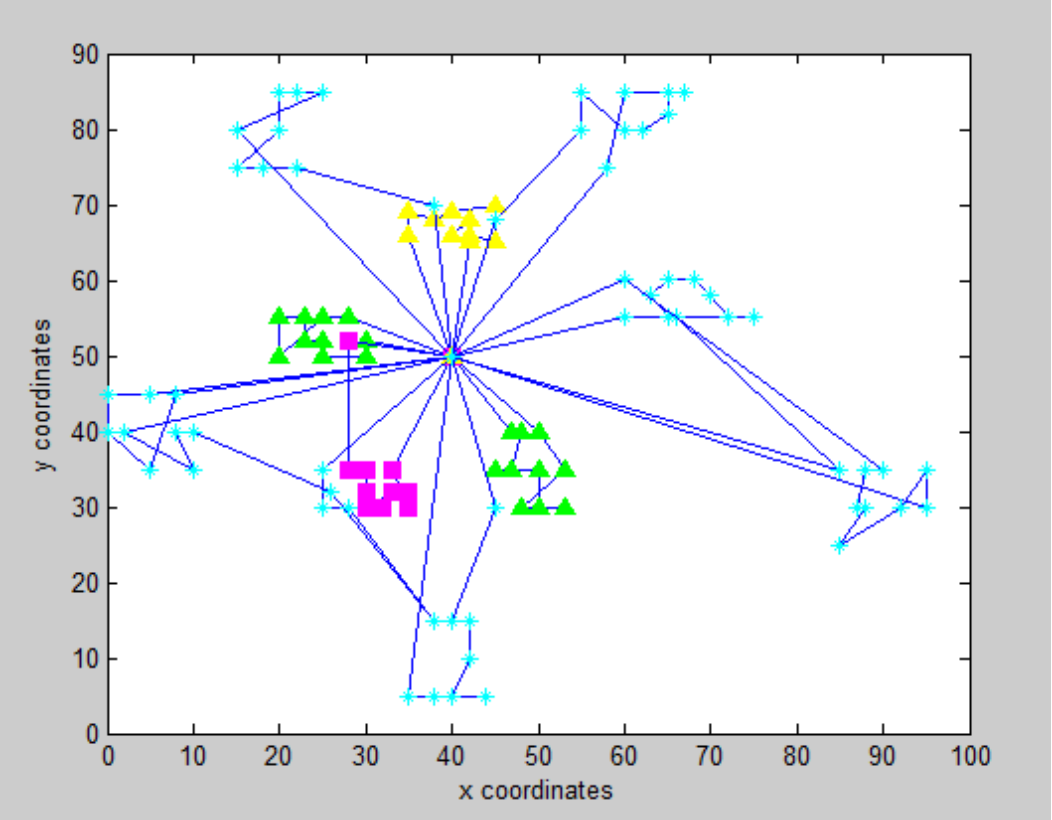
PARADEIGMA 14

epanalipseis 1-1 exchange	50		100		200		500		1000	
methodos	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp	plisiesteros	grasp
1	1134,8	1163,4	1134,8	1164,5	1134,8	1181,2	1134,8	1147,7	1134,8	1146,9
2	1134,8	1158,4	1134,8	1179,1	1134,8	1154,8	1134,8	1168,2	1134,8	1096,1
3	1134,8	1155,8	1134,8	1186,2	1134,8	1119	1134,8	1162,8	1134,8	1153,1
4	1134,8	1154,3	1134,8	1128,5	1134,8	1147,1	1134,8	1165,8	1134,8	1169
5	1134,8	1184,2	1134,8	1122,5	1134,8	1176,4	1134,8	1140,8	1134,8	1150
6	1134,8	1172,8	1134,8	1162,7	1134,8	1115,8	1134,8	1156,6	1134,8	1169,2
7	1134,8	1164	1134,8	1155,5	1134,8	1157,1	1134,8	1146,8	1134,8	1178,2
8	1134,8	1148,8	1134,8	1153,3	1134,8	1146,4	1134,8	1167,7	1134,8	1154
9	1134,8	1162,8	1134,8	1176,5	1134,8	1155,7	1134,8	1168	1134,8	1187,8
10	1134,8	1163,6	1134,8	1154,9	1134,8	1154,8	1134,8	1143,2	1134,8	1131,1
mesos oros	1134,8	1162,81	1134,8	1158,37	1134,8	1150,8	1134,8	1156,76	1134,8	1153,54

Πίνακας 5.14: Υπολογιστικά αποτελέσματα για το παράδειγμα 14

Στο τελευταίο παράδειγμα, παρατηρείται ότι η αρχική λύση του πλησιέστερου γείτονα είναι αυτή που εμφανίζει τη καλύτερη τιμή 1134,8. Παρόλα αυτά σε ορισμένες επαναλήψεις της grasp παρουσιάζονται μικρότερες τιμές, όμως οι τελικοί μέση όροι είναι χειρότεροι από το 1134,8 που προαναφέρθηκε με τον μικρότερο από αυτούς να λαμβάνει τη τιμή 1150,8 στις 200 επαναλήψεις 1-1 exchange.

Το μικρότερο κόστος, και επομένως η καλύτερη διαδρομή, παρουσιάζεται με εφαρμογή της μεθόδου grasp, στις 1000 επαναλήψεις 1-1 exchange και $2^{η}$ εφαρμογή του κώδικα.



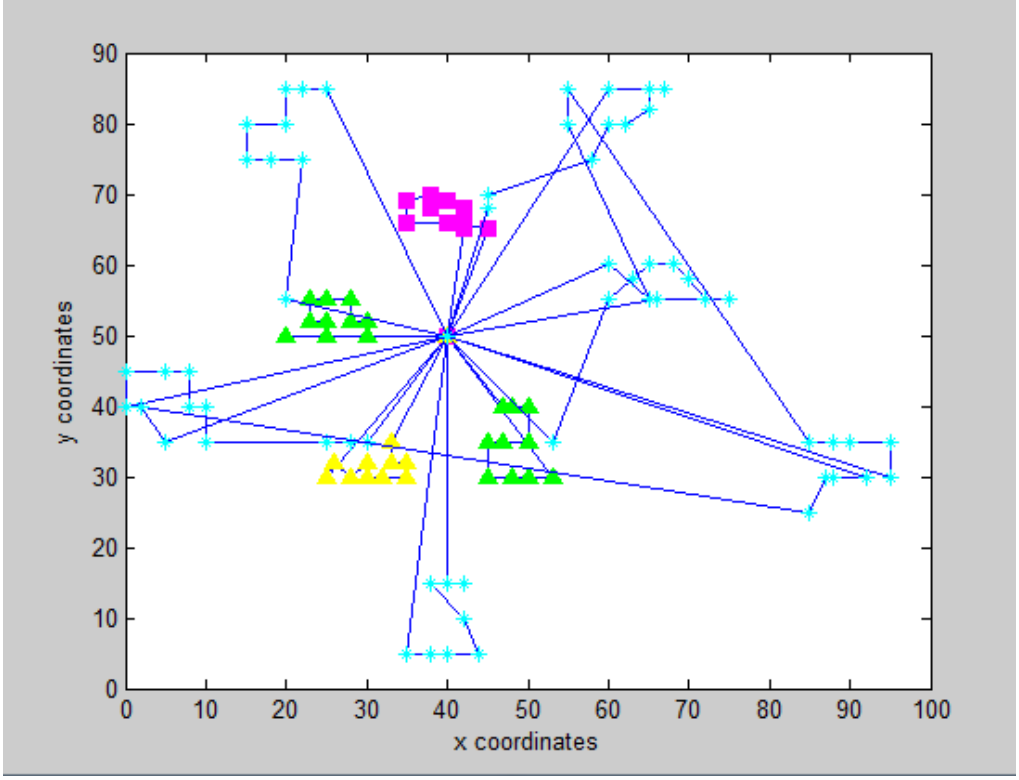
Γράφημα 5.27: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας για το παράδειγμα 14

Συνολικό κόστος: 1096,1

Διαδρομή:

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

Το μικρότερο κόστος για τη μέθοδο του πλησιέστερου γείτονα ισούται με την αρχική του λύση.



Γράφημα 5.28: Γραφική αναπαράσταση της καλύτερης λύσης που βρήκε ο κώδικας με τη μέθοδο του πλησιέστερου γείτονα για το παράδειγμα 14

Συνολικό κόστος: 1134,8

Διαδρομή:

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

Κεφάλαιο 6: Παρατηρήσεις - Συμπεράσματα

Παρατηρείται ότι στα μικρά προβλήματα 50 κόμβων (1 και 6), η grasp αποδίδει αρκετά καλύτερα από τη μέθοδο του πλησιέστερου γείτονα. Ακόμη η συχνότητα με την οποία εμφανίζονται πιο συμφέρουσες λύσεις είναι μεγάλη, αφού στο παράδειγμα 6 όλες οι λύσεις που παράχθηκαν με τη μέθοδο grasp παρουσίαζαν μικρότερο κόστος διαδρομής σε σχέση με αυτές του πλησιέστερου γείτονα. Επίσης, και για το παράδειγμα 1 ισχύει ότι κατά μεγάλη πλειοψηφία οι λύσεις της grasp συνέφεραν περισσότερο. Παρόλα αυτά, παρατηρείται ότι με αύξηση των πελατών ο πλησιέστερος γείτονας επιστρέφει κατά μέσο όρο λύσεις με χαμηλότερο κόστος από αυτές που επιστρέφει η μέθοδος grasp. Όμως υπάρχουν και παραδείγματα με μεγάλο πλήθος κόμβων (παραδείγματα 8, 11, και 13) για τα οποία η grasp αποδίδει κατά μέσο όρο καλύτερα. Ακόμη στα παραδείγματα 12 και 14, παρόλο που στη πλειοψηφία τους οι μικρότερες λύσεις φαίνεται να προκύπτουν με τη μέθοδο του πλησιέστερου γείτονα, η καλύτερη λύση εμφανίζεται με χρήση της μεθόδου grasp.

Όσων αφορά την αύξηση του αριθμού εφαρμογών της μεθόδου 1-1 exchange στις αρχικές λύσεις τόσο του πλησιέστερου γείτονα όσο και της grasp, δεν φαίνεται να επηρεάζει ιδιαίτερα τη ποιότητα των λύσεων. Συγκεκριμένα στη περίπτωση του πλησιέστερου γείτονα, παρατηρείται πολλές φορές ότι η αρχική λύση παραμένει βέλτιστη και μετά την εφαρμογή της 1-1 exchange. Παρόλα αυτά σε ορισμένες περιπτώσεις όπως στο παράδειγμα 6, με αύξηση των επαναλήψεων 1-1 exchange η λύση ήταν πιο πιθανό να βελτιωθεί.

Ακόμα αξίζει να σημειωθεί ότι κατά τη διάρκεια εκπόνησης της εργασίας είχε παρατηρηθεί πως η αύξηση του μεγέθους της λίστας υποψηφίων της μεθόδου grasp επέφερε λύσεις με αυξημένο κόστος διαδρομής. Επιπλέον παρατηρήθηκε ότι ο αλγόριθμος grasp λειτουργεί καλύτερα όταν αυξάνουμε τις επαναλήψεις του.

Από τα παραπάνω προκύπτουν τα εξής συμπεράσματα:

- Ο αλγόριθμος grasp λειτουργεί καλύτερα για παραδείγματα με μικρό αριθμό κόμβων χωρίς όμως να αποκλείεται η εύρεση καλών αποτελεσμάτων και σε μεγαλύτερα παραδείγματα.
- Η αύξηση των επαναλήψεων τοπικής αναζήτησης με τη μέθοδο 1-1 exchange δεν επηρεάζει ιδιαίτερα τη παραγόμενη λύση.
- Μεγάλο πλήθος επαναλήψεων του αλγόριθμου grasp και επομένως αρχικών τυχαίων λύσεων αυξάνει τη πιθανότητα εύρεσης καλής τελικής λύσης.
- Η αύξηση του μεγέθους της λίστας υποψηφίων και κατ' επέκταση της τυχαιότητας με την οποία παράγονται οι διαδρομές οδηγεί σε χειρότερες λύσεις.

Για τη βελτίωση της ποιότητας των παραγόμενων λύσεων με τη μέθοδο grasp, κάποιες ενέργειες που θα μπορούσαν να πραγματοποιηθούν είναι:

- Αύξηση των επαναλήψεων κατασκευής αρχικών λύσεων.
- Χρήση διαφορετικού αλγόριθμου τοπικής αναζήτησης όπως για παράδειγμα 2-opt ή 3-opt.
- Μετά την εφαρμογή της πρώτης τοπικής αναζήτησης στην αρχική λύση, χρήση επιπλέον μεθόδων τοπικής αναζήτησης για περεταίρω βελτίωσή της.

βιβλιογραφία

[1] Beamon, B.M. (1998) Supply chain design and analysis: Models and methods, *International Journal of Production Economics,* 55(3), 281–294.

[2] Christofides, N., Mingozzi, A. , Toth, P. (1979) The vehicle routing problem, in: N. Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (Editors), *Combined optimization*, Wiley, Chichester, 315–338.

[3] Clarke, G., Wright, J.V. (1964) Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research,* 12 (4), 568–581.

[4] Cornuejols, G., Harche, F. (1993) Polyhedral study of the capacitated vehicle routing problem, *Mathematical Programming,* 60(1), 21-52.

[5] Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., Vogel, U. (2011) Vehicle routing with compartments: applications, modelling and heuristics, *OR Spectrum*, 33(4), 885-914.

[6] El Fallahi, A., Prins′ C., Calvo, R.W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem, *Computers & Operations Research*, 35(5), 1725-1741.

[7] Feo, T.A., Resende, M. G.C. (1995) Greedy Randomized Adaptive Search Procedures, *Journal of Global Optimization*, 6(2), 109-133.

[8] Fisher, M.L., Jaikumar, R. (1981) A generalized assignment heuristic for the vehicle routing problem, *Networks,* 11(2), 109–124.

[9] Fisher, M. L., Jaikumar, R., Van Wassenhove, L.N. (1986) A Multiplier Adjustment Method for the Generalized Assignment Problem, *Management Science*, 32(9), 1095-1103.

[10] Gillett, B.E., Miller, L.R. (1974) A heuristic algorithm for the vehicle dispatch problem, *Operations Research,* 22(2) ,340–349.

[11] Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I-M. (1998) Metaheuristics in vehicle routing, in: T. Crainic, G., Laporte, G. (Editors), *Fleet management and logistics*, Kluwer, Boston, 33–56.

[12] Henke, T., Speranza, M.G., Wäscher, G. (2015) The multi-compartment vehicle routing problem with flexible compartment sizes, *European Journal of Operational Research*, 246(3), 730-743.

[13] Laporte, G. (2007) What You Should Know about the Vehicle Routing Problem, *Naval Research Logistics*, 54(8), 811-819.

[14] Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G.R., Shmoys, D.B. (1985), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* Wiley and Sons.

[15] Pillac , V., Gendreau M., Guéret, C., Medaglia, A.L. (2013) A review of dynamic vehicle routing problems, *European Journal of Operational Research*, 225(1), 1-11.

[16] Prins, C. (2004) A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers & Operations Research,* 31(12), 1985–2002.

[17] Rochat, Y., Taillard, É.D. (1995) Probabilistic diversification and intensification in local search for vehicle routing, *Journal of Heuristics*, 1(1), 147–167.

[18] Toth, P., Vigo, D. (2002), *The vehicle routing problem*, SIAM.

[19] Voudouris, C., Tsang, E. (1999) Guided local search and its application to the traveling salesman problem, *European Journal of Operational Research*, 113(2), 469-499.

[20]Waters, C.D.J. (1987) A solution Procedure for the Vehicle Scheduling Problem Based on Iterative Route Improvement, *The Journal of the Operational Research Society*, 38(9), 833-839.

[21] Μαρινάκης, Ι., Μυγδαλάς, Α. (2008), Σχεδιασμός και βελτιστοποίηση της εφοδιαστικής αλυσίδας, Εκδόσεις σοφία.

[22]http://s3.amazonaws.com/academia.edu.documents/30458400/greedy_algorithms.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1444310843&Signature=MliMzA6v6f5yf68ptojxz5G4Hbs%3D&response-content-disposition=inline

[23] http://www.scholarpedia.org/article/Metaheuristic_Optimization