ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλγόριθμος άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης για το πρόβλημα δρομολόγησης οχημάτων σε περιορισμένη απόσταση

(Greedy randomized adaptive search procedure for the distance-constrained vehicle routing problem)

Νικολακάκη Στέλλα

Επιβλέπων : Δρ. Ιωάννης Μαρινάκης, Επίκουρος Καθηγητής

Χανιά 2015

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Ιωάννη Μαρινάκη για την καθοδήγηση και την υποστήριξη που μου έδωσε όλο αυτό το χρονικό διάστημα, τον βοηθό του κ. Μαρινάκη, Ηρακλή-Δημήτριο Ψύχα για την πολύτιμη βοήθειά του, όπως επίσης και την κα. Μάγδα Μαρινάκη.

Ακόμα, ένα μεγάλο ευχαριστώ στην οικογένειά μου και στους φίλους μου οι οποίοι αποτέλεσαν το μεγαλύτερο στήριγμα για εμένα όλα αυτά τα χρόνια.

Περίληψη

Η συγκεκριμένη διπλωματική έχει σαν αντικείμενο το πρόβλημα δρομολόγησης οχημάτων σε περιορισμένη απόσταση. Στόχος, είναι η αποδοτική επίλυση του προβλήματος για τη μείωση της ευκλείδειας απόστασης που τα οχήματα της εφοδιαστικής αλυσίδας καλούνται να διανύσουν. Για την περιγραφή του προβλήματος ορίζονται κατάλληλα οι απαραίτητοι περιορισμοί για τον αριθμό και τις τοποθεσίες των πελατών, τον συνολικό χρόνο που έχει τη δυνατότητα να δαπανήσει το φορτηγό στο δρόμο, καθώς και τον χρόνο εξυπηρέτησης του κάθε πελάτη. Στην παρούσα εργασία χρησιμοποιείται και υλοποιείται η Διαδικασία της Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure (GRASP)). Στο πρώτο μέρος της επίλυσης, χρησιμοποιείται ένας απλός αλγόριθμος απληστίας, ο οποίος τυχαιοποιείται κατάλληλα για την εύρεση μιας αρχικής εφικτής λύσης που ικανοποιεί τους περιορισμούς, και στο δεύτερο μέρος εφαρμόζεται ένας αλγόριθμος τοπικής αναζήτησης 1-1 ανταλλαγή (1-1 exchange). Στην εργασία παρουσιάζεται η υλοποίηση της μεθόδου, καθώς και τα αποτελέσματα από τη χρήση των αλγορίθμων. Για την ανάπτυξη του αλγορίθμου χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον την MATLAB.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1Εφοδιαστική Αλυσίδα (Supply Chain)

Οι αυξανόμενες ανταγωνιστικές πιέσεις καθώς και η παγκοσμιοποίηση της αγοράς καθιστούν επιτακτική την ανάγκη εφαρμογής του management. Κάθε επιχείρηση εκτός από το να καλύπτει τις απαιτήσεις και τις ανάγκες των πελατών της, έχει ως στόχο την αύξηση του μεριδίου της στην αγορά καθώς και την εξασφάλιση της κερδοφορίας της. Η Εφοδιαστική Αλυσίδα (Supply Chain) έρχεται να καλύψει τις ανάγκες αυτές, καθώς αποτελεί βασικό συντελεστή για την εφαρμογή του management.

Μία από τις πιο σημαντικές αλλαγές της σύγχρονης διαχείρισης των επιχειρήσεων είναι ότι οι μεμονωμένες επιχειρήσεις δεν ανταγωνίζονται ως αποκλειστικά αυτόνομες οντότητες, αλλά ως αλυσίδες εφοδιασμού. Με τον όρο εφοδιαστική αλυσίδα, εννοούμε όλες τις δραστηριότητες που εκτελούνται από μια επιχείρηση με στόχο την ικανοποίηση των πελατών της. Η εφοδιαστική αλυσίδα αναφέρεται δηλαδή στην ροή, όχι μόνο προϊόντων αλλά και πληροφοριών από τον προμηθευτή μέχρι τον τελικό καταναλωτή. Ουσιαστικά αποτελείται από τους κατασκευαστές και προμηθευτές, τα κέντρα διανομών, τους μεταφορείς, τους πελάτες, τις πρώτες ύλες, τα έτοιμα προϊόντα κλπ. Όλα τα στάδια που εμπλέκονται στην αλυσίδα έχουν ως στόχο να διευθύνουν, να ελέγξουν και να βελτιώσουν τη ροή αυτή. Με τον τρόπο αυτό επιτυγχάνεται όχι μόνο η μείωση των συνολικών δαπανών αλλά και η ικανοποίηση των πελατών, η οποία με τη σειρά της οδηγεί στην αύξηση της κερδοφορίας της επιχείρησης.

Τα τρία βασικά στάδια της εφοδιαστικής αλυσίδας είναι οι **προμήθειες**, η **παραγωγή** και η **διανομή**. Κάθε στάδιο της αλυσίδας εκτελεί συγκεκριμένες διαδικασίες μέσα και έξω από τα εταιρικά όρια, αλλά ταυτόχρονα αλληλεπιδρά με τα υπόλοιπα στάδια της αλυσίδας. Οι βασικές διαδικασίες της εφοδιαστικής αλυσίδας είναι:

Η διαχείριση των πελατειακών σχέσεων

Η διαχείριση εξυπηρέτησης πελατών

Η διαχείριση της ζήτησης

Η εκτέλεση παραγγελιών

Η διαχείριση των προμηθειών

Η ανάπτυξη προϊόντων

Οι Επιστροφές

Στο Σχήμα 1.1 παρουσιάζεται η γενική δομή ενός δικτύου της εφοδιαστικής αλυσίδας.

Σχήμα 1.1 Δομή δικτύου εφοδιαστικής αλυσίδας.

1.2 Εφοδιαστική (Logistics)

Τα Logistics έχουν διαδραματίσει έναν θεμελιώδη ρόλο στην παγκόσμια ανάπτυξη σχεδόν 5000 χρόνια τώρα. Σύμφωνα με ιστορικές αναφορές τα Logistics ή όπως έχει μεταφραστεί στα ελληνικά Εφοδιαστική έκαναν την εμφάνισή τους από την εποχή κατασκευής των πυραμίδων στην αρχαία Αίγυπτο, όπου ήταν απαραίτητος ο κατάλληλος σχεδιασμός για την μεταφορά και την αποθήκευση των υλικών που χρησιμοποιούνταν. Με την σημερινή έννοια βέβαια η πρώτη εμφάνιση τους θα μπορούσε να θεωρηθεί κατά τα ρωμαϊκά χρόνια όταν για τον εφοδιασμό των λεγεώνων ή ύπαρξη ενός εξελιγμένου οδικού συστήματος βελτίωνε τη μεταφορά πολεμοφοδίων, υλικών και ανδρών.

Το Council of Logistics Management έχει ορίσει τη διαχείριση της εφοδιαστικής ως «τη διαδικασία προγραμματισμού, υλοποίησης και ελέγχου για εφικτή και αποτελεσματική ροή και αποθήκευση αγαθών υπηρεσιών και συναφών πληροφοριών από την πηγή στο σημείο κατανάλωσης με σκοπό τη συμμόρφωση με τις απαιτήσεις του πελάτη».

Ένας άλλος ορισμός των Logistics είναι η διαχείριση της ροής των αγαθών μεταξύ του σημείου προέλευσης και του σημείου της κατανάλωσης, έτσι ώστε να ανταποκριθούν οι επιχειρήσεις στις απαιτήσεις των πελατών τους. Τα αγαθά αυτά μπορεί να είναι είτε υλικά π.χ τρόφιμα, καύσιμα κ.α είτε άυλα π.χ υπηρεσίες.

Το πεδίο των Logistics δηλαδή έχει ως στόχο το τελικό προϊόν να φτάσει στον πελάτη τη σωστή χρονική στιγμή, στην κατάλληλη ποσότητα και ποιότητα, στο επιθυμητό μέρος και στη σωστή τιμή.

Κεφάλαιο 2

Δρομολόγηση οχημάτων

2.1 Το πρόβλημα δρομολόγησης οχημάτων (VRP)

Η διαχείριση ενός στόλου οχημάτων αποτελούσε πάντα σημαντικό πρόβλημα για τις μεταφορικές και τις εταιρίες Logistics. Το υψηλό κόστος των μεταφορών, οι απαιτήσεις των πελατών, η ύπαρξη πολύπλοκων περιορισμών, έχουν οδηγήσει τις επιχειρήσεις στην ανάπτυξη νέων μεθόδων και διαδικασιών οι οποίες στοχεύουν όχι μόνο στην απρόσκοπτη λειτουργία αλλά και στην βελτίωση του κόστους.

Η μεταφορά και διανομή προϊόντων και υπηρεσιών είναι από τις σημαντικότερες δραστηριότητες της εφοδιαστικής αλυσίδας και συνήθως αποτελούν το μεγαλύτερο ποσοστό των δαπανών Logistics μιας επιχείρησης. Η διανομή προϊόντων αφορά την εξυπηρέτηση, σε μια δεδομένη χρονική περίοδο, ενός συνόλου από πελάτες μέσω ενός πλήθους οχημάτων, τα οποία έχουν αφετηρία τους μια συγκεκριμένη τοποθεσία π.χ αποθήκη, χρησιμοποιούνται από δεδομένο αριθμό οδηγών οι οποίοι ακολουθούν ένα συγκεκριμένο οδικό δίκτυο.

Το πρόβλημα δρομολόγησης οχημάτων (VRP) αποτελεί ένα από τα σημαντικότερα προβλήματα βελτιστοποίησης και έχει ως στόχο τον προσδιορισμό των βέλτιστων δρομολογίων στόλου οχημάτων για τη μεταφορά αγαθών από μια κεντρική αποθήκη στις τοποθεσίες των πελατών. Ωστόσο, για την επίλυση του προβλήματος σε πραγματικό περιβάλλον πρέπει να ληφθούν υπόψη αρκετοί περιορισμοί. Αυτοί είναι:

- **Σημεία παραγωγής:** Στην πιο απλή περίπτωση, υπάρχει ένα σημείο διάθεσης που εξυπηρετεί κάποιο συγκεκριμένο αριθμό πελατών κάνοντας χρήση συγκεκριμένου αριθμού οχημάτων.

- **Στόλος οχημάτων:** Ο περιορισμός αυτός αφορά οχήματα με διαφορετική χωρητικότητα ή εξοπλισμό.

- **Χρόνος επίσκεψης:** Τα οχήματα πρέπει να εξυπηρετήσουν κάποια σημεία του δικτύου μεταξύ συγκεκριμένων χρονικών ορίων (χρονικά παράθυρα).

- **Χρόνος εξυπηρέτησης:** Ο χρόνος αυτός θεωρητικά είναι σταθερός. Στην πράξη όμως δεν συμβαίνει κάτι τέτοιο. Αυτό μπορεί να εξαρτηθεί κάθε φορά από τον πελάτη καθώς και από την ποσότητα που του παραδίδεται.

- **Σημαντικότητα πελατών:** Για την κατάστρωση πραγματικών επιχειρησιακών δικτύων δεν λαμβάνεται με τον ίδιο τρόπο υπόψη ο εκάστοτε

πελάτης. Συνήθως δίνεται προτεραιότητα στην εξυπηρέτηση εκείνων που είναι πιο κερδοφόροι για την επιχείρηση.

- **Χαρακτηριστικά δικτύου:** Ένα πραγματικό δίκτυο δρομολόγησης δεν έχει κάθε χρονική στιγμή την ίδια συμπεριφορά. Για τη μοντελοποίηση του πραγματικού δικτύου πρέπει να λαμβάνεται υπόψη η στοχαστικότητα των χρόνων μεταφοράς η οποία εξαρτάται από εξωτερικούς παράγοντες.

Με βάση τους περιορισμούς που αναφέραμε παραπάνω έχουν προκύψει αρκετές παραλλαγές του βασικού προβλήματος (VRP). Οι κατηγορίες των διαφόρων προβλημάτων ποικίλουν ανάλογα με τη χωρητικότητα των οχημάτων, τα χρονικά παράθυρα, την ύπαρξη πολλαπλών αποθηκών, την παραλαβή και παράδοση προϊόντων και τη διανυθείσα απόσταση που αποτελεί και θέμα της παρούσας διπλωματικής εργασίας. Στη συνέχεια ακολουθεί μια μικρή αναφορά στις διάφορες κατηγορίες των περιορισμών.

- *Η χωρητικότητα των οχημάτων (Capacitated Vehicle Routing Problem)*

  Στο περιορισμένης χωρητικότητας πρόβλημα δρομολόγησης οχημάτων έχουμε μια κεντρική αποθήκη, έναν αριθμό πελατών γεωγραφικά διασκορπισμένο σε μια περιοχή και ένα στόλο οχημάτων όπου το κάθε όχημα έχει συγκεκριμένη χωρητικότητα. Κάθε πελάτης έχει μια συγκεκριμένη ζήτηση. Στόχος του προβλήματος είναι η ελαχιστοποίηση του κόστους (αριθμός δρομολογίων ή/και χρόνος-μήκος ταξιδιού) έχοντας πάντα υπόψη πως η συνολική ζήτηση των πελατών που θα εξυπηρετηθούν από ένα όχημα δεν μπορεί να ξεπεράσει τη συνολική χωρητικότητα του οχήματος.

- *Το χρονικό πλαίσιο της επίσκεψης των πελατών - χρονικά παράθυρα (Vehicle Routing Problem with Time Windows)*

  Το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα αποτελεί ουσιαστικά επέκταση του προβλήματος περιορισμένης χωρητικότητας στο όποιο ισχύουν οι περιορισμοί της χωρητικότητας του CVRP αλλά επιπλέον ο κάθε πελάτης πρέπει να εξυπηρετηθεί μέσα σε μια χρονική περίοδο $[a, b]$ που ονομάζεται χρονικό παράθυρο. Σκοπός είναι να βρεθεί ένα σύνολο από διαδρομές, όπου κάθε διαδρομή θα ξεκινά και θα τελειώνει στην αποθήκη χωρίς να παραβιάζονται οι περιορισμοί της χωρητικότητας, οι περιορισμοί από τα χρονικά παράθυρα και να ελαχιστοποιείται το συνολικό μήκος διαδρομών.

- *Οι παραλαβές και η διανομή προϊόντων κατά τη διάρκεια της διαδρομής. (Vehicle Routing Problem with Pick up & Delivery)*

Στην συγκεκριμένη περίπτωση ο εκάστοτε πελάτης έχει τη δυνατότητα εκτός από το να του διανεμηθούν προϊόντα από το όχημα που θα περάσει να γίνει και παραλαβή προϊόντων από αυτόν. Ένας από τους πιο σημαντικούς περιορισμούς αυτού του προβλήματος είναι ότι η διανομή γίνεται πριν την παραλαβή. Επίσης, κάθε κύκλος ξεκινά από την αποθήκη, κάθε πελάτης επισκέπτεται από ένα μόνο κύκλο, κάθε όχημα αντιστοιχεί σε μια μόνο διαδρομή και η συνολική ποσότητα που μεταφέρει κάθε όχημα δεν θα πρέπει να ξεπερνά τη συνολική χωρητικότητα του οχήματος.

- *Η χρήση πολλαπλών αποθηκών (Multi - Depot Vehicle Routing Problem)*

  Εδώ η εξυπηρέτηση των πελατών γίνεται από περισσότερες από μια αποθήκες. Αυτό το πρόβλημα μπορεί να αντιμετωπιστεί με δυο τρόπους: είτε η κάθε αποθήκη να έχει τον δικό της αριθμό οχημάτων και πελατών όπου έτσι ουσιαστικά δημιουργούνται πολλαπλά VRP, είτε το κάθε όχημα να ξεκινά από κάποια αποθήκη να τερματίζει σε μια άλλη ή ακόμα και να σταματά για ανεφοδιασμό σε μια τρίτη αποθήκη. Στόχος, η ελαχιστοποίηση της απόστασης που θα διανύσει το κάθε όχημα επιστέφοντας πάντα στο τέλος σε μια αποθήκη.

2.2 Πρόβλημα δρομολόγησης οχημάτων σε περιορισμένη απόσταση (Distance-Constrained Vehicle Routing Problem)

Το πρόβλημα δρομολόγησης οχημάτων σε περιορισμένη απόσταση (Distance-Constrained Vehicle Routing Problem, DCVRP) αποτελεί μια παραλλαγή του γενικότερου προβλήματος δρομολόγησης οχημάτων.

Στο συγκεκριμένο πρόβλημα έχουμε ένα σύνολο κόμβων (πελατών) γεωγραφικά διασκορπισμένο σε μια συγκεκριμένη περιοχή. Η κάθε λύση του προβλήματος πρέπει να ικανοποιεί δυο δομικούς περιορισμούς. Αυτοί είναι η επίσκεψη σε κάθε πελάτη να γίνεται μια μόνο φορά και να ικανοποιείται ο περιορισμός της απόστασης(χρόνου). Στόχος του προβλήματος είναι η εύρεση διαδρομών εξυπηρέτησης που να ξεκινάνε και να τελειώνουν στην αποθήκη χωρίς να παραβιάζονται οι παραπάνω περιορισμοί.

Το πρόβλημα δρομολόγησης οχημάτων σε περιορισμένη απόσταση μπορεί να αναπαρασταθεί σε ένα μη προσανατολισμένο γράφημα **G=(V,E)** όπου ένας κόμβος **j ∈ V** αντιστοιχεί σε έναν πελάτη και ένα τόξο **e∈E** εκφράζει μια διαδρομή μεταξύ ενός ζεύγους πελατών. Ως **n** χαρακτηρίζεται ο αριθμός των πελατών. Η κεντρική αποθήκη συμβολίζεται με τον κόμβο 0. Κάθε όχημα του στόλου μπορεί να διανύσει συγκεκριμένη απόσταση εντός των ορίων $\mathbf{[d^k_{min}, d^k_{max}]}$.

Πίνακας 2.1

Παράμετροι

n: αριθμός πελατών
m: αριθμός οχημάτων
V: το σύνολο των κορυφών V = {0,…..,n} όπου το 0 είναι η αποθήκη
E: το σύνολο των τόξων E = {(i,j):i, j ∈ V}
c_{ij}^{k}: το κόστος της διανυθείσας απόστασης από το i στο j που εκτελείται από το όχημα k
d_{ij} : η απόσταση από τον πελάτη i στον j
$[d_{min}^{k}, d_{max}^{k}]$:Το εύρος της συνολικής απόστασης που το εκάστοτε όχημα k μπορεί να εκτελέσει

Μεταβλητές

$$\boldsymbol{x_{ij}^{k}} = \begin{cases} 1 \; \alpha\nu \; \tau o \; \tau\acute{o}\xi o \; (i,j) \varepsilon\kappa\tau\varepsilon\lambda\varepsilon\acute{\imath}\tau\alpha\iota \; \alpha\pi\acute{o} \; \tau o \; \acute{o}\chi\eta\mu\alpha \; k \\ 0 \; \sigma\varepsilon \; \kappa\acute{\alpha}\theta\varepsilon \; \acute{\alpha}\lambda\lambda\eta \; \pi\varepsilon\rho\acute{\imath}\pi\tau\omega\sigma\eta \end{cases}$$

$$\boldsymbol{y^{k}} = \begin{cases} 1 \; \alpha\nu \; \tau o \; k \; \chi\rho\eta\sigma\iota\mu o\pi o\iota\varepsilon\acute{\imath}\tau\alpha\iota \\ 0 \; \sigma\varepsilon \; \kappa\acute{\alpha}\theta\varepsilon \; \acute{\alpha}\lambda\lambda\eta \; \pi\varepsilon\rho\acute{\imath}\pi\tau\omega\sigma\eta \end{cases}$$

Η αντικειμενική συνάρτηση γίνεται:

$$\text{MinF(x)} = \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} x_{ij}^{k} * c_{ij}^{k} \qquad (1)$$

υπό :

$$\sum_{j=1}^{n} x_{0j}^{k} = 1 \qquad k=1,\ldots\ldots,n \qquad (2)$$

$$\sum_{i=1}^{n} x_{i0}^{k} = 1 \qquad k=1,\ldots\ldots\ldots,n \qquad (3)$$

$$\sum_{i=0}^{n} \sum_{k=1}^{m} \sum_{i=1 \neq j, i=0}^{n} x_{ij}^{k} = 1 \qquad j=1,\ldots..,m \qquad (4)$$

$\sum_{i=0}^{n} x_{ij}^{k}$ -$\sum_{i=0}^{n} x_{ij}^{k}$=0 j=1,…..,m, k=1,…..n (5)

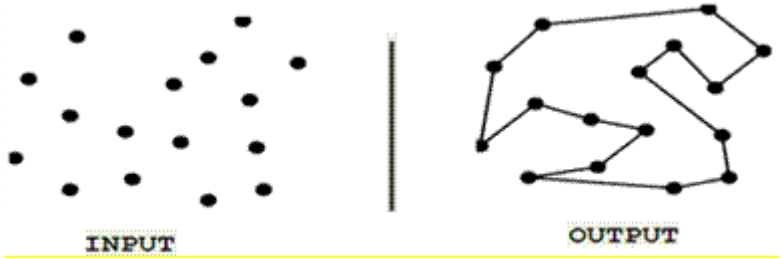
$\sum_{i\in S}\sum_{j\in S} x_{ij}^{k} \leq |S| - 1$ k=1,….,n $\forall S \subseteq$ V,|S|$\in 2, \ldots\ldots, m$ (6)

$d_{min}^{k}\, y^{k} \leq \sum_{i=0}^{n}\sum_{j\neq i,j=0}^{n} d_{ij} x_{ij}^{k} \leq d_{max}^{k} y^{k}$,k=1,….,n (7)

$x_{ij}^{k}\, y^{k}$ $\in${0,1},i=0,…..,m, j=1,…..,m, k=1,……,n

Η αντικειμενική συνάρτηση (1) εκφράζει την ελαχιστοποίηση του κόστους των συνολικών διαδρομών. Οι περιορισμοί (2) και (3) εξασφαλίζουν ότι κάθε διαδρομή θα ξεκινήσει και θα τελειώσει στην αποθήκη. Ο περιορισμός (4) εγγυάται ότι κάθε πελάτης εξυπηρετείται από ένα μόνο όχημα. Ο περιορισμός (5) αποτελεί την εξίσωση διατήρησης της ροής και διασφαλίζει τη συνέχεια της διαδρομής του κάθε οχήματος. Η εκάστοτε διαδρομή που εκτελείται και άρα δεν μπορεί να επαναληφθεί ξανά καλύπτεται από τον περιορισμό (6). Τέλος, η επιτρεπόμενη διανυθείσα απόσταση εκφράζεται από τον περιορισμό (7).

Αξίζει να σημειωθεί πως το συγκεκριμένο πρόβλημα δρομολόγησης οχημάτων παρουσιάζει παρόμοια δομή με το πρόβλημα του πλανόδιου πωλητή (**TSP**). Το πρόβλημα αυτό, απαιτεί τον καθορισμό ενός κύκλου ελαχίστου κόστους που περνά από κάθε κόμβο του συσχετιζόμενου γραφήματος ακριβώς μία φορά.Ο κύκλος αυτός είναι γνωστός και ως κύκλος του Hamilton. Με λίγα λόγια, το πρόβλημα του πλανόδιου πωλητή αφορά την εύρεση συντομότερης διαδρομής για ένα όχημα με αφετηρία ένα σημείο και επιστροφή στο ίδιο, αφού έχει επισκεφτεί έναν αριθμό πελατών ακριβώς μια φορά. Το πρόβλημα του πλανόδιου πωλητή είναι ένα από τα σημαντικότερα προβλήματα συνδυαστικής βελτιστοποίησης. Η απλότητά του και ταυτόχρονα η δυσκολία επίλυσής του το έχουν κάνει ένα ελκυστικό πρόβλημα που έχει οδηγήσει πολλούς ερευνητές να ασχοληθούν με την επίλυσή του κατά τις τελευταίες δεκαετίες.



Σχήμα 2.1

Κεφάλαιο 3

Μεθευρετικοί αλγόριθμοι

Οι μεθευρετικοί αλγόριθμοι είναι μέθοδοι επίλυσης που συνδυάζουν διαδικασίες τοπικής αναζήτησης και υψηλού επιπέδου στρατηγικές για να δημιουργήσουν μια διαδικασία που μπορεί να ξεφύγει από κάποιο τοπικό ελάχιστο. Τα τελευταία χρόνια οι αλγόριθμοι που έχουν κατασκευαστεί για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης ανήκουν στην κατηγορία των μεθευρετικών αλγορίθμων.

3.1 Διαδικασία Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure (GRASP))

Η διαδικασία άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης είναι μια επαναληπτική διαδικασία για την εύρεση προσεγγιστικών λύσεων σε προβλήματα συνδυαστικής βελτιστοποίησης. Αυτή η τυχαιοποιημένη τεχνική παρέχει μια εφικτή λύση σε κάθε επανάληψη. Οι επαναλήψεις της διαδικασίας άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης σταματούν όταν κάποιο κριτήριο τερματισμού ικανοποιείται. Το τελικό αποτέλεσμα είναι απλά η καλύτερη λύση που βρέθηκε από όλες τις επαναλήψεις. Το GRASP ουσιαστικά αποτελείται από δυο φάσεις: κατασκευή και επανεπεξεργασία [Feo and Resende, 1995]. Στην πρώτη φάση γίνεται σταδιακά η κατασκευή μιας αρχικής λύσης (construction phase) ενώ στην δεύτερη φάση γίνεται βελτίωση αυτής της λύσης με μια διαδικασία τοπικής αναζήτησης (local search phase). Στη φάση κατασκευής, μια τυχαιοποιημένη συνάρτηση απληστίας χρησιμοποιείται για να κατασκευαστεί μια αρχική λύση. Αυτή η αρχική λύση στη συνέχεια βελτιώνεται με τη χρήση της διαδικασίας τοπικής αναζήτησης.

Η φάση της κατασκευής παρουσιάζει τρία βασικά χαρακτηριστικά:

1) Μια συνάρτηση απληστίας η οποία χρησιμοποιείται για να καθοριστεί ποιο υποψήφιο στοιχείο θα πρέπει να προστεθεί στην μη ολοκληρωμένη λύση κάθε φορά.

2) Μια τυχαία επιλογή από μια λίστα υποψηφίων που ονομάζεται λίστα περιορισμού των υποψηφίων (Restricted Candidate List). Η λίστα αυτή περιλαμβάνει τους καλύτερους υποψήφιους και εφόσον η επιλογή είναι τυχαία δεν επιλέγεται πάντα ο κορυφαίος υποψήφιος.

3)Τέλος, παρουσιάζει το χαρακτηριστικό της προσαρμοστικότητας με την έννοια ότι η συνάρτηση επιλογής προσαρμόζεται για να προσμετρήσει τα στοιχεία που έχουν ήδη επιλεγεί.

Η κατασκευή της λίστας περιορισμού των υποψηφίων είναι ίσως το πιο σημαντικό κομμάτι της μεθόδου, αφού από αυτό ελέγχεται η διασπορά των λύσεων που θα παραχθούν. Αν, για παράδειγμα, η λίστα είναι πολύ μικρή τότε οι λύσεις που θα παράγονται θα είναι σχεδόν όμοιες μεταξύ τους. Από την άλλη μεριά αν η λίστα είναι πολύ μεγάλη τότε θα πρόκειται πλέον για αλγόριθμο που κατασκευάζεται με τυχαίο τρόπο. Έχουν παρουσιαστεί δύο τρόποι για την κατασκευή της λίστας περιορισμού των υποψηφίων. Στον πρώτο τρόπο η λίστα πάντα έχει ένα καθορισμένο μήκος. Στη δεύτερη περίπτωση για να μπει κάποιο στοιχείο στη λίστα πρέπει η τιμή του να είναι μικρότερη από κάποια τιμή κατωφλίου. Αν υποθέσουμε ότι με $cmin$ και $cmax$ συμβολίζουμε τις μέγιστες και τις ελάχιστες τιμές που θα μπορούσε να έχει ένα στοιχείο που θα συμπεριλαμβανόταν στη λύση τότε για να μπει κάποιο στοιχείο στη λίστα θα πρέπει η τιμή του να είναι ανάμεσα στο διάστημα $\boldsymbol{[cmin, cmin + \alpha \times (cmax - cmin)]}$. Το α είναι μία παράμετρος που δίνεται από το χρήστη και είναι πάρα πολύ σημαντικό διότι αν είναι ίση με 1 τότε ο αλγόριθμος είναι καθαρά ένας τυχαιοποιημένος αλγόριθμος αφού μπορεί να επιλεγεί οποιοδήποτε από τα στοιχεία, ενώ αν είναι ίσο με το 0 τότε είναι καθαρά άπληστος αλγόριθμος γιατί μόνο το βέλτιστο στοιχείο θα εισέλθει. Όσο μικρότερο είναι το α τόσο λιγότερα είναι τα στοιχεία που έχουν πιθανότητες να εισέλθουν στη λύση. Υπάρχουν τρεις τρόποι να επιλεγεί το α. Ο πρώτος τρόπος είναι ο στατικός όπου στο ξεκίνημα του αλγορίθμου επιλέγεται η τιμή του α. Στο δεύτερο τρόπο η τιμή του α επιλέγεται δυναμικά όπου αλλάζει στο ξεκίνημα της κάθε επανάληψης. Ο τρίτος τρόπος είναι να επιλεγεί προσαρμοστικά,όπου κάθε φορά και για κάθε στοιχείο παίρνει και μία διαφορετική τιμή. [Μαρινάκης-Μυγδαλάς, 2008]

Η λύση που δημιουργείται στη φάση κατασκευής δεν εγγυάται ότι είναι τοπικό ελάχιστο, και για αυτό το λόγο μια φάση τοπικής αναζήτησης εφαρμόζεται για να παράξει μια λύση που να περιέχει τοπικό ελάχιστο. Για να εφαρμοστεί η φάση τοπικής αναζήτησης, καθορίζεται μια συνάρτηση που να κάνει αναζήτηση στη γειτονιά της αρχικής λύσης. Κάθε πρόβλημα χρειάζεται διαφορετικές συναρτήσεις απληστίας, διαφορετικές διαδικασίες τοπικής αναζήτησης για την εύρεση του τοπικού ελάχιστου όπως και διαφορετικές συναρτήσεις γειτονιάς που θα γίνει η αναζήτηση. Όταν αυτά προσδιοριστούν τότε το μόνο που μας ενδιαφέρει να εξετάσουμε στην διαδικασία της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης είναι ο αριθμός των επαναλήψεων και το μέγεθος της λίστας. Πολλές φορές για να βελτιωθεί η απόδοση του GRASP μπορεί να χρησιμοποιηθεί ακόμα μία φάση όπου η καλύτερη ή κάποιες από τις καλύτερες λύσεις συνδυάζονται με την καινούρια λύση με στόχο να επιτευχθούν ακόμα καλύτερες λύσεις.

Στην συνέχεια παρουσιάζεται μια γενικευμένη μορφή του αλγορίθμου της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης.

Διαδικασία ΄Απληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης

Αρχικοποίηση

$c(s*) = \infty$

repeat

Κατασκευή μιας αρχικής λύσης s

Εφαρμογή τοπικής αναζήτησης στην s

if $c(s) < c(s*)$ ***then***

$s* = s$ ***endif***

until για όσο το κριτήριο τερματισμού δεν ικανοποιείται

Επέστρεψε τη βέλτιστη λύση $(s*)$.

Ένα μειονέκτημα του GRASP είναι η ανεξαρτησία των επαναλήψεων, δηλαδή το γεγονός ότι δεν μαθαίνει από τις λύσεις που βρέθηκαν σε προηγούμενες επαναλήψεις. Αυτό συμβαίνει διότι ο βασικός αλγόριθμος απορρίπτει πληροφορίες για οποιαδήποτε λύση που βρήκε και δεν είναι η βέλτιστη.

3.2Άλλοι μεθευρετικοί αλγόριθμοι

Περιορισμένη Αναζήτηση (Tabu Search (TS))

Ο μεθευρετικός αλγόριθμος της περιορισμένης αναζήτησης παρουσιάστηκε για πρώτη φορά από τον Grover. Η περιορισμένη αναζήτηση χρησιμοποιεί ένα ευρετικό αλγόριθμο για να μετακινηθεί από τη μια λύση στην άλλη. Όμως υπάρχει η πιθανότητα η λύση να παγιδευτεί σε τοπικό ελάχιστο. Στην περιορισμένη αναζήτηση για να ξεφύγει η λύση από το τοπικό ελάχιστο η επιλογή της επόμενης λύσης δεν γίνεται τυχαία. Ο συγκεκριμένος μεθευρετικός αλγόριθμος έχει μνήμη και κρατά τις προηγούμενες λύσεις που έχουν πραγματοποιηθεί. Για να αποφευχθούν οι συνεχείς επιστροφές σε προηγούμενες λύσεις οι τελευταίες κινήσεις καταγράφονται σε μια λίστα, η οποία ονομάζεται λίστα περιορισμένων κινήσεων (tabulist) και οι συγκεκριμένες κινήσεις απαγορεύεται να επιστρέψουν στη λύση για ένα συγκεκριμένο αριθμό επαναλήψεων.

Αλγόριθμος Επανασύνδεσης Διαδρομών (Path Relinking (PR))

Ο αλγόριθμος της επανασύνδεσης διαδρομών δημιουργεί καινούριες λύσεις εξετάζοντας διαφορετικές τροχιές που αποτελούνται από υψηλές ποιοτικά λύσεις. Ο αλγόριθμος ξεκινάει από μία από αυτές τις λύσεις που ονομάζεται εναρκτήρια λύση (starting solution) και μια λύση στόχου (target solution). Η βασική ιδέα είναι να δημιουργηθούν τροχιές στο χώρο που να συνδέουν την αρχική με την τελική λύση.

Πολλές φορές η καλύτερη από τις δύο λύσεις παίζει το ρόλο της αρχικής λύσης και η χειρότερη παίζει το ρόλο της τελικής λύσης αλλά μπορεί να συμβεί και το αντίστροφο. Στόχος είναι στο τέλος των επαναλήψεων η απόσταση μεταξύ των δυο λύσεων να γίνει μηδενική.

Αλγόριθμος Επαναληπτικής Τοπικής Αναζήτησης (Iterated Local Search (ILS))

Στον αλγόριθμο της επαναληπτικής τοπικής αναζήτησης προσπαθούμε να βελτιώσουμε διαδοχικά τοπικά ελάχιστα. Δηλαδή δημιουργούμε μια αρχική λύση για το πρόβλημα, έπειτα εφαρμόζουμε κάποιον αλγόριθμο τοπικής αναζήτησης και έπειτα σε κάθε επανάληψη εισάγουμε μια διαταραχή στην εκάστοτε λύση που προήλθε από τη διαδικασία της τοπικής αναζήτησης. Στην καινούρια λύση εφαρμόζεται πάλι μία διαδικασία τοπικής αναζήτησης και αυτή η διαδικασία επαναλαμβάνεται μέχρι να επιτευχθεί το κριτήριο του τερματισμού.

Αλγόριθμος Μεταβλητής Γειτονιάς Αναζήτησης (Variable Neighborhood Search (VNS))

Ο αλγόριθμος της μεταβλητής γειτονιάς αναζήτησης προτάθηκε από τους Hansen και Mladenovic. Η βασική ιδέα είναι η χρήση πολλών μεθόδων τοπικής αναζήτησης για να βρεθεί μια καλύτερη λύση ή για να ξεφύγει ο αλγόριθμος από κάποιο τοπικό ελάχιστο. Η μέθοδος αυτή εκμεταλλεύεται το γεγονός ότι διαφορετικές μέθοδοι τοπικής αναζήτησης μπορούν να οδηγήσουν σε διαφορετικά τοπικά ελάχιστα.

Κεφάλαιο 4

Περιγραφή και επίλυση του προβλήματος δρομολόγησής οχημάτων σε περιορισμένη απόσταση

4.1 Εισαγωγή

Στο Κεφάλαιο αυτό παρουσιάζονται πιο αναλυτικά τα επιμέρους χαρακτηριστικά οχημάτων και πελατών καθώς και οι δοθέντες περιορισμοί. Με βάση τα δεδομένα αυτά, περιγράφεται η μοντελοποίηση του προβλήματος και η αναπαράσταση τους στο προγραμματιστικό περιβάλλοντος της MATLAB. Στη συνέχεια, αναλύονται τα επιμέρους βήματα της μεθόδου της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης (**GRASP)** με αρχικό στάδιο την εύρεση αρχικής λύσης με κάποιον αλγόριθμο απληστίας και τελικό στάδιο την εύρεση βέλτιστης λύσης με εφαρμογή τοπικής αναζήτησης.

4.2 Περιγραφή και μοντελοποίηση του προβλήματος

Αρχικά πρέπει να αναφέρουμε ποιες μεταβλητές μας έχουν δοθεί σαν δεδομένα του προβλήματος και ποιες μεταβλητές κατασκευάσαμε εμείς. Ένα από τα δεδομένα που έχουμε είναι η αρίθμηση των πελατών, με τον πελάτη 0 να είναι η αποθήκη. Στο προγραμματιστικό περιβάλλον της Matlab όμως η αποθήκη αναπαρίσταται με τον αριθμό 1, ο πελάτης 1 με τον αριθμό 2, κ.ο.κ. Ένα άλλο δεδομένο που μας δίνεται είναι οι συντεταγμένες χ και y που αντιστοιχούν σε κάθε πελάτη καθώς και ο χρόνος εξυπηρέτησης. Οι συντεταγμένες χ και y αποθηκεύτηκαν σε φύλλα excel, ως .xlsx και διαβάστηκαν από την Matlab με την εντολή xlsread. Πιο συγκεκριμένα, αποθηκεύτηκαν σε ένα δισδιάστατο πίνακα ‘A’.

Αναλυτικότερα,

n: αριθμός κόμβων

A: δισδιάστατος πίνακας στον οποίο είναι αποθηκευμένη η τετμημένη χ και η τεταγμένη y του κάθε πελάτη.

Προκειμένου να επιλύσουμε το πρόβλημα που έχουμε θα πρέπει να κατασκευάσουμε κάποιους επιπλέον πίνακες. Ο πρώτος πίνακας που χρειαζόμαστε είναι αυτός που θα υπολογίζει τις αποστάσεις μεταξύ όλων των κόμβων. Ο πίνακας αυτός συμβολίζεται ως C(i,j). Η απόσταση που υπολογίζεται είναι η ευκλείδεια, ο πίνακας είναι συμμετρικός που σημαίνει ότι η απόσταση που διανύουμε για να πάμε από τον κόμβο i στον κόμβο j, ισούται με την απόσταση από τον j στον i. Επίσης, η παραμονή σε ένα κόμβο απαγορεύεται για αυτό το λόγο η απόσταση C(i,i) μηδενίζεται έτσι ώστε να μην ληφθεί υπόψη στην επίλυση του προβλήματος. Η διάσταση του πίνακα C θα είναι nxn δηλαδή όσο είναι και το πλήθος των κόμβων (πελάτες και αποθήκη μαζί). Ο υπολογισμός της ευκλείδειας απόστασης έγινε ως εξής:

$$C(i,j)=\sqrt{((A(i,1)-A(j,1))^2-(A(i,2)-A(j,2))^2}$$

Πρέπει να σημειωθεί ότι ο χρόνος που θα χρειαστούμε για να μεταβούμε από τον πελάτη i στον j ισούται με την απόσταση που θα διανύσουμε. Στην συνέχεια, όταν θα χρησιμοποιούμε τα δεδομένα του πίνακα C, θα αναφερόμαστε στο χρόνο.

4.3 Εύρεση αρχικής λύσης με τον αλγόριθμο του πλησιέστερου γείτονα

Για την εύρεση μιας αρχικής εφικτής λύσης χρησιμοποιούμε έναν απλό αλγόριθμο απληστίας. Στην συγκεκριμένη διπλωματική εργασία χρησιμοποιείται ο αλγόριθμος του πλησιέστερου γείτονα. Σε αυτήν εδώ τη διαδικασία ξεκινάμε από την αποθήκη και στη συνέχεια επισκεπτόμαστε τον πελάτη που είναι πλησιέστερος σε αυτήν. Από αυτόν πηγαίνουμε στον επόμενο πλησιέστερο του που δεν έχουμε επισκεφτεί ως τώρα, και συνεχίζουμε την ίδια διαδικασία μέχρι να επισκεφτούμε όλους τους πελάτες και να επιστρέψουμε στην αποθήκη από όπου ξεκινήσαμε. Ο βασικός επαναληπτικός βρόχος ο οποίος έχει ως στόχο να ταξινομήσει τους πελάτες με βάση το διάνυσμα της απόστασης (χρόνου) εκτελείτε μέχρι να ελεγχθούν όλοι οι πελάτες. Ο αλγόριθμός ελέγχει σε κάθε βήμα αν το όχημα μπορεί να εξυπηρετήσει έναν νέο πελάτη χωρίς να παραβιάσει τους χρονικούς περιορισμούς. Αν οι περιορισμοί παραβιάζονται το όχημα επιστρέφει στην αποθήκη και ένα νέο όχημα ξεκινά. Η εκτέλεση συνεχίζεται μέχρι να εξυπηρετηθούν όλοι οι πελάτες.

Αρχικά, αποθηκεύουμε για λόγους ευκολίας τον πίνακα αποστάσεων **C** σε έναν πίνακα **c**. Στο σημείο αυτό θέλουμε να φτιάξουμε ένα διάνυσμα στο οποίο θα είναι ταξινομημένοι όλοι οι πελάτες. Για την εύρεση των μη εξυπηρετημένων πελατών, μπορούμε να αξιοποιήσουμε την ιδιότητα της δομής **c** ότι πελάτες στους οποίους δεν έχει πραγματοποιηθεί επίσκεψη θα έχουν στήλη αποστάσεων διάφορη του απείρου. Οι πελάτες που έχουν ήδη εξυπηρετηθεί θα έχουν στήλη με όλες τις τιμές ίση με το άπειρο ώστε να μην υπάρχει πιθανότητα να ξαναχρησιμοποιηθούν. Όταν εκτελεστούν όλες οι επαναλήψεις και έχει κατασκευαστεί ο ταξινομημένος πίνακας στη συνέχεια θα πρέπει να ελεγχτούν οι περιορισμοί. Για να ελέγξουμε τον χρονικό περιορισμό θα πρέπει να δημιουργήσουμε μια μεταβλητή η οποία θα κρατά το χρόνο που δαπανάτε για να φτάσει το φορτηγό αρχικά από την αποθήκη στον πρώτο πελάτη, από τον πρώτο στον δεύτερο κ.ο.κ συν το χρόνο εξυπηρέτησης (ίδιος για κάθε πελάτη). Βέβαια στον περιορισμό αυτό θα πρέπει να λαμβάνεται υπόψη και ο χρόνος επιστροφής στην αποθήκη. Για παράδειγμα έστω ότι έχουμε έναν συμμετρικό πίνακα αποστάσεων 5x5 (όπου D η αποθήκη) . Έστω, ότι ο συνολικός χρόνος που θα πρέπει να διανύσει το όχημα σε κάθε διαδρομή είναι 160.

	D	1	2	3	4
D	—	87	64	37	93
1	87	—	56	32	35
2	64	56	—	91	54
3	37	32	91	—	43
4	93	35	54	43	—

Ξεκινώντας από την αποθήκη πλησιέστερος στην αποθήκη πελάτης είναι ο 3 και από τον 3 πλησιέστερος είναι ο 1. Από τον 1 στον 4 δεν μπορούμε να μεταβούμε διότι θα παραβιαστεί ο περιορισμός του χρόνου. Επομένως, επιστρέφουμε στην αποθήκη και ελέγχουμε ποιος είναι ο επόμενος πλησιέστερος πελάτης που δεν έχουμε επισκεφθεί. Ο πελάτης αυτός είναι ο 2. Από τον 2 όμως δεν μπορούμε να μεταβούμε στον 4 διότι καταπατάται ο περιορισμός κατά την επιστροφή από τον 4 στην αποθήκη. Οπότε επιστρέφουμε άμεσα από τον 2 στην αποθήκη. Άρα, οι διαδρομές διαμορφώνονται ως εξής:

$$D \rightarrow 3 \rightarrow 1 \rightarrow D$$

$$D \rightarrow 2 \rightarrow D$$

$$D \rightarrow 4 \rightarrow D$$

Έτσι, με τον έλεγχο του περιορισμού κατασκευάζουμε το κατάλληλο διάνυσμα **ROUTE** το οποίο περιλαμβάνει τα δρομολόγια της διαδρομής που θα πραγματοποιηθεί και υπολογίζουμε το συνολικό κόστος της διαδρομής το οποίο εκχωρείται στην μεταβλητή **COST.** Στην συνέχεια, εκχωρούμε την μεταβλητή **COST** στην **BEST_COST** και το διάνυσμα **ROUTE** στο **BEST_ROUTE** έτσι ώστε να τις συγκρίνουμε με την λύση του αλγορίθμου της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης και να κρατήσουμε την καλύτερη.

4.4Υλοποίηση αλγορίθμου άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης

4.4.1Εύρεση Αρχικής Εφικτής Λύσης με τον Αλγόριθμο Απληστίας

Για την κατασκευή της αρχικής λύσης που αποτελεί την πρώτη φάση για την υλοποίηση του GRASP χρησιμοποιούμε μια τυχαιοποιημένη συνάρτηση απληστίας. Χρησιμοποιώντας και πάλι τον αλγόριθμο του πλησιέστερου γείτονα αφού κατασκευάσουμε τη συνάρτηση απληστίας μπορούμε πολύ εύκολα να την τυχαιοποιήσουμε.

Πιο συγκεκριμένα, δημιουργούμε χρησιμοποιώντας τον αλγόριθμο του πλησιέστερου γείτονα ένα ταξινομημένο διάνυσμα πελατών

route_nearest_neighbor. Μετά, τυχαιοποιούμε το διάνυσμα αυτό. Κάθε φορά επιλέγουμε ανά πόσους υποψήφιους θέλουμε να γίνει η τυχαιοποίηση πχ ανά 2,ανά 5, ανά 10 κ.ο.κ. Στην συνέχεια, κατασκευάζουμε το διάνυσμα **index_of_points_new** το οποίο αποτελείται από μια τυχαιοποιημένη σειρά δεικτών ομαδοποιημένων ανά 2,5,10 κ.λ.π υποψήφιους (ανάλογα την κάθε περίπτωση). Για το σκοπό αυτό χρησιμοποιούμε την συνάρτηση της Matlab 'randperm()', που επιστρέφει τυχαία αρίθμηση. Δηλαδή για εκτέλεση randperm(4) έχουμε μία τυχαία σειρά των αριθμών από 1 έως 4.

ans = 3 1 2 4

Επίσης, στην περίπτωση π.χ που έχουμε μια λίστα υποψηφίων με 6 πελάτες και θέλουμε να τους τυχαιοποιήσουμε ανά 2 η συνάρτηση που χρησιμοποιούμε είναι:

(N-1)* nodes of group +randperm(nodes of group)

Όπου το nodes of group αντιστοιχεί στο 2, N= $\frac{\text{arithmos pelatwn}}{\text{nodes of group}}$ αντιστοιχεί στις ομάδες στις οποίες χωρίζονται οι υποψήφιοι σε κάθε περίπτωση. Έτσι σε μια επαναληπτική διαδικασία η οποία εκτελείται από 1 μέχρι N (3 στο παράδειγμά μας) για την πρώτη ομάδα πελατών όπου το N=1 θα έχουμε randperm(2) για τη δεύτερη ομάδα όπου N=2 θα έχουμε 2+ randperm(2) και για την τρίτη (3-1)*2+ randperm(2)=4+ randperm(2). Με αυτόν τον τρόπο κατασκευάζουμε ένα διάνυσμα στο οποίο οι πρώτες 2 θέσεις θα είναι είτε 1 2 είτε 2 1, οι επόμενες 2 θα είναι είτε 3 4 είτε 4 3 και οι τελευταίες 2, 5 6 ή 6 5. Έστω ότι το διάνυσμα που προκύπτει είναι:

2 1 3 4 6 5

Στη συνέχεια, χρησιμοποιώντας την εντολή **route_nearest_neighbor (index_of_points_new)** δημιουργούμε το νέο διάνυσμα **route_nearest_neighbor** στο οποίο οι ταξινομημένοι πελάτες έχουν αλλάξει με βάση την αλλαγή των δεικτών. Αντί λοιπόν να τυχαιοποιήσουμε τις δομές δεδομένων (**route_nearest_neighbor**) με βάση το αποτέλεσμα της ταξινόμησης τυχαιοποιούμε δείκτες. Στην συνέχεια, ελέγχουμε τους χρονικούς περιορισμούς όπως αναφέρθηκε παραπάνω και δημιουργούμε τις μεταβλητές **ROUTE** και **COST**.

4.4.2Εφαρμογή τοπικής αναζήτησης

Η λύση που δημιουργείται στη φάση κατασκευής της αρχικής εφικτής λύσης δεν εγγυάται ότι είναι τοπικό ελάχιστο, και για αυτό το λόγο εφαρμόζεται μια φάση τοπικής αναζήτησης. Για να εφαρμοστεί η φάση τοπικής αναζήτησης, καθορίζεται μια συνάρτηση που να κάνει αναζήτηση στη γειτονιά της αρχικής λύσης, όπως ανταλλαγές κόμβων μεταξύ διαφορετικών δρομολογίων, είτε κόμβων του ίδιου δρομολογίου.

Αρχικά, αποθηκεύουμε το διάνυσμα **routenew** σε έναν καινούριο διάνυσμα **temproutenew.** Χρησιμοποιώντας δυο βοηθητικές μεταβλητές εκχωρούμε σε αυτές με την βοήθεια της συνάρτησης randi(n-1) δυο τυχαίες τιμές από το 1 μέχρι το n (αριθμός των κόμβων). Έτσι, χρησιμοποιούμε την randi(n-1) για να μην ληφθεί υπόψη η αποθήκη. Εδώ, πρέπει να σημειωθεί ότι οι μεταβλητές αυτές αντιπροσωπεύουν θέσεις στο διάνυσμα temproutenew και όχι στοιχεία. Στην συνέχεια, κάνουμε αντιμετάθεση των δύο μεταβλητών. Για παράδειγμα, σε ένα διάνυσμα 7 4 9 5 8 3 10 2 6 αν το randi(9)=3 και το randi(9)=7 τότε θα γίνει αντιμετάθεση του 9 με το 10 και το διάνυσμα θα γίνει 7 4 10 5 8 3 9 2 6.

Εφόσον έχει γίνει η ανταλλαγή μεταξύ δυο κόμβων είτε του ίδιου δρομολογίου είτε διαφορετικών ελέγχονται και πάλι οι περιορισμοί του χρόνου και κατασκευάζεται το **TEMPROUTE** το οποίο περιλαμβάνει όλες τις διαδρομές και το **TEMPCOST.** Αν το **TEMPCOST** είναι μικρότερο του κόστους **COST** που βρήκαμε κατά την κατασκευή της αρχικής εφικτής λύσης στο GRASP τότε στην μεταβλητή **COST** θα εκχωρηθεί το **TEMPCOST** και στο **ROUTE** θα εκχωρηθεί το διάνυσμα **TEMPROUTE.** Σε αντίθετη περίπτωση, θα διατηρηθούν οι τιμές του **ROUTE** και του **COST** που βρήκαμε κατά τη δημιουργία του τυχαιοποιημένου άπληστου αλγορίθμου. Αυτή η φάση ολοκληρώνεται μετά από **Num_local** επαναλήψεις. Στο τέλος κάθε επανάληψης i, γίνεται σύγκριση της **ROUTE** που βρέθηκε σε αυτή την επανάληψη με το **BEST_ROUTE** της i-1 και αν το **COST** είναι μικρότερο ή ίσο του **BEST_COST** τότε η λύση του **BEST_ROUTE** αντικαθιστάται με αυτή της **ROUTE.** Το ίδιο συμβαίνει και για τη μεταβλητή του κόστους. Ο αλγόριθμος τερματίζει όταν ολοκληρωθούν όλες οι επαναλήψεις **Num_GRASP.**

4.4.3 Τερματισμός αλγορίθμου και ψευδοκώδικας

Η επαναληπτική διαδικασία επαναλαμβάνεται για όσες επαναλήψεις προκαθορίσουμε εμείς. Όσο περισσότερες είναι οι επαναλήψεις τόσο αυξάνονται οι πιθανότητες μας να βρούμε καλύτερο αποτέλεσμα.

Η διαδικασία επίλυσης του προβλήματος παρουσιάζεται στον ακόλουθο ψευδοκώδικα:

Αρχικοποίηση

Επέλεξε παράδειγμα
Επέλεξε αριθμό λύσεων GRASP (Num_GRASP)
Επέλεξε αριθμό επαναλήψεων 1-1 exchange (Num_local)
Υπολόγισε πίνακα κόστους C
Πάραξε την πρώτη διαδρομή με πλησιέστερο γείτονα (ROUTE)
Λαμβάνοντας τους περιορισμούς υπολόγισε το συνολικό κόστος διαδρομών (COST)
Εκχώρησε το ROUTE στο BEST_ROUTE
Εκχώρησε το COST στο BEST_COST

Κυρίως αλγόριθμος

Για Num_GRASP επαναλήψεις
    Πάραξε λύση με βάση τον αλγόριθμο GRASP (ROUTE)
    Υπολόγισε το συνολικό κόστος διαδρομών (COST)
    Αν COST≤ BEST_COST τότε
        BEST_ROUTE=ROUTE

```
            BEST_COST=COST
        Τέλος_Αν
        Για Num_local επαναλήψεις
            Αποθήκευσε το   ROUTE στο διάνυσμα TEMPROUTE
            Κάνε 1-1 exchange στην TEMPROUTE
            Υπολόγισε το κόστος της TEMPROUTE (TEMPCOST)
            Αν TEMPCOST≤ COST τότε
                ROUTE=TEMPROUTE
                COST= TEMPCOST
            Τέλος_Αν
        Τέλος_για
       Αν COST≤ BEST_COST τότε
            BEST_ROUTE=ROUTE
            BEST_COST=COST
       Τέλος_Αν
Τέλος_για
Επέστρεψε το BEST_ROUTE και το BEST_COST
```

Κεφάλαιο 5

5.1 Περιγραφή και ανάλυση αποτελεσμάτων-Συμπεράσματα

Στην παρούσα εργασία επιλύθηκαν έξι προβλήματα δρομολόγησης τα όποια ποικίλουν όσον αφορά το πλήθος των κόμβων, τον χρόνο παραμονής σε κάθε πελάτη, το συνολικό χρόνο διαδρομής αλλά και την ομαδοποίηση των υποψήφιων που πραγματοποιήθηκε κατά την κατασκευή της αρχικής λύσης του GRASP.

Για κάθε πρόβλημα ο αλγόριθμος εκτελέστηκε πολλές φορές ώστε να πραγματοποιηθούν δόκιμες στον κώδικα και στις τιμές των παραμέτρων που θα χρησιμοποιηθούν ώστε να επιτευχθούν τα καλυτέρα δυνατά αποτελέσματα. Οι παράμετροι αφορούσαν τον αριθμό των επαναλήψεων και το μέγεθος της λίστα των υποψήφιων πελατών. Σε κάθε πρόβλημα πραγματοποιήθηκαν 10 εκτελέσεις του αλγορίθμου ώστε να υπάρχει μια ολοκληρωμένη εικόνα της απόδοσης του.

Α. Παράδειγμα par6 με 51 κόμβους

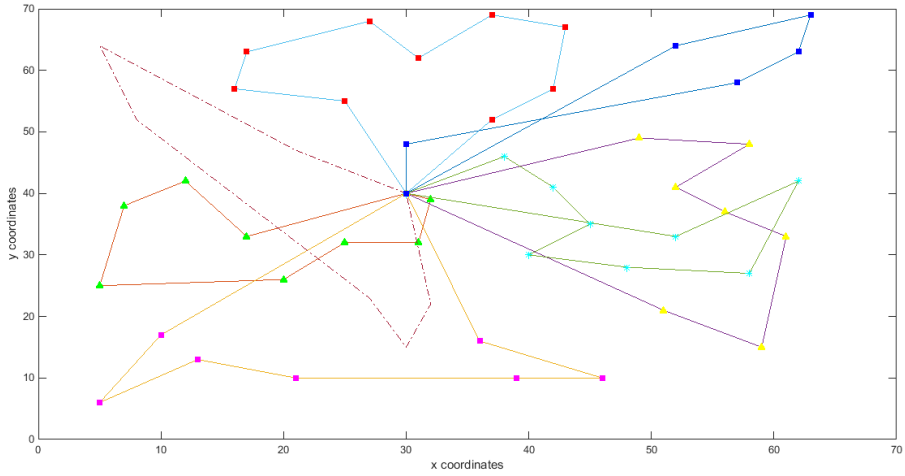
Στο συγκεκριμένο πρόβλημα η λίστα περιορισμού των υποψηφίων ομαδοποιήθηκε ανά 2, 5 και 10 πελάτες. Σε κάθε περίπτωση υλοποιήσαμε 3 διαφορετικές κατηγορίες επαναλήψεων. Για αριθμό επαναλήψεων του εξωτερικού βρόχου 100 (Num_GRASP=100) και αριθμό επαναλήψεων 20.000 για τον αλγόριθμο της τοπικής αναζήτησης (Num_local=20.000), για αριθμό επαναλήψεων του εξωτερικού βρόχου 100 (Num_GRASP=100) και αριθμό επαναλήψεων 200.000 (Num_local=200.000), και τέλος για Num_GRASP=1.000 και Num_local=200.000. Παρακάτω παρουσιάζονται οι πίνακες των αποτελεσμάτων για τις 10 εκτελέσεις του αλγορίθμου για τις 3 κατηγορίες του αριθμού της λίστας υποψήφιων πελατών (2 πελάτες, 5 πελάτες και 10 πελάτες) και τις 3 κατηγορίες επαναλήψεων, καθώς και τα γραφήματα των καλύτερων λύσεων που βρέθηκαν σε κάθε κατηγορία.

Λίστα περιορισμού των υποψήφιων με 2 πελάτες

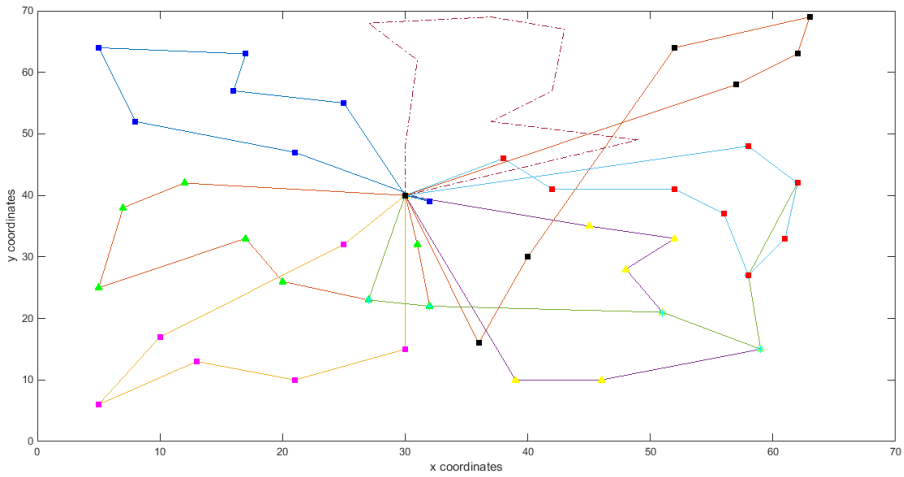
Πίνακας 5.1 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par6 για τυχαιοποίηση ανά 2 υποψήφιους πελάτες.

Par6	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	**720,6072**	721,1012	700,6722
2	756,9446	759,8659	712,9073
3	788,6553	697,8387	703,4645
4	807,6958	741,3883	736,7462
5	770,8927	757,8789	713,8829
6	795,101	696,6279	710,6117
7	761,2081	**695,9116**	716,1409
8	766,9977	757,4485	**673,4346**
9	797,1494	776,8453	688,5321
10	737,7001	760,2736	718,4576
M.O.	770,2952	760,2736	707,485

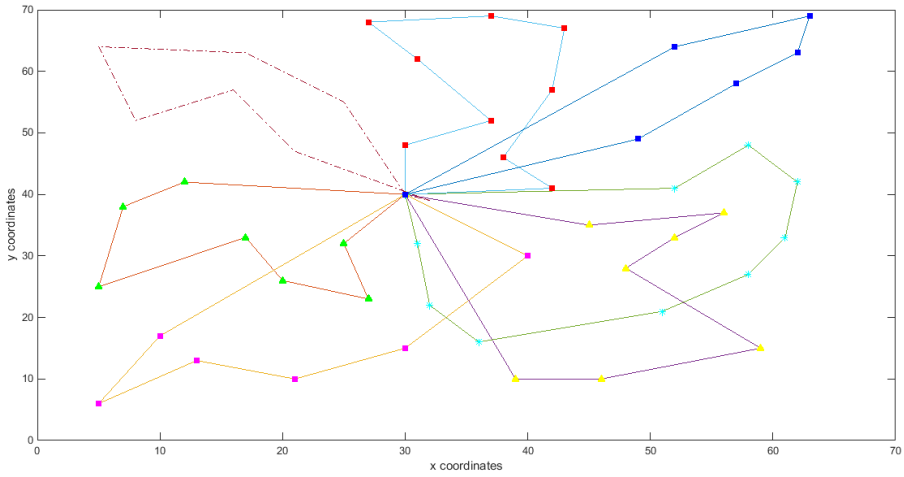
Εικόνα 5.1 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 1 με COST=720,6072)



Εικόνα 5.2 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=695,9116)



Εικόνα 5.3 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=673,4346)



Για λίστα περιορισμού με 2 υποψήφιους πελάτες, παρατηρούμε ότι όσο αυξάνεται ο αριθμός των επαναλήψεων τόσο βελτιώνονται οι λύσεις μας. Η καλύτερη λύση καθώς

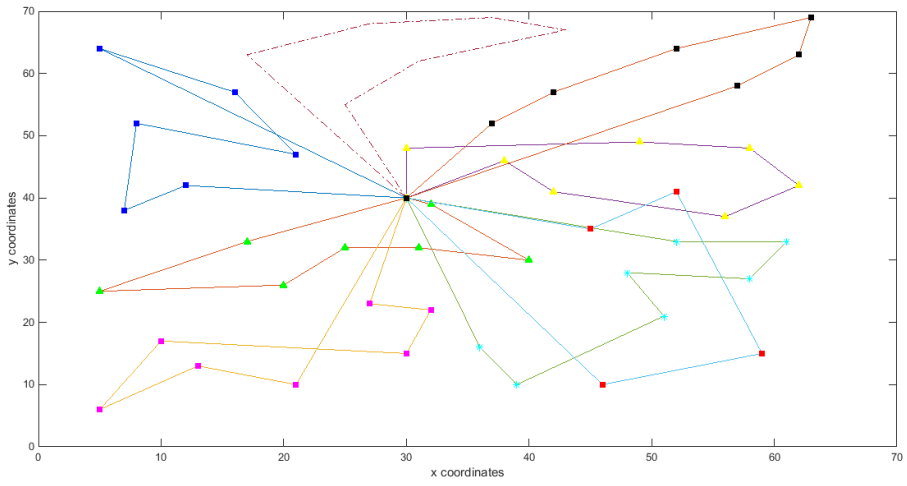
και ο καλύτερος μέσος όρος των 10 εκτελέσεων βρέθηκε για τον μέγιστο αριθμό επαναλήψεων που έχουμε θέσει (Num_GRASP=1.000 & Num_local=200.000). Αυτό είναι λογικό εφόσον πραγματοποιείται μεγαλύτερη διερεύνηση στη διαδικασία της επίλυσης.

Λίστα περιορισμού των υποψήφιων με 5 υποψήφιους πελάτες

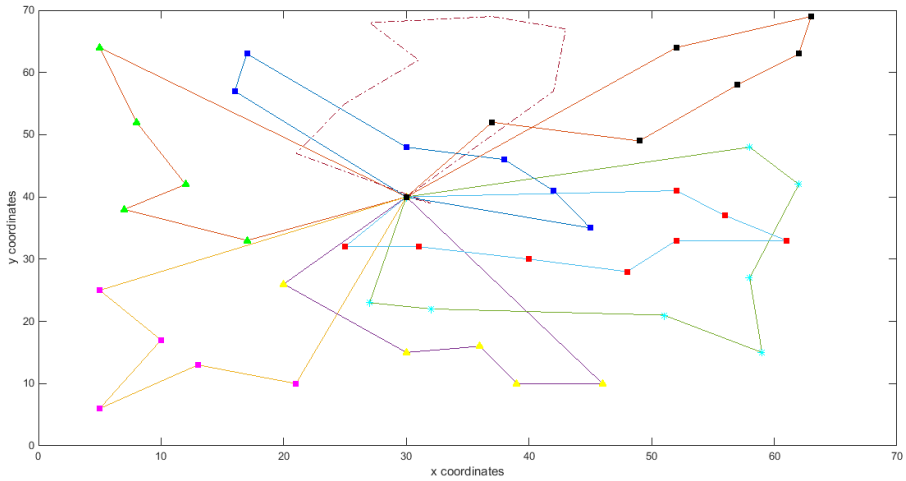
Πίνακας 5.2 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par6 για τυχαιοποίηση ανά 5 υποψήφιους πελάτες.

Par6	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	863,1034	744,7373	749,4294
2	901,0783	749,8753	763,7709
3	779,0219	784,022	771,3529
4	768,7179	783,9575	779,845
5	795,8965	780,3803	789,6595
6	793,3446	826,3774	765,1408
7	**774,1997**	805,0954	**737,9484**
8	795,2706	800,181	769,5061
9	832,2401	763,3005	794,3937
10	788,1971	**744,5099**	751,3778
M.O.	809,1071	778,2437	767,2424

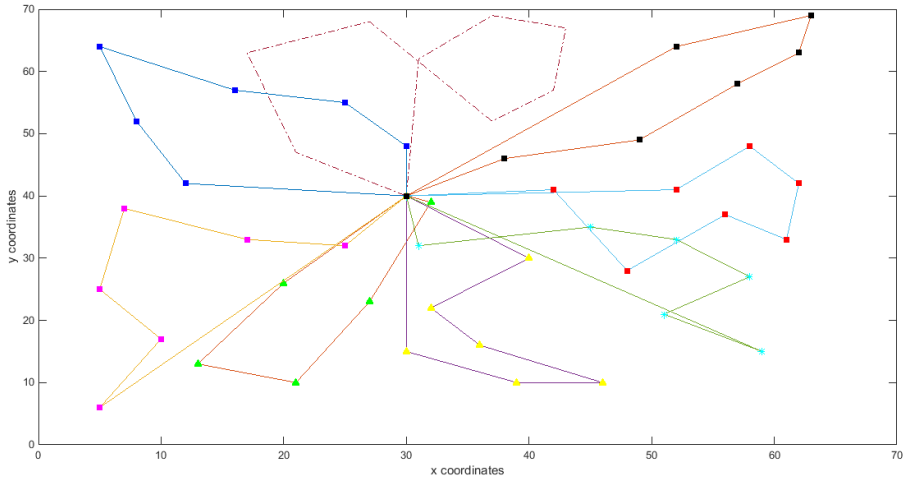
Εικόνα 5.4 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=774,1997)



Εικόνα 5.5 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=744,5099)



Εικόνα 5.6 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=737,9484)



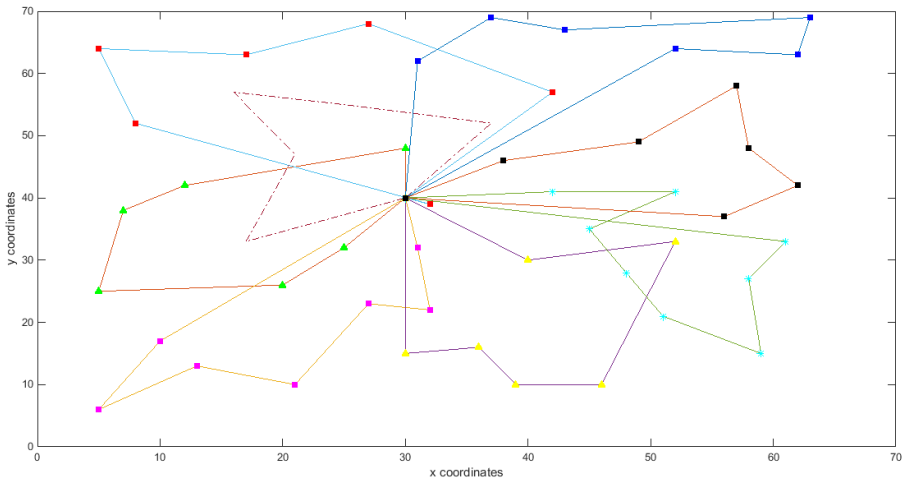
Στην λίστα με τους 5 υποψηφίους, όπως και στην προηγούμενη περίπτωση (par6, 2 υποψήφιοι πελάτες) παρατηρούμε ότι όσο αυξάνεται ο αριθμός των επαναλήψεων τόσο καλύτερα αποτελέσματα βρίσκουμε. Έτσι, η καλύτερη λύση (COST=737,9484) καθώς και ο καλύτερος μέσος όρος (M.O.=767,2424) βρέθηκε για τον μέγιστο αριθμό επαναλήψεων που έχει τεθεί στην παρούσα ομαδοποίηση των πελατών.

Λίστα περιορισμού των με 10 υποψήφιους πελάτες

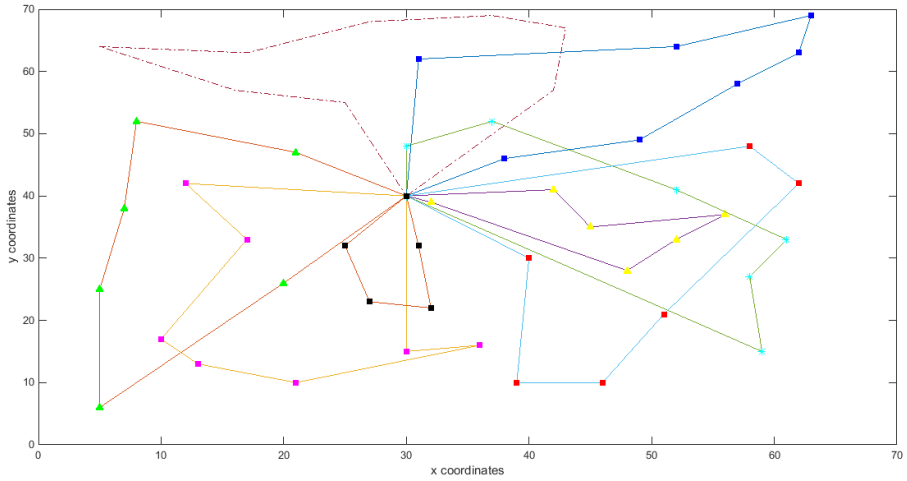
Πίνακας 5.3 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par6 για τυχαιοποίηση ανά 10 υποψήφιους πελάτες.

Par6	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	786,2477	793,9478	757,9157
2	901,0783	**753,2372**	767,0427
3	871,9803	816,5083	788,6915
4	900,9739	848,9526	813,1742
5	857,5687	827,5553	808,9804
6	806,3159	822,7395	780,9876
7	**751,7304**	802,4008	809,465
8	815,5768	759,1463	797,013
9	785,5322	792,2725	845,0898
10	803,0916	814,0377	**755,3877**
M.O.	828,0096	803,0798	792,3748

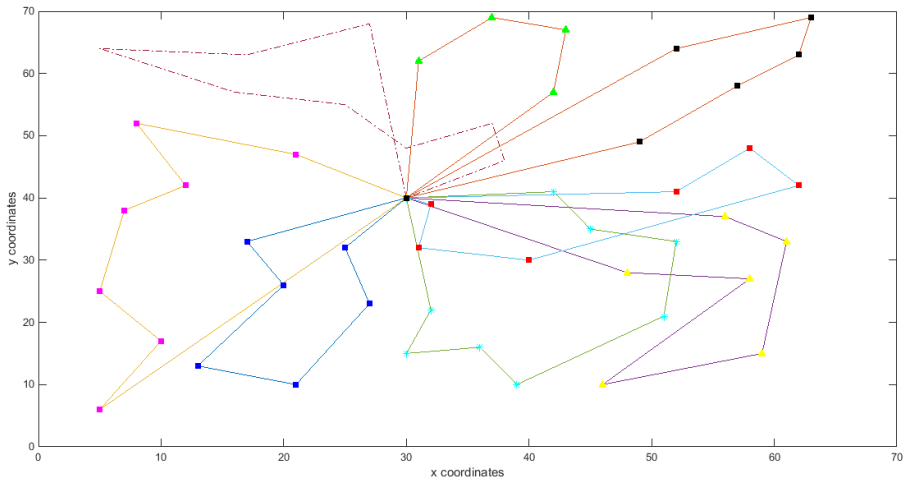
Εικόνα 5.7 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 10 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=751,7304)



Εικόνα 5.8 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 10 υποψήφιους (Βέλτιστη εκτέλεση η No 2 με COST=753,2372)



Εικόνα 5.9 Βέλτιστη διαδρομή για παράδειγμα par6 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 10 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=755,3877)



Εδώ, καλύτερη λύση βρήκαμε για τον ελάχιστο αριθμό επαναλήψεων. Αυτό, ίσως να οφείλεται στην τυχαιότητα του αλγορίθμου. Βέβαια, όσο αυξάνεται ο αριθμός των επαναλήψεων τόσο βελτιώνονται οι μέσοι όροι των 10 εκτελέσεων με καλύτερο μέσο όρο για Num_GRASP=1.000 και Num_local=200.000 (Μ.Ο.=792,3748).

Αυτό που μπορούμε να παρατηρήσουμε στο συγκεκριμένο παράδειγμα των 51 κόμβων είναι, ότι όταν η λίστα περιορισμού των υποψήφιων περιλαμβάνει 2 πελάτες βρίσκουμε καλύτερα αποτελέσματα. Αυτό συμβαίνει διότι μειώνεται το εύρος της τυχαιοποίησης. Όσο αυξάνεται ο αριθμός της λίστας και κατ' επέκταση το εύρος τόσο χειρότερα τα αποτελέσματα. Επίσης, παρατηρούμε ότι οι λύσεις βελτιώνονται με την αύξηση του αριθμού των επαναλήψεων λόγω της μεγαλύτερης διερεύνησης που πραγματοποιείται στις γειτονίες αναζήτησης.

Β. Παράδειγμα par7 με 76 κόμβους

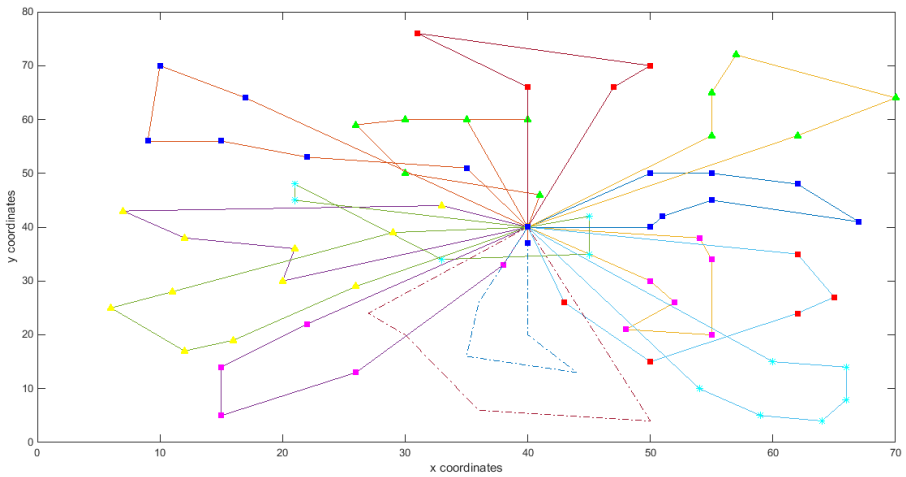
Στο συγκεκριμένο πρόβλημα η λίστα των υποψήφιων πελατών περιλάμβανε 5 και 15 πελάτες. Και εδώ εκτελέσαμε τρείς κατηγορίες επαναλήψεων, για αριθμό επαναλήψεων του εξωτερικού βρόχου 100 (Num_GRASP=100) και αριθμό επαναλήψεων 20.000 για τον αλγόριθμο της τοπικής αναζήτησης (Num_local=20.000), για Num_GRASP=100 και Num_local=200.000 και για Num_GRASP=1.000 και Num_local=200.000. Τα αποτελέσματα που προέκυψαν παρουσιάζονται στις επόμενες σελίδες, μέσω των πινάκων και των σχεδιαγραμμάτων.

Λίστα περιορισμού των υποψήφιων με 5 υποψήφιους πελάτες

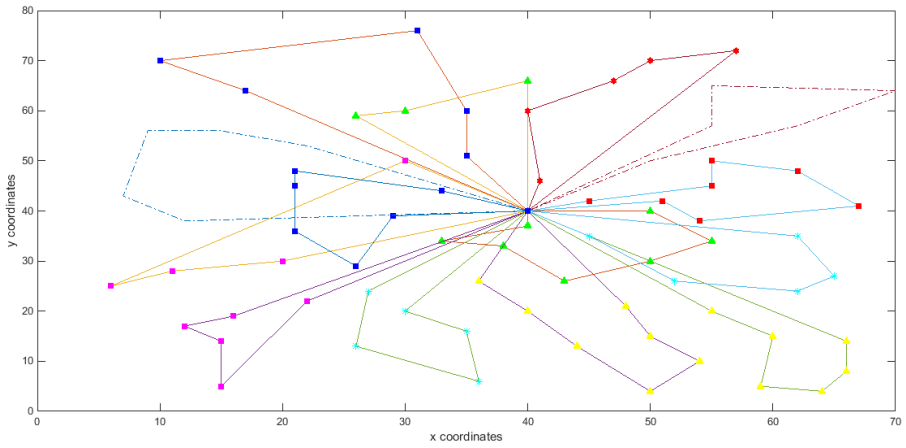
Πίνακας 5.4 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par7 για τυχαιοποίηση ανά 5 υποψήφιους.

Par7	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	1193,696	1194,1	1160,193
2	1173,623	1172,767	1173,022
3	1174,366	1114,083	1125,874
4	1173,199	1137,412	1163,912
5	1159,351	1170,89	1171,657
6	1188,237	1105,524	1095,278
7	**1125,012**	1167,85	**1089,99**
8	1169,064	**1101,243**	1141,693
9	1190,049	1172,551	1190,104
10	1187,901	1176,999	1190,566
M.O.	1173,45	1150,229	1151,342

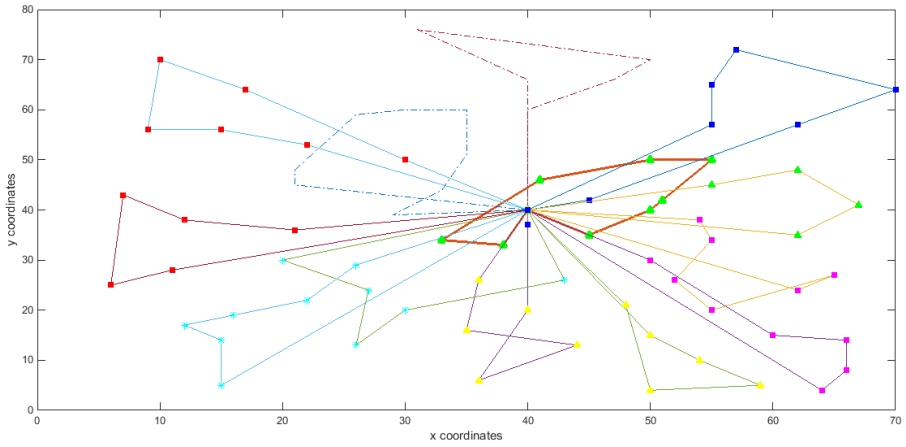
Εικόνα 5.10 Βέλτιστη διαδρομή για παράδειγμα par7 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=1125,0118)



Εικόνα 5.11 Βέλτιστη διαδρομή για παράδειγμα par7 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1101,2429)



Εικόνα 5.12 Βέλτιστη διαδρομή για παράδειγμα par7 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψηφίους (Βέλτιστη εκτέλεση η No 7 με COST=1089,9895)



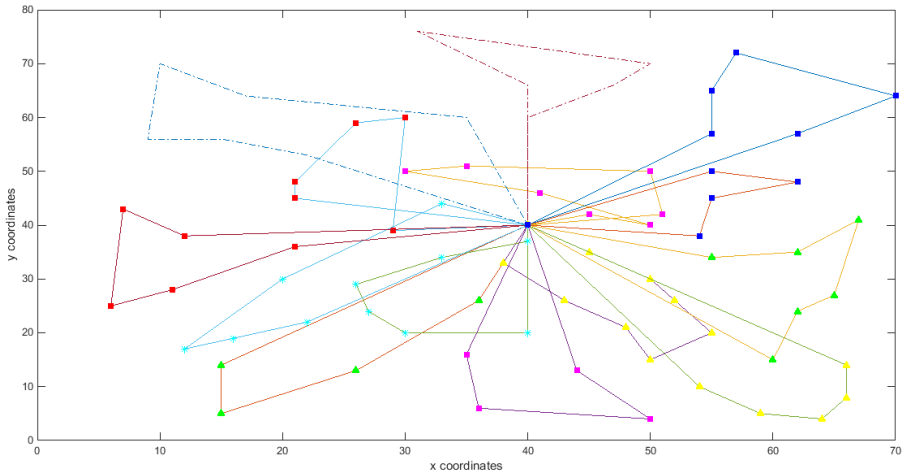
Στη συγκεκριμένη περίπτωση καλύτερη λύση βρέθηκε για Num_GRASP=1.000 και Num_local=200.000 (COST=1089,9895). Εδώ όμως, ο μέσος όρος των 10 εκτελέσεων ήταν χειρότερος από την δεύτερη κατηγορία επαναλήψεων για Num_GRASP=100 και Num_local=200.000. Αυτή η χειρότερη λύση που βρέθηκε μπορεί να οφείλεται σε πολλούς παράγοντες που επηρεάζουν το τελικό μας βέλτιστο αποτέλεσμα, όπως η τυχαιότητα που διακρίνει την αρχική εφικτή λύση ή τον αλγόριθμο της τοπικής αναζήτησης. Ωστόσο, σε μεγάλο βαθμό το τελικό βέλτιστο κόστος είναι ικανοποιητικό καθώς βρίσκεται κοντά στις προηγούμενες βέλτιστες λύσεις που παρουσιάστηκαν και στις προηγούμενες ενότητες.

Λίστα περιορισμού των υποψήφιων με 15 υποψήφιους πελάτες

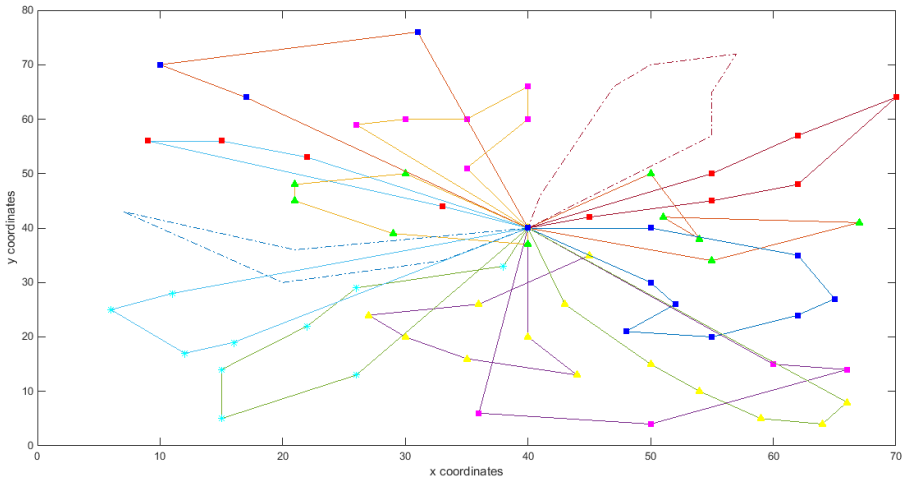
Πίνακας 5.5 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par7 για τυχαιοποίηση ανά 15 υποψήφιους.

Par7	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	1177,108	1176,593	1184,408
2	1134,212	**1128,905**	1117,554
3	1147,499	1156,427	1133,034
4	1194,196	1159,724	1172,99
5	1143,613	1136,01	1136,834
6	1178,539	1148,983	1179,293
7	1185,093	1187,75	1171,447
8	1161,417	1170,262	**1107,955**
9	1142,087	1152,708	1189,499
10	**1132,008**	1184,849	1188,172
Μ.Ο.	1159,577	1160,221	1158,119

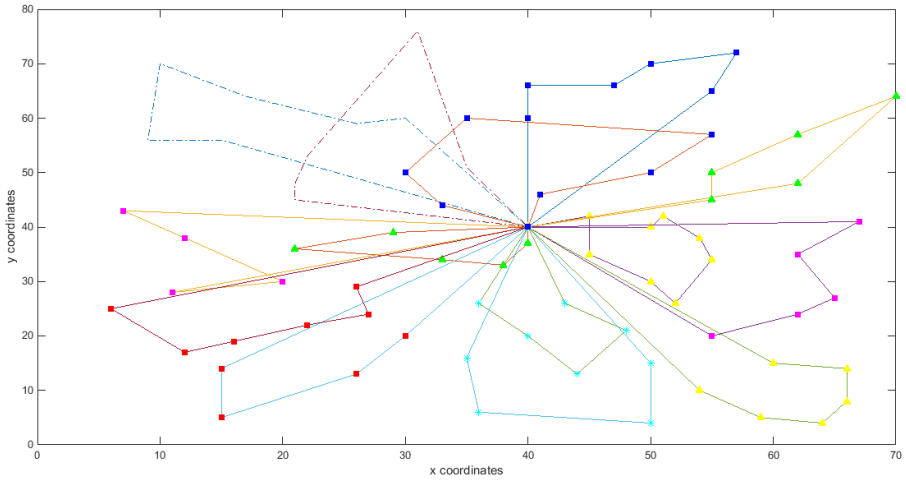
Εικόνα 5.13 Βέλτιστη διαδρομή για παράδειγμα par7 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 15 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=1132,0075)



Εικόνα 5.14 Βέλτιστη διαδρομή για παράδειγμα par7 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 15 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=1128,9050)



Εικόνα 5.15 Βέλτιστη διαδρομή για παράδειγμα par7 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 15 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=1107,9551)



Στην τυχαιοποίηση ανά 15,η βέλτιστη λύση COST=1107,9551 καθώς και ο βέλτιστος μέσος όρος των 10 τρεξιμάτων βρέθηκε για τον μέγιστο αριθμό επαναλήψεων. Και στην περίπτωση των 76 κόμβων επιβεβαιώνεται το συμπέρασμα ότι όσο πιο μειωμένο είναι το εύρος της τυχαιοποίησης τόσο καλύτερα είναι τα αποτελέσματα μας. Το καλύτερο αποτέλεσμα βρέθηκε για τον μέγιστο αριθμό επαναλήψεων στην λίστα με τους 5 υποψήφιους με COST=1089,9895.

Γ. Παράδειγμα par8 με 101 κόμβους

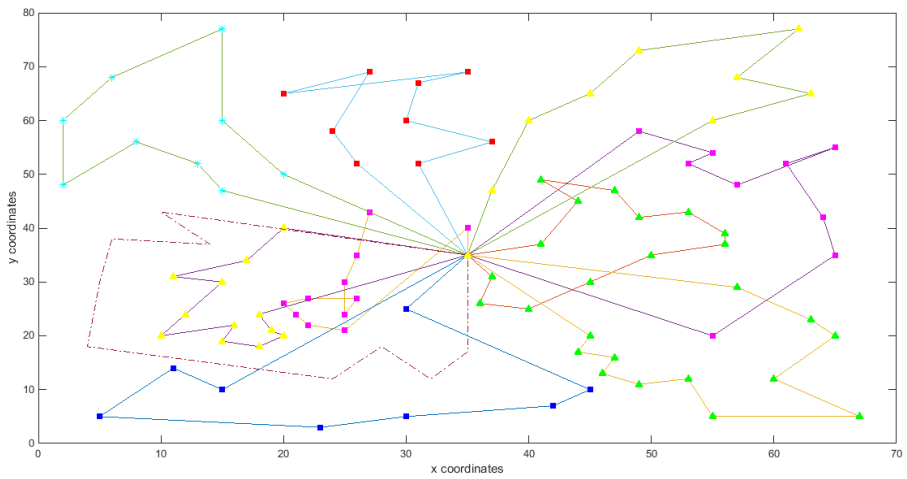
Στο **Παράδειγμα par8** πραγματοποιήσαμε τις τρεις κατηγορίες τρεξιμάτων που εκτελέσαμε και στις προηγούμενες περιπτώσεις (Num_GRASP=100 & Num_local=20.000, Num_GRASP=100 & Num_local=200.000 και Num_GRASP=1.000 & Num_local=200.000) για μέγεθος της λίστας περιορισμού των υποψηφίων 2, 4, 5 και 10.

Λίστα περιορισμού των υποψήφιων με 2 υποψήφιους πελάτες

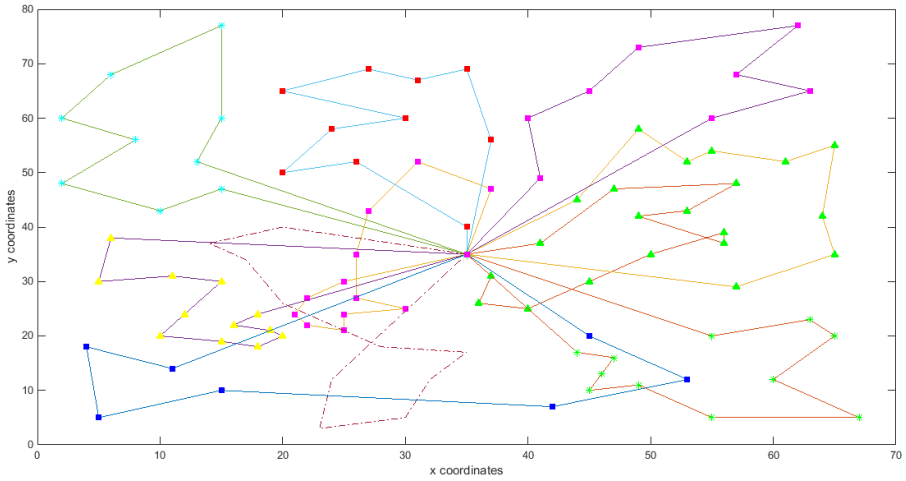
Πίνακας 5.6 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par8 για τυχαιοποίηση ανά 2 υποψήφιους.

Par8	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100 Num_local=20000	Num_GRASP=100 Num_local=200000	Num_GRASP=100 Num_local=200000
1	1126,669	1111,758	1145,222
2	1086,547	1093,498	1102,531
3	**1053,243**	1068,334	1071,142
4	1139,415	1113,751	1059,399
5	1101,611	**1067,226**	1084,244
6	1135,746	1130,842	1064,191
7	1088,421	1104,259	1086,875
8	1116,904	1111,411	1142,137
9	1099,656	1094,047	1104,228
10	1142,522	1076,298	**1055,795**
M.O.	1109,073	1097,142	1091,576

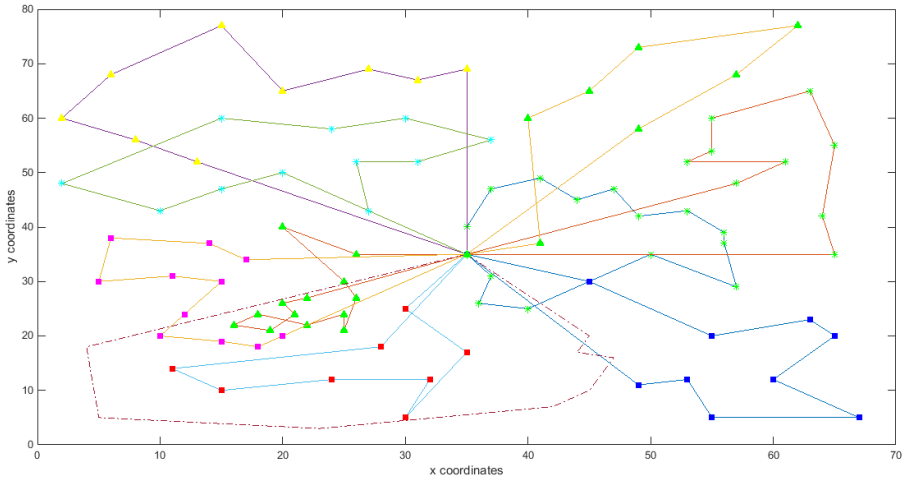
Εικόνα 5.16 Βέλτιστη διαδρομή για παράδειγμα par8 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 3 με COST=1053,2430)



Εικόνα 5.17 Βέλτιστη διαδρομή για παράδειγμα par8 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 5 με COST=1067,2257)



Εικόνα 5.18 Βέλτιστη διαδρομή για παράδειγμα par8 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=1055,7952)



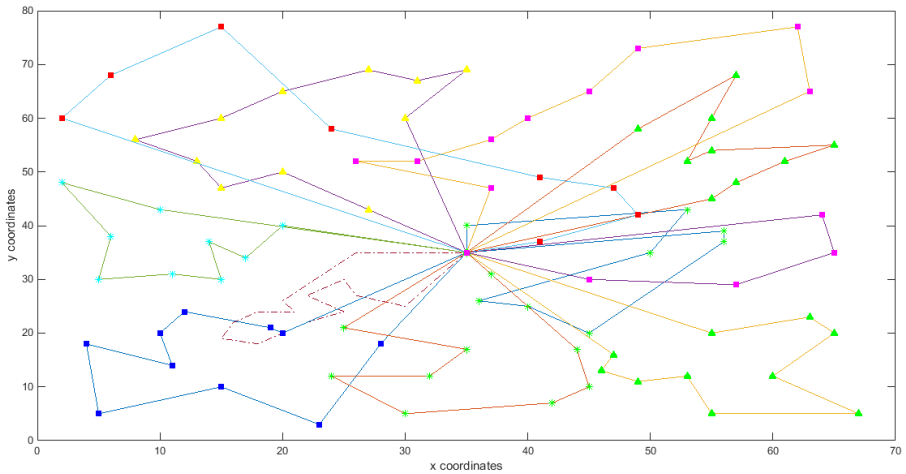
Στην περίπτωση που η λίστα περιλαμβάνει 2 υποψήφιους η βέλτιστη λύση (COST=1055,7952) καθώς και βέλτιστος μέσος όρος των 10 τρεξιμάτων (M.O.=1091,5764) βρέθηκαν για Num_GRASP=1.000 & Num_local=200.000.

Λίστα περιορισμού των υποψηφίων με 4 υποψήφιους πελάτες

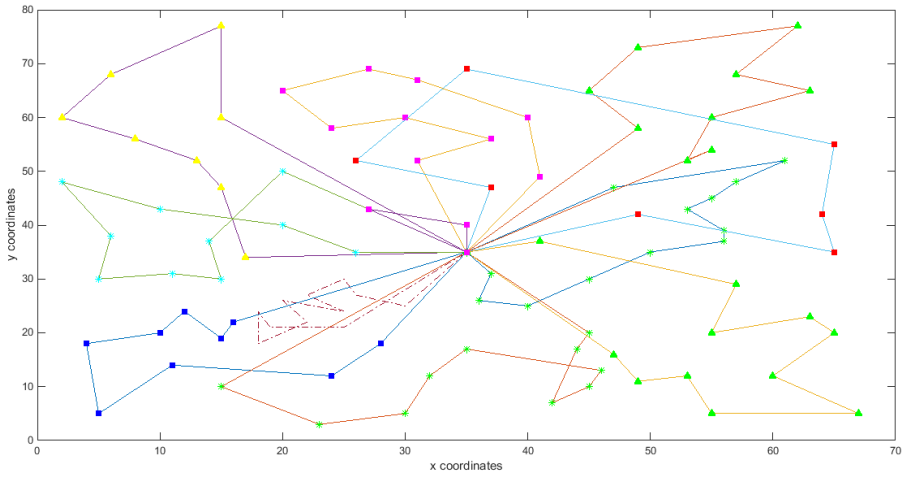
Πίνακας 5.7 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par8 για τυχαιοποίηση ανά 4 υποψήφιους.

Par8	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1000, Num_local=200.000
1	1136,565	1128,537	1101,195
2	1128,36	1125,936	1147,207
3	1145,417	1144,516	**1063,351**
4	1132,178	1132,363	1115,181
5	1132,895	1143,582	1084,244
6	1125,61	1129,161	1142,918
7	1142,138	1134,794	1124,854
8	**1076,39**	**1099,233**	1135,319
9	1137,13	1120,253	1146,466
10	1133,443	1126,941	1102,552
M.O.	1129,012	1128,532	1116,329

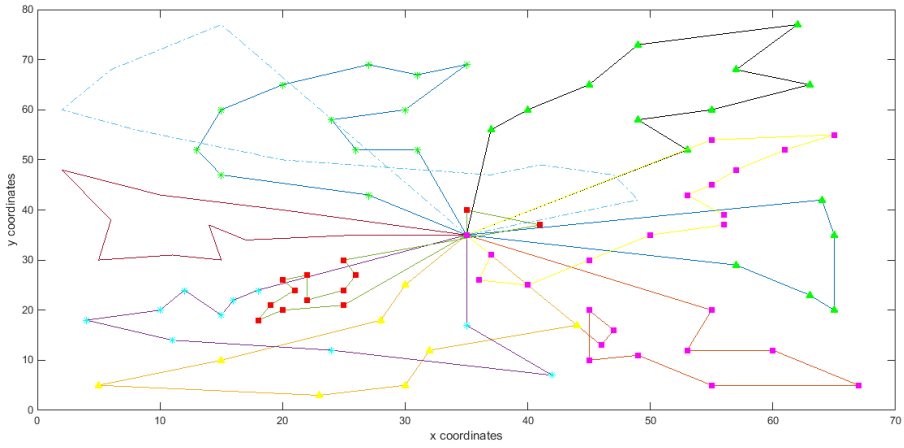
Εικόνα 5.19 Βέλτιστη διαδρομή για παράδειγμα par8 με Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1076,3896)



Εικόνα 5.20 Βέλτιστη διαδρομή για παράδειγμα par8 με Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1099,2326)



Εικόνα 5.21 Βέλτιστη διαδρομή για παράδειγμα par8 με Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1063,3505)



Και στο συγκεκριμένο παράδειγμα μπορούμε να διακρίνουμε πως όσο αυξάνεται ο αριθμός των επαναλήψεων τόσο βελτιώνεται η ποιότητα των αποτελεσμάτων μας. Η βέλτιστη λύση ισούται με COST=1063,3505 και βρέθηκε για Num_GRASP=1.000 & Num_local=200.000 επαναλήψεις.

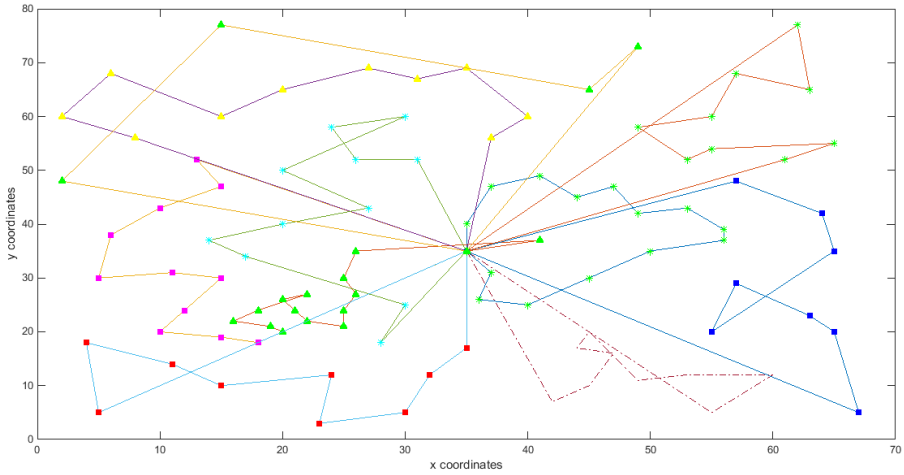
Λίστα περιορισμού των υποψηφίων με 5 και με 10 υποψήφιους πελάτες

Στην περίπτωση που η λίστα των υποψήφιων περιλαμβάνει 5 ή 10 πελάτες αντίστοιχα δεν μπόρεσε να βρεθεί καλύτερη λύση από αυτή που βρέθηκε με τον αλγόριθμο του πλησιέστερου γείτονα για καμία κατηγορία επαναλήψεων όσες φορές και αν υλοποιήσαμε τον αλγόριθμο.

Πίνακας 5.8 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par8 για τυχαιοποίηση ανά 5 & 10 υποψήφιους

Par8	Συνδυασμοί Num_GRASP και Num_local		
Νο τρεξίματος	Num_GRASP=100 Num_local=20.000	Num_GRASP=100 Num_local=200.000	Num_GRASP=1.000 Num_local=200.000
1	1147,207	1147,207	1147,207
2	1147,207	1147,207	1147,207
3	1147,207	1147,207	1147,207
4	1147,207	1147,207	1147,207
5	1147,207	1147,207	1147,207
6	1147,207	1147,207	1147,207
7	1147,207	1147,207	1147,207
8	1147,207	1147,207	1147,207
9	1147,207	1147,207	1147,207
10	1147,207	1147,207	1147,207
Μ.Ο.	1147,207	1147,207	1147,207

Εικόνα 5.22 Βέλτιστη διαδρομή για παράδειγμα par8 για όλες τις κατηγορίες επαναλήψεων και τυχαιοποίηση ανά 5 & 10 υποψήφιους (COST= 1147,2069)



Συνολικά, στην περίπτωση των 101 κόμβων και συγκεκριμένα για το par8 για την λίστα με τους 2 υποψήφιους και μέγιστο αριθμό επαναλήψεων βρήκαμε το βέλτιστο αποτέλεσμα. Στην περίπτωση όμως που λίστα περιορισμού των υποψήφιων (Restricted CandidateList) περιλάμβανε 5 και 10 υποψήφιους αντίστοιχα δεν καταφέραμε να βελτιώσουμε τη λύση που βρέθηκε με τον αλγόριθμο του πλησιέστερου γείτονα. Αυτό οφείλεται κυρίως στην τυχαιότητα του αλγορίθμου η οποία εντοπίζεται και κατά την κατασκευή της αρχικής εφικτής λύσης του GRASP αλλά και κατά τη φάση της τοπικής αναζήτησης. Η αστοχία μας αυτή είναι πιθανό να επηρεάστηκε και από το αυξημένο πλήθος πελατών.

Δ. Παράδειγμα par12 με 101 κόμβους

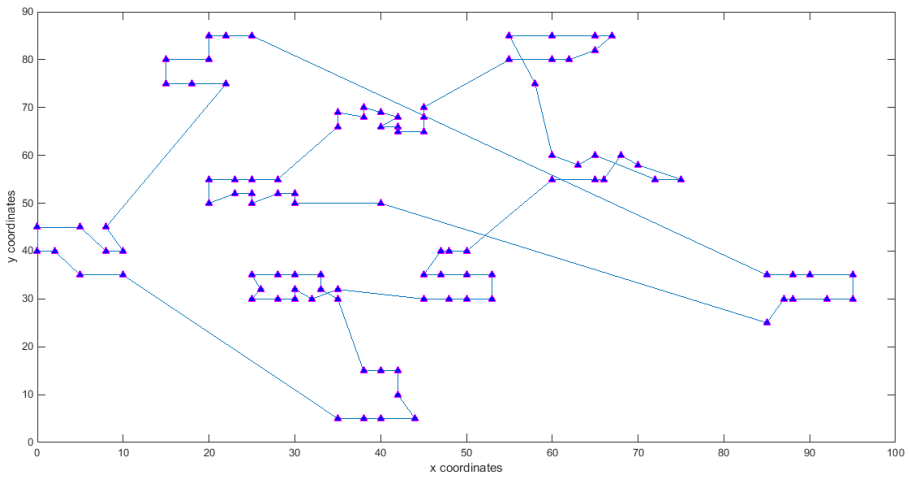
Στο **Παράδειγμα par12** για τις ίδιες κατηγορίες επαναλήψεων και τις ίδιες λίστες υποψήφιων με το **Παράδειγμα par8** (2, 4, 5 και 10 πελάτες) βρέθηκαν τα ακόλουθα αποτελέσματα.

Λίστα περιορισμού των υποψήφιων με 2 υποψήφιους πελάτες

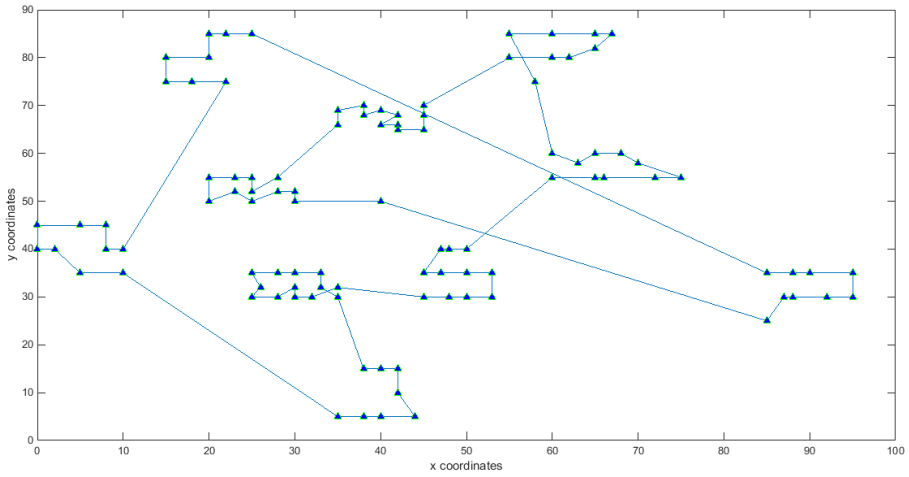
Πίνακας 5.8 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par12 για τυχαιοποίηση ανά 2 υποψήφιους

Par12	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	618,8017	611,2853	622,3228
2	620,1147	**602,605**	**599,772**
3	609,1365	605,9421	601,7131
4	615,742	611,3012	603,9244
5	615,5208	605,3552	613,6811
6	622,1234	622,2341	604,9965
7	**603,8434**	614,536	625,2581
8	617,4873	610,8534	609,9079
9	621,7401	615,6673	609,5368
10	612,0469	609,7732	605,8359
M.O.	615,6557	610,9553	609,6949

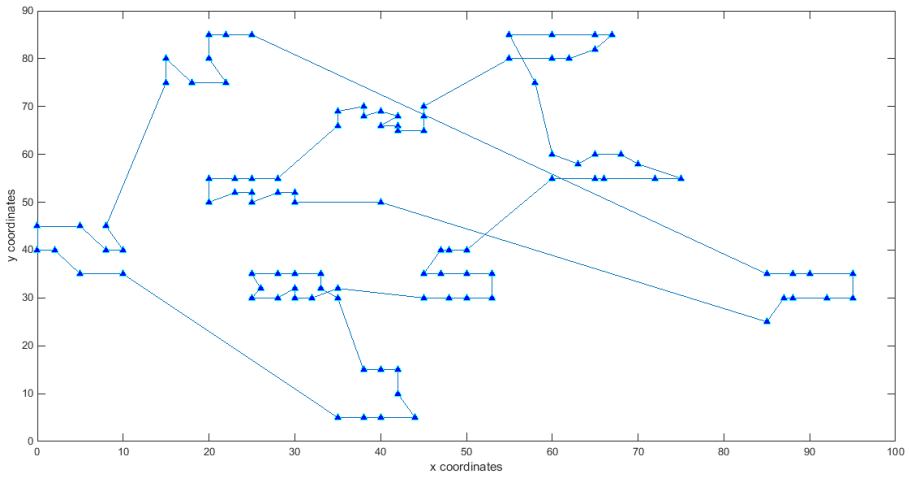
Εικόνα 5.23 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=100 & Num_local=20.000 για τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=603,8434)



Εικόνα 5.24 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=100 & Num_local=200.000 για τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 2 με COST=602,605)



Εικόνα 5.25 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=1.000 & Num_local=200.000 για τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 2 με COST=599,772)



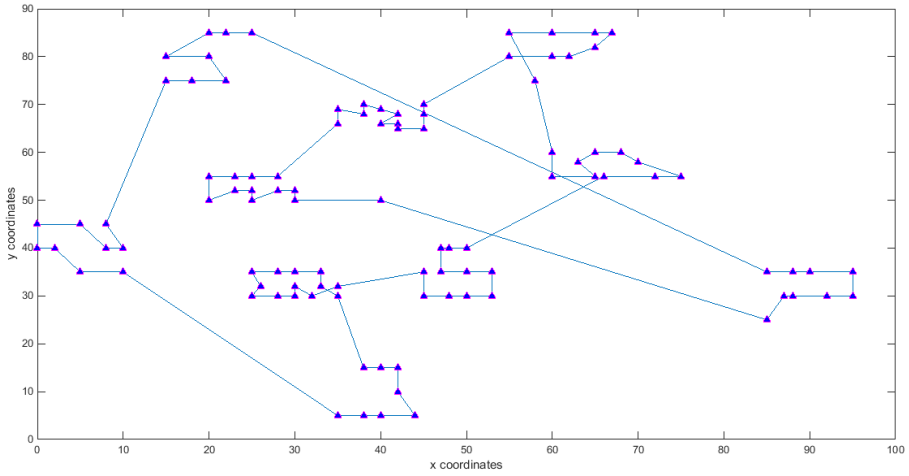
Στην λίστα με τους 2 υποψήφιους, για b=1.000 & i=200.000 βρήκαμε τη βέλτιστη διαδρομή με κόστος COST=599,772 καθώς και τον βέλτιστο μέσο όρο (M.O.=609,6949).

Λίστα περιορισμού των υποψήφιων με 4 υποψήφιους πελάτες

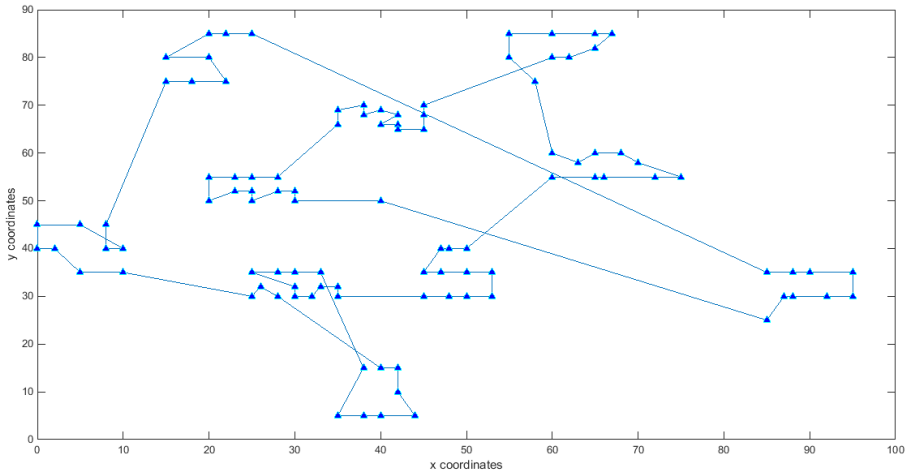
Πίνακας 5.9 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par12 για τυχαιοποίηση ανά 4 υποψήφιους

Par12	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	616,7966	616,425	605,7824
2	**609,7732**	625,1862	607,2189
3	615,3264	614,807	612,3994
4	624,6153	617,8161	624,9552
5	625,3939	605,7919	615,4899
6	621,853	**605,1444**	621,2073
7	619,5554	617,757	**602,1536**
8	627,1258	622,633	606,206
9	624,5483	612,9437	615,9923
10	619,8917	614,4625	611,5557
M.O.	620,488	615,2967	612,2961

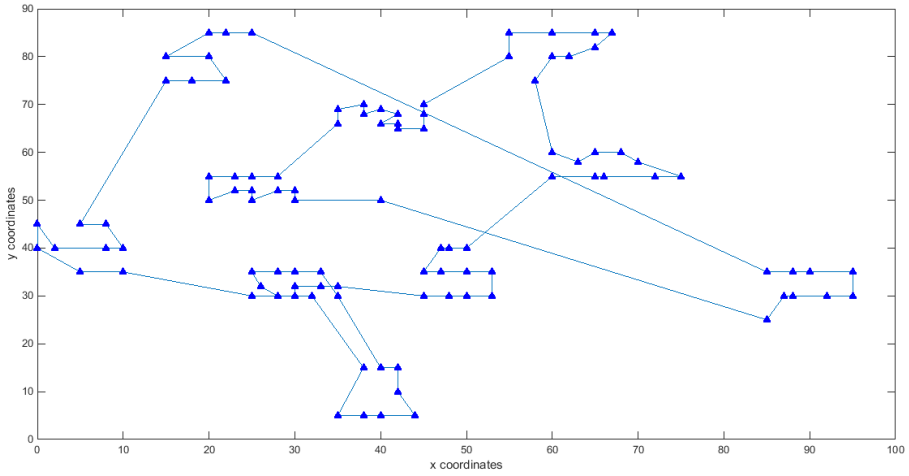
Εικόνα 5.26 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=100 & Num_local=20.000 για τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 2 με COST=609,7732)



Εικόνα 5.27 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=100 & Num_local=200.000 για τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 6 με COST=605,1444)



Εικόνα 5.28 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=1.000 & Num_local=200.000 για τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=602,1536)



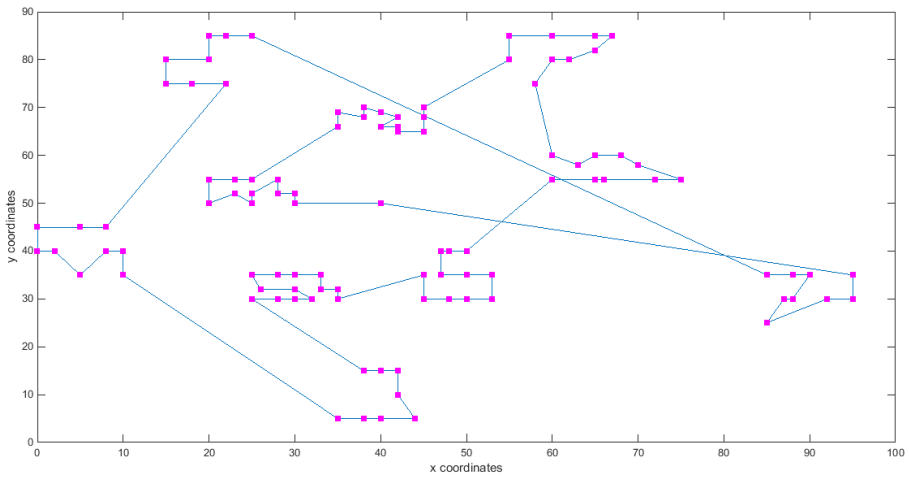
Και στην συγκεκριμένη περίπτωση επιβεβαιώνεται πως όταν αυξάνεται ο αριθμός των επαναλήψεων βελτιώνεται η ποιότητα των αποτελεσμάτων με το καλύτερο αποτέλεσμα να εμφανίζεται στον μέγιστο αριθμό επαναλήψεων (COST=602,1536).

Λίστα περιορισμού των υποψηφίων με 5 υποψήφιους πελάτες

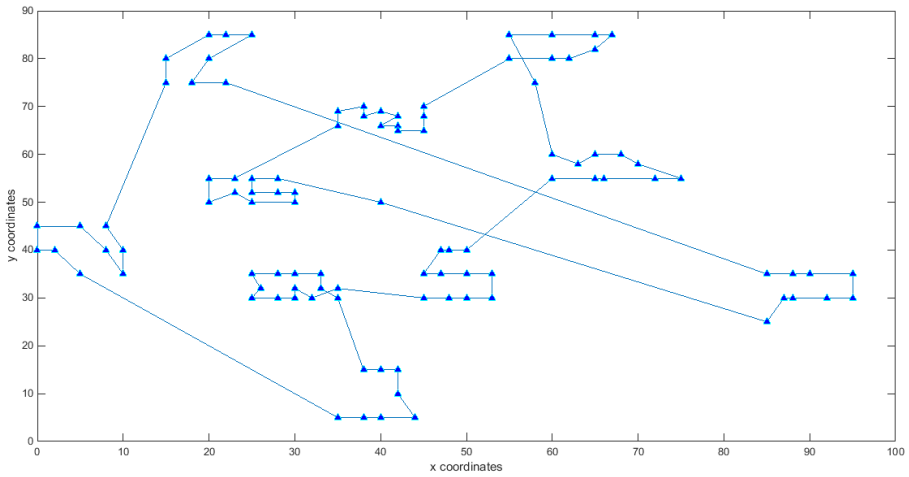
Πίνακας 5.10 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par12 για τυχαιοποίηση ανά 5 υποψήφιους.

Par12	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	622,062	616,8301	613,0172
2	**612,7064**	627,1258	627,1258
3	627,1258	619,8371	**606,4632**
4	626,7394	627,1258	627,1258
5	626,2317	**606,5706**	620,7397
6	624,4285	627,1258	613,0144
7	619,1836	615,8794	619,8147
8	617,8612	627,1258	627,1258
9	620,2641	627,1258	618,5213
10	626,8337	627,1258	627,1258
M.O.	622,3436	622,1872	620,0074

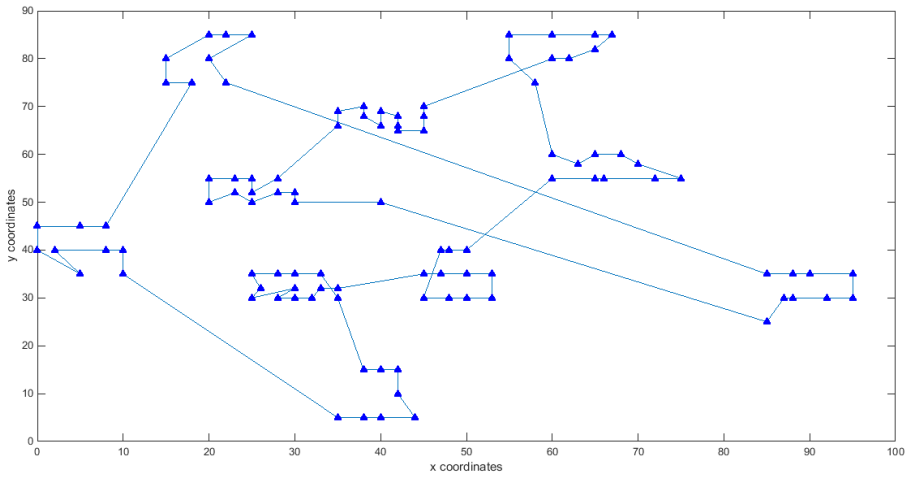
Εικόνα 5.29 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=100 & Num_local=20.000 για τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 2 με COST=612,7064)



Εικόνα 5.30 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=100 & Num_local=200.000 για τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 5 με COST=606,5706)



Εικόνα 5.31 Βέλτιστη διαδρομή για παράδειγμα par12 για Num_GRASP=1.000 & Num_local=200.000 για τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 5 με COST=606,5706)



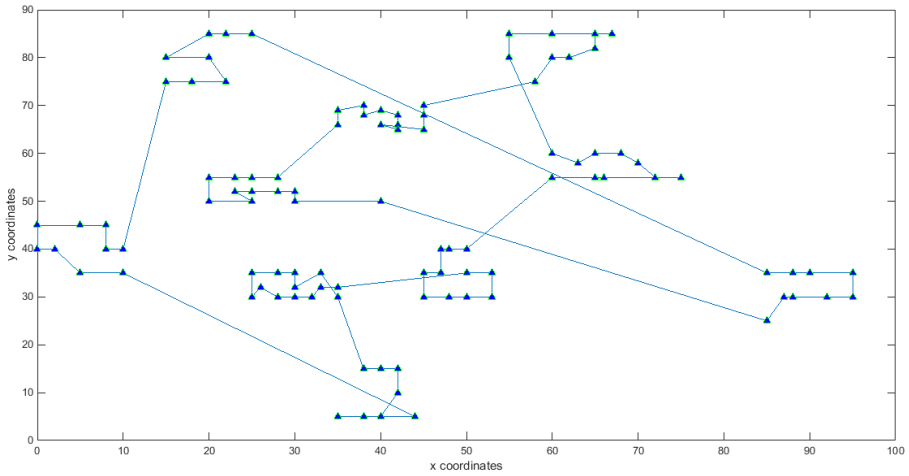
Και εδώ μπορούμε να παρατηρήσουμε ότι σε πολλές εκτελέσεις του αλγορίθμου η λύση που βρέθηκε ήταν αυτή του πλησιέστερου γείτονα. Στο συγκεκριμένο παράδειγμα βέβαια καταφέραμε να εντοπίσουμε και καλύτερες λύσεις για όλες τις κατηγορίες των επαναλήψεων. Αυτό προφανώς οφείλεται στο ότι έχουμε διαφοροποιήσεις στις παραμέτρους του προβλήματος.

Λίστα περιορισμού των υποψηφίων με 10 υποψήφιους πελάτες

Πίνακας 5.11 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par12 για τυχαιοποίηση ανά 10 υποψήφιους

Par12	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	627,1258	627,1258	627,1258
2	627,1258	627,1258	627,1258
3	627,1258	627,1258	627,1258
4	627,1258	627,1258	627,1258
5	627,1258	627,1258	627,1258
6	627,1258	627,1258	627,1258
7	627,1258	627,1258	627,1258
8	627,1258	627,1258	627,1258
9	627,1258	627,1258	627,1258
10	627,1258	627,1258	627,1258
Μ.Ο.	627,1258	627,1258	627,1258

Εικόνα 5.32 Βέλτιστη διαδρομή για παράδειγμα par12 για όλες τις κατηγορίες επαναλήψεων και τυχαιοποίηση ανά 10 υποψήφιους (COST=627,1258)



Στο Παράδειγμα 12, βέλτιστη λύση βρέθηκε για το μικρότερο μέγεθος της λίστας των υποψηφίων και τον μέγιστο αριθμό επαναλήψεων (COST=599,772). Αυτό είναι λογικό διότι όσο αυξάνεται το μέγεθος της λίστας τόσο ποιο τυχαίες γίνονται οι αλλαγές που πραγματοποιεί ο αλγόριθμος στις λύσεις. Αυτό μπορεί να αιτιολογήσει και το γεγονός ότι για 10 πελάτες δεν βρέθηκε καλύτερη λύση από αυτή του αλγορίθμου της απληστίας.

Ε. Παράδειγμα par14 με 101 κόμβους

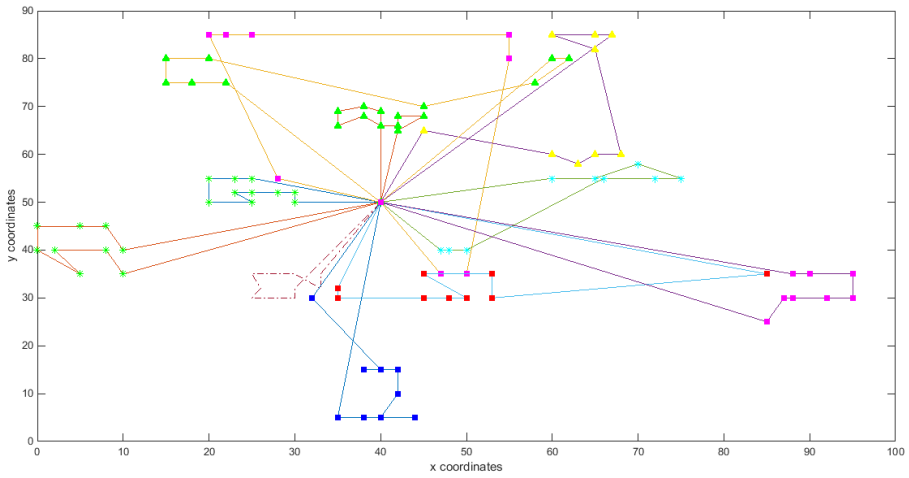
Για αριθμό επαναλήψεων Num_GRASP=100 & Num_local=20.000, για Num_GRASP=100 & Num_local=200.000 και τέλος για Num_GRASP=1.000 & Num_local=200.000 και για μέγεθος της λίστας 2, 4 ,5, 10 έχουμε τα ακόλουθα αποτελέσματα.

Λίστα περιορισμού των υποψηφίων με 2 υποψήφιους πελάτες

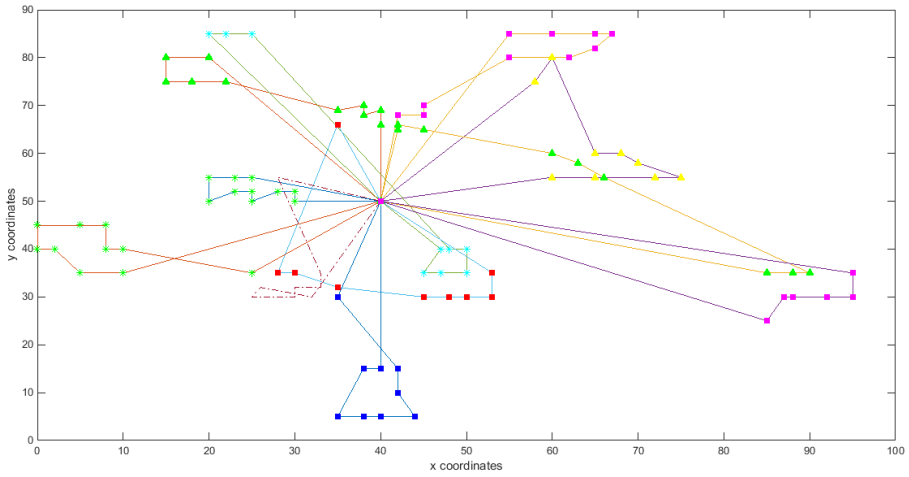
Πίνακας 5.11 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par14 για τυχαιοποίηση ανά 2 υποψήφιους

Par14	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	1191,43	1170,288	1170,422
2	1202,606	1181,691	1153,803
3	1195,224	1213,353	**1020,455**
4	1171,39	1151,362	1190,507
5	1199,611	1122,177	1161,485
6	1165,357	1162,635	1192,008
7	**1144,107**	1189,282	1142,352
8	1154,079	1156,197	1140,427
9	1158,992	**1112,89**	1149,618
10	1162,56	1146,401	1147,879
M.O.	1174,536	1160,628	1144,282

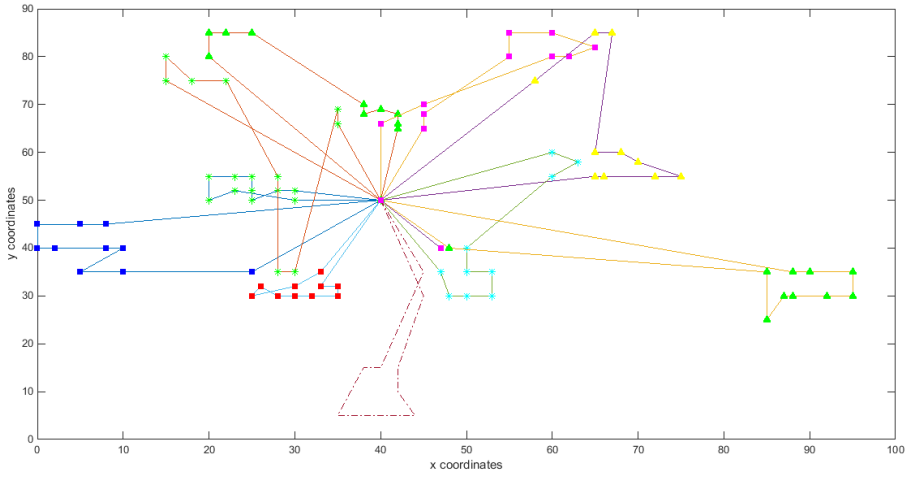
Εικόνα 5.33 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 7 με COST=1144,107)



Εικόνα 5.34 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 9 με COST=1112,89)



Εικόνα 5.35 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 3 με COST=1020,455)



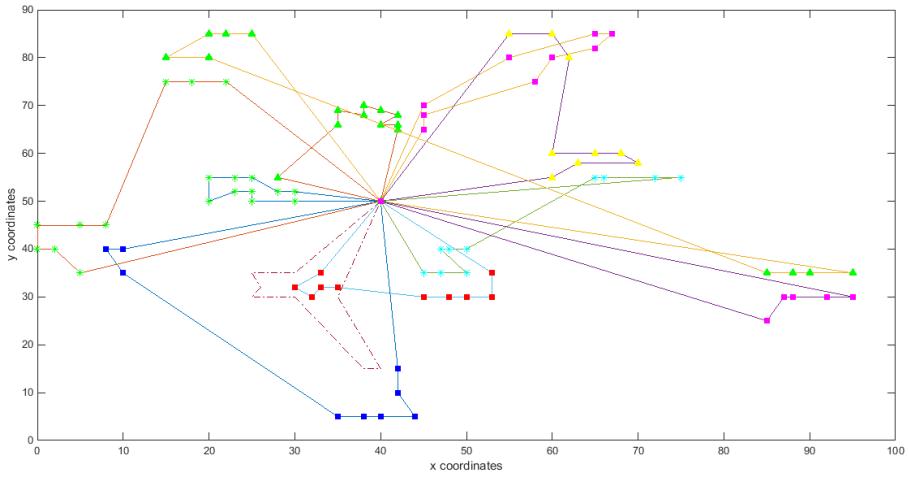
Βέλτιστη διαδρομή για τυχαιοποίηση ανά 2 υποψήφιους πελάτες καθώς και βέλτιστος μέσος όρος των 10 καλύτερων λύσεων βρέθηκε για Num_GRASP=1.000 & Num_local=200.000 (COST=1020,455, M.O.=1144,282).

Λίστα περιορισμού των υποψηφίων με 4 υποψήφιους πελάτες

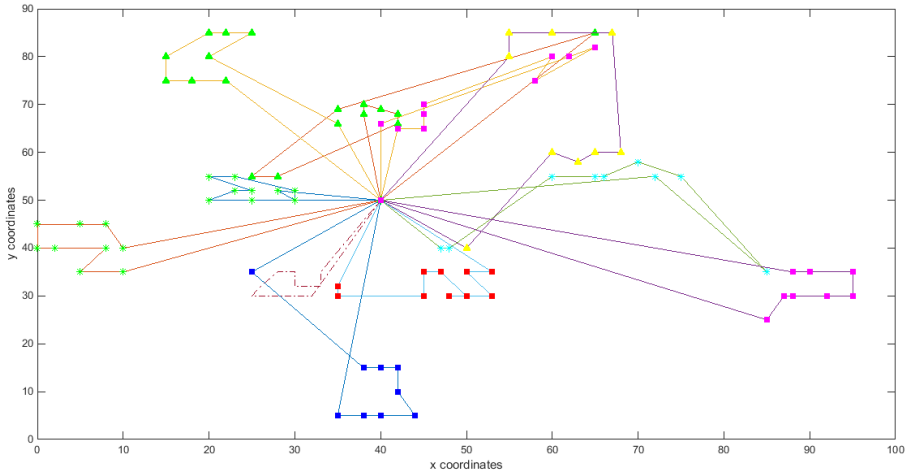
Πίνακας 5.12 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par14 για τυχαιοποίηση ανά 4 υποψήφιους

Par14	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	**1156,418**	1183,791	1198,257
2	1179,107	1183,169	1176,319
3	1164,252	1144,279	**1114,902**
4	1140,684	1181,025	1180,613
5	1205,087	1207,783	1164,818
6	1174,262	1171,94	1199,396
7	1212,057	1180,3	1187,8
8	1179,214	**1153,53**	1172,045
9	1167,196	1135,744	1197,118
10	1173,528	1203,003	1140,434
M.O.	1175,18	1174,456	1173,17

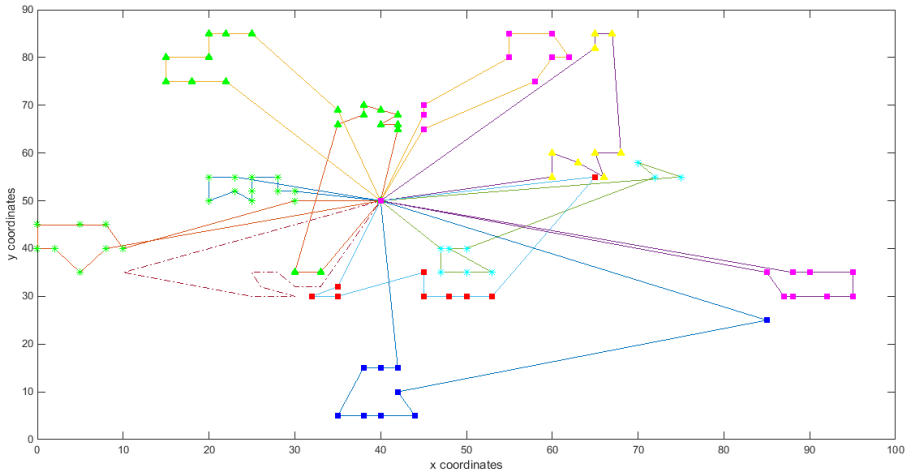
Εικόνα 5.37 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 1 με COST=1156,418)



Εικόνα 5.38 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1153,53)



Εικόνα 5.39 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 4 υποψήφιους (Βέλτιστη εκτέλεση No 3 COST=1114,902)



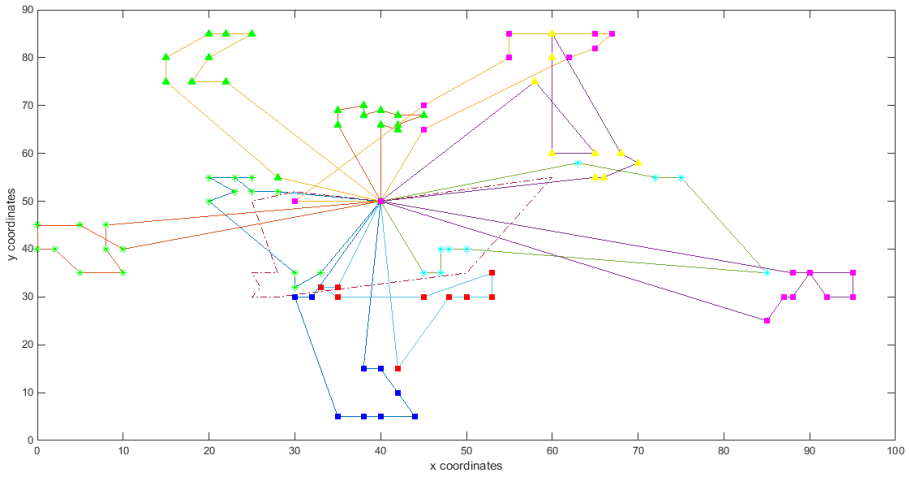
Για τον μέγιστο αριθμό επαναλήψεων έχουμε την καλύτερη διαδρομή (COST=1114,902) αλλά και τα καλυτέρα ποιοτικά αποτελέσματα (Μ.Ο.=1173,17)

Λίστα περιορισμού των υποψηφίων με 5 υποψήφιους πελάτες

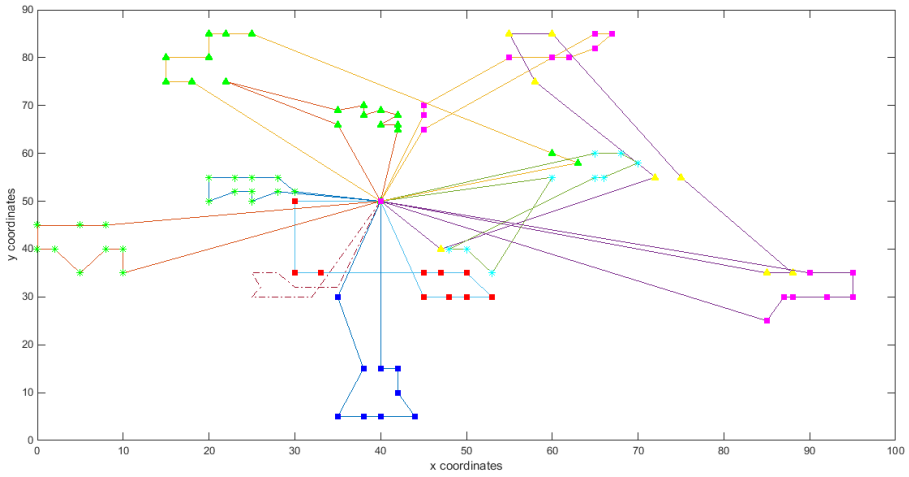
Πίνακας 5.13 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par14 για τυχαιοποίηση ανά 5 υποψήφιους

Par14	Συνδυασμοί Num_GRASP και Num_local		
Νο τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	1198,585	1169,286	1144,477
2	1176,953	1164,924	1145,09
3	1214,775	1201,26	**1115,007**
4	1210,325	1181,504	1142,617
5	1195,3	**1123,241**	1131,072
6	1207,769	1181,193	1186,581
7	1192,074	1175,256	1206,451
8	1205,548	1164,924	1171,153
9	**1158,058**	1137,86	1201,26
10	1181,402	1199,34	1135,994
Μ.Ο.	1194,079	1169,879	1157,97

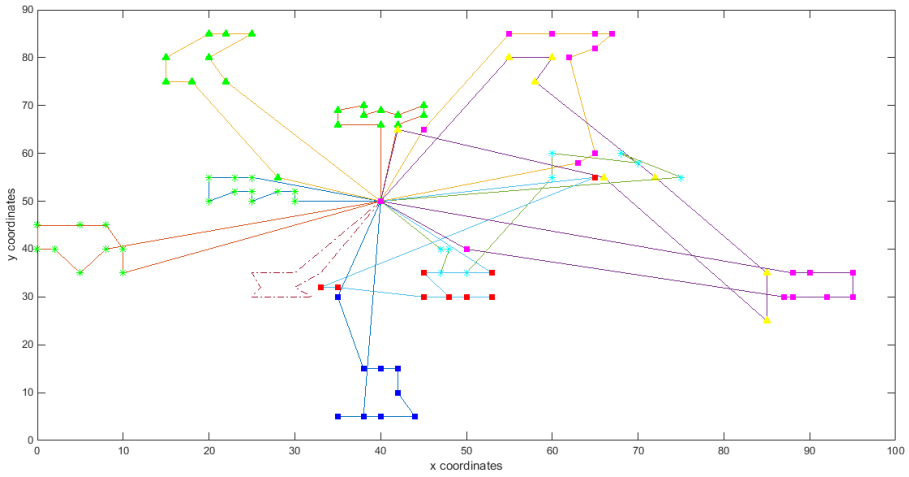
Εικόνα 5.40 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 9 με COST=1158,058)



Εικόνα 5.41 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No 9 με COST=1123,241)



Εικόνα 5.42 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση No 3 COST=1115,007)



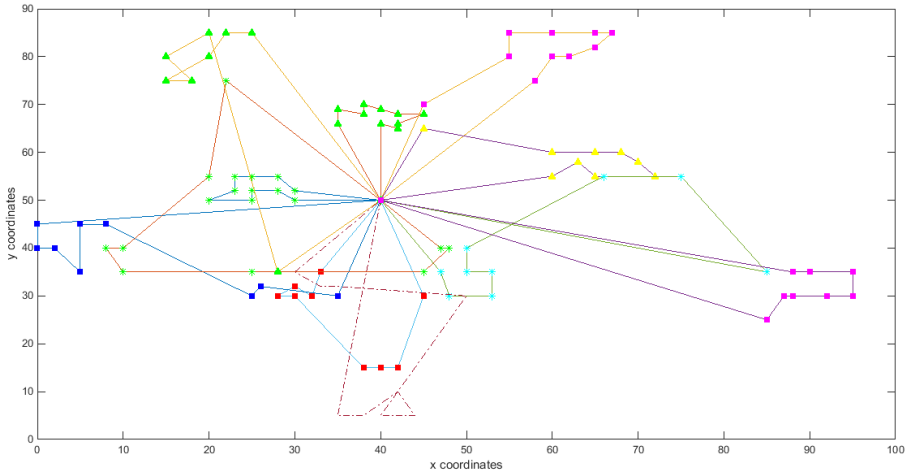
Η βέλτιστη διαδρομή για το συγκεκριμένο παράδειγμα όπου το μέγεθος της λίστας των υποψηφίων είναι 5 εντοπίστηκε για Num_GRASP=1.000 & Num_local=200.000 (COST=1115,007).

Λίστα περιορισμού των υποψήφιων με 10 υποψήφιους πελάτες

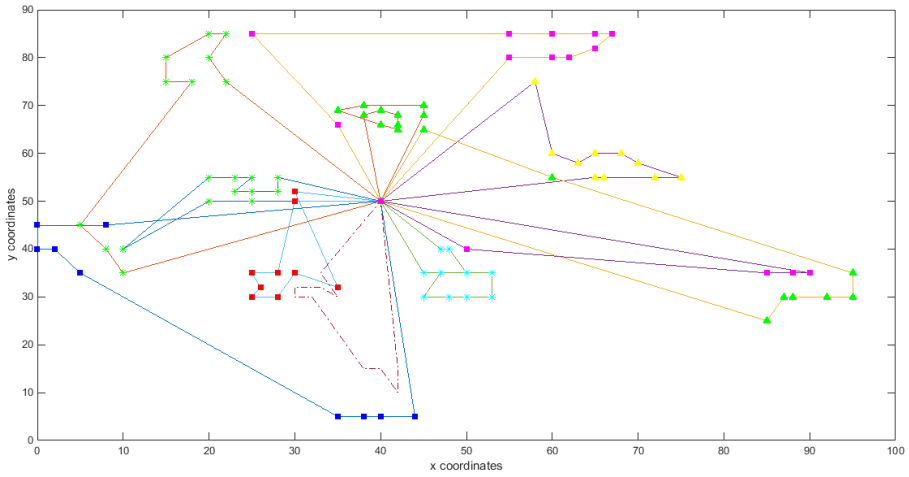
Πίνακας 5.14 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par14 για τυχαιοποίηση ανά 10 υποψήφιους

Par14	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	**1156,827**	1197,384	1138,589
2	1214,277	1187,616	1206,877
3	1213,519	1214,775	1176,721
4	1194,363	1209,334	1182,311
5	1187,058	1207,228	1202,41
6	1203,144	1211,036	1177,085
7	1213,842	1155,263	1208,426
8	1205,215	**1148,131**	1190,89
9	1189,378	1205,519	1214,775
10	1204,433	1202,252	**1041,871**
Μ.Ο.	1198,206	1193,854	1173,996

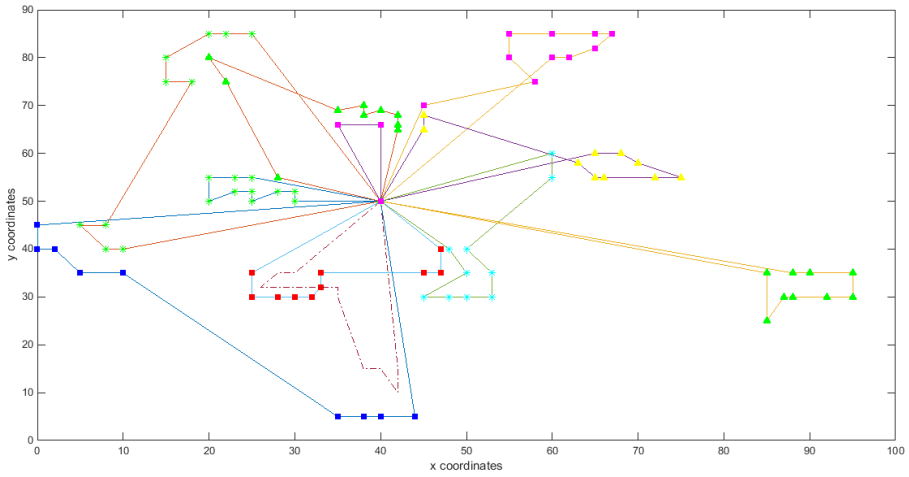
Εικόνα 5.43 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 10 υποψήφιους (Βέλτιστη εκτέλεση η No 1 με COST=1156,827)



Εικόνα 5.44 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 5 υποψήφιους (Βέλτιστη εκτέλεση η No8 με COST=1148,131)



Εικόνα 5.44 Βέλτιστη διαδρομή για παράδειγμα par14 για Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 10 υποψήφιους (Βέλτιστη εκτέλεση η No10 με COST=1041,871)



Στο Παράδειγμα 14 (101 κόμβοι) καταφέραμε να βελτιώσουμε τη λύση του πλησιέστερου γείτονα ανεξάρτητα από το μέγεθος της λίστας. Βέβαια, όπως ήταν αναμενόμενο το καλύτερο αποτέλεσμα το βρήκαμε όταν η λίστα περιορισμού των υποψηφίων είχε μέγεθος 2. Το αμέσως καλύτερο όμως, βρέθηκε για τους 10 υποψήφιους πελάτες αντί για τους 4 ή τους 5.Οπως έχουμε αναφέρει πολλές φορές ο αλγόριθμος μας κατά ένα μέρος βασίζεται στην τυχαιότητα γεγονός που εξηγεί τη συγκεκριμένη συμπεριφορά.

Ζ. Παράδειγμα par9 με 151 κόμβους

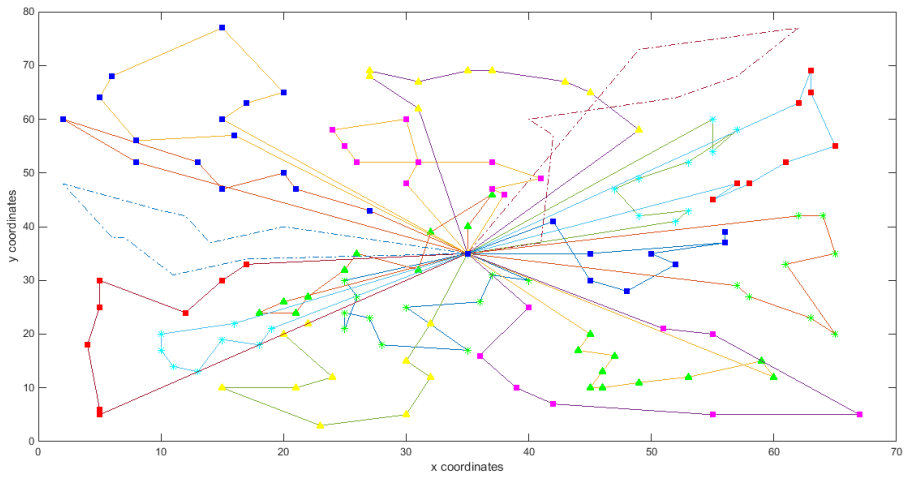
Λίστα περιορισμού των υποψηφίων με 2 υποψήφιους πελάτες

Στο πρόβλημα των 151 κόμβων το μέγεθος της λίστας που επιλέξαμε να εξετάσουμε είναι 2, 5 και 10 υποψήφιοι. Οι 3 κατηγορίες επαναλήψεων που εφαρμόσαμε είναι ίδιες με τις παραπάνω περιπτώσεις.

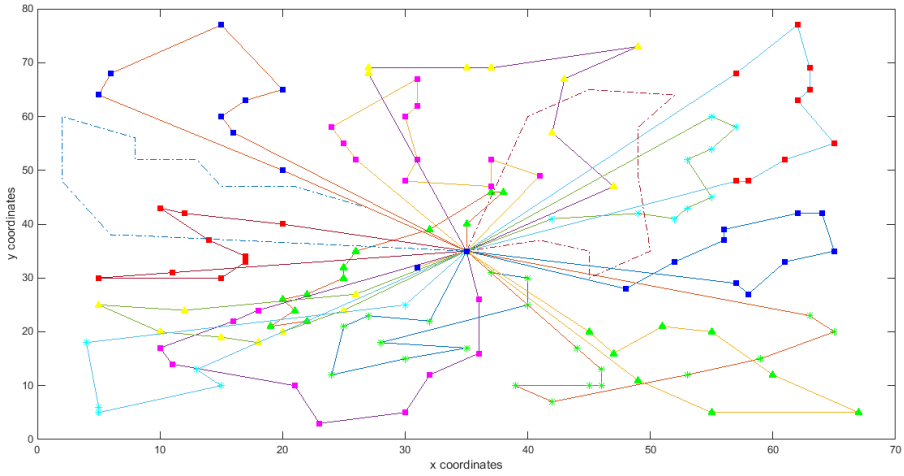
Πίνακας 5.15 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par9 για τυχαιοποίηση ανά 2 υποψήφιους

Par9	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	1529,444	1503,454	1513,935
2	1525,238	1499,747	1522,308
3	1525,553	1473,257	1506,025
4	1530,314	1499,577	1508,004
5	1513,901	1517,629	1483,041
6	1500,134	1532,754	1527,742
7	1529,678	1520,38	1495,158
8	1532,359	**1456,17**	**1454,237**
9	1498,886	1500,586	1505,875
10	**1495,812**	1499,652	1483,116
Μ.Ο.	1518,132	1500,321	1499,944

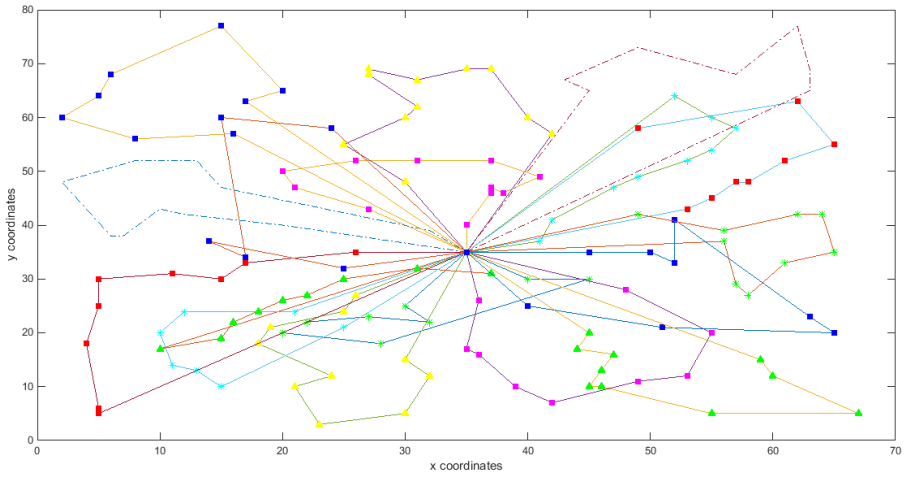
Εικόνα 5.46 Βέλτιστη διαδρομή για παράδειγμα par9 για Num_GRASP=100 & Num_local=20.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 10 με COST=1495,812)



Εικόνα 5.47 Βέλτιστη διαδρομή για παράδειγμα par9 για Num_GRASP=100 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1456,17)



Εικόνα 5.48 Βέλτιστη διαδρομή για παράδειγμα par9 για Num_GRASP=1.000 & Num_local=200.000 και τυχαιοποίηση ανά 2 υποψήφιους (Βέλτιστη εκτέλεση η No 8 με COST=1454,237)



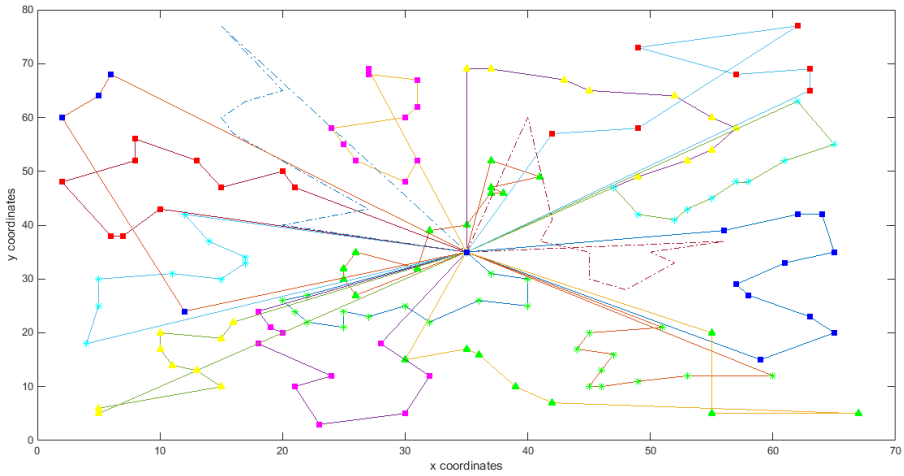
Βέλτιστη διαδρομή και βέλτιστος μέσος όρος για τη λίστα μεγέθους 2 υποψήφιων στο παράδειγμα par9 βρέθηκε για τον μέγιστο αριθμό επαναλήψεων (COST=1454,237, Μ.Ο.=1499,944).

Λίστα περιορισμού των υποψήφιων με 5 και 10 υποψήφιους πελάτες

Πίνακας 5.16 Αποτελέσματα 10 τρεξιμάτων με το παράδειγμα par9 για τυχαιοποίηση ανά 5 και 10 υποψήφιους

Par9	Συνδυασμοί Num_GRASP και Num_local		
No τρεξίματος	Num_GRASP=100, Num_local=20.000	Num_GRASP=100, Num_local=200.000	Num_GRASP=1.000, Num_local=200.000
1	1534,669	1534,669	1534,669
2	1534,669	1534,669	1534,669
3	1534,669	1534,669	1534,669
4	1534,669	1534,669	1534,669
5	1534,669	1534,669	1534,669
6	1534,669	1534,669	1534,669
7	1534,669	1534,669	1534,669
8	1534,669	1534,669	1534,669
9	1534,669	1534,669	1534,669
10	1534,669	1534,669	1534,669
Μ.Ο.	1534,669	1534,669	1534,669

Εικόνα 5.49 Βέλτιστη διαδρομή για παράδειγμα par9 για όλες τις κατηγορίες επαναλήψεων και τυχαιοποίηση ανά 5 και 10 υποψήφιους (Βέλτιστη εκτέλεση COST=1534,669)



Λόγω του αυξημένου πλήθους των κόμβων αλλά και του μεγάλου μεγέθους της λίστας των υποψήφιων (5 και 10) δεν καταφέραμε να βελτιώσουμε την λύση που βρήκαμε με την εφαρμογή του αλγορίθμου του πλησιέστερου γείτονα.

Για μέγεθος της λίστας περιορισμού των υποψηφίων ίσο με 2 βελτιώνουμε αρκετά το κόστος διαδρομής εφόσον μειώνεται το ποσοστό της τυχαιότητας και αυξάνεται το ποσοστό της απληστίας.

Συμπεράσματα

Στην διπλωματική αυτή προσπαθήσαμε να λύσουμε μια ομάδα παραδειγμάτων με τη χρήση του αλγορίθμου της άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης για το πρόβλημα δρομολόγησης οχημάτων σε περιορισμένη απόσταση. Κατά την εφαρμογή του αλγορίθμου στα διάφορα αυτά προβλήματα παρατηρήθηκαν τα εξής

- ✓ Ο αλγόριθμος GRASP λειτουργεί καλύτερα όσο αυξάνουμε τις επαναλήψεις του.
- ✓ Για μικρό μέγεθος της λίστας περιορισμού των υποψηφίων, ανεξάρτητα από το πλήθος των κόμβων η ποιότητα των αποτελεσμάτων είναι καλύτερη.
- ✓ Στα παραδείγματα που έχουμε αυξημένο πλήθος κόμβων, δεν μπόρεσε να βρεθεί καλύτερη λύση από τον πλησιέστερο γείτονα, λόγω του μεγάλου όγκου δεδομένων, αλλά και της τυχαιότητας των αλλαγών που πραγματοποιούνται κατά την κατασκευή της αρχικής εφικτής λύσης και της τοπικής αναζήτησης.

Για την μελλοντική βελτίωση της ποιότητας των αποτελεσμάτων του αλγορίθμου GRASP θα μπορούσαμε να χρησιμοποιήσουμε διαφορετικό αλγόριθμο τοπικής αναζήτησης (για παράδειγμα 2-opt),να αυξήσουμε τον αριθμό των επαναλήψεων, ή να χρησιμοποιήσουμε έναν επιπλέον αλγόριθμο τοπικής αναζήτησης ούτος ώστε να βελτιώσουμε περισσότερο τη λύση μας.

Βιβλιογραφία

[1] Lambert, D., Cooper, M., (2000) Issues in supply chain Management, *Industrial Marketing Management*, 29, 1, 65-83.

[2] Laporte, G. (1992) The traveling salesman problem: An overview of exact and approximate algorithms, *European Journal of Operational Research*, 59, 231-247.

[3] Laptorte, G., Desrochers, M., Nobert, Y. (1984) Two exact algorithms for the distance-constrained vehicle-routing problem, *Networks*, 14, 161-172.

[4] Thomas, D., Griffin, P. (1996) Coordinated supply chain management, *European Journal of Operational Research*, 94, 1-15.

[5] Thomas, A. F., Resende, M. G. C. (1995) Greedy Randomized adaptive search procedures, Journal of Global Optimization, 6, 2, 109-133.

[6] Tlili , T., Faiz, S., Krichen, S. (2014) A Hybrid Metaheuristic for the Distance-constrained Capacitated Vehicle Routing Problem, *Procedia - Social and Behavioral Sciences*, 109, 779–783.

[7] Toth, P., Vigo, D. (2002) Models, relaxations and exact approaches for the capacitated vehicle routing problem, *Discrete Applied Mathematics*, 123, 487 – 512.

[8] Μαρινάκης, Ι., Μυγδαλάς, Α. (2008*) Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας*, Εκδόσεις "σοφία".

[9] Μαρινάκης, Ι., Μαρινάκη, Μ. (2010) *Εξελικτικοί Αλγόριθμοι και Βελτιστοποίηση Συστημάτων Μεγάλης Κλίμακας,* Σημειώσεις Μαθήματος Εξελικτικοί Αλγόριθμοι και Βελτιστοποίηση Συστημάτων Μεγάλης Κλίμακας.

[10] https://en.wikipedia.org/wiki/NP-hardness

[11] www.logistics-managment.com