ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ
ΤΟΜΕΑΣ ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ

ΑΝΑΠΤΥΞΗ ΕΞΕΛΙΚΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΣΤΟΧΑΣΤΙΚΗ ΖΗΤΗΣΗ

Διατριβή που υπεβλήθη για τη μερική ικανοποίηση των απαιτήσεων για την Απόκτηση Μεταπτυχιακού Διπλώματος

υπό

Παρασκευή Απ. Σπανού

Χανιά, 2010

Η διατριβή της Παρασκευής Απ. Σπανού εγκρίνεται

Ιωάννης Μαρινάκης _____
Λέκτορας, επιβλέπων

Νικόλαος Ματσατσίνης _____
Καθηγητής

Ευάγγελος Γρηγορούδης _____
Επίκουρος Καθηγητής

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1
Εισαγωγή

1.1 Αντικείμενο της διατριβής

Στη σύγχρονη εποχή, διαπιστώνεται ότι η διαχείριση των υλικών και των προμηθειών διαδραματίζει πρωταρχικό ρόλο στην καλή λειτουργία των επιχειρήσεων, με αποτέλεσμα να δίνεται ιδιαίτερη βαρύτητα στις διαδικασίες της διαχείρισης της Αλυσίδας Εφοδιασμού. Οι δύο κύριες κατηγορίες της Αλυσίδας Εφοδιασμού είναι: 1) σχεδιασμός της Εφοδιαστικής Αλυσίδας (Supply Chain Planning, SCP), που περιλαμβάνει τις διαδικασίες που αφορούν στην πρόβλεψη των απαιτούμενων υλικών, στο σχεδιασμό παραγωγής και διανομής κ.α. & 2) εκτέλεση της Εφοδιαστικής Αλυσίδας (Supply Chain Execution, SCE), που εστιάζει στην εφαρμογή των αποτελεσμάτων του Σχεδιασμού, χρησιμοποιώντας διαδικασίες όπως ο έλεγχος της παραγωγής και των αποθεμάτων, η διαχείριση των αποθεμάτων, η μεταφορά και η διανομή.

1.2 Δομή της διατριβής

Η παρούσα μελέτη επικεντρώνεται στη μεταφορά των προϊόντων από τις εγκαταστάσεις αποθήκευσης προς τους πελάτες (Vehicle Routing Problem, VRP). Η βελτιστοποίηση της δρομολόγησης των οχημάτων γίνεται σύμφωνα με τις απαιτήσεις των πελατών και τις συνθήκες διανομής. Σκοπός είναι η εύρεση της βέλτιστης διαδρομής βάση της οποίας θα εξυπηρετηθούν όλοι οι πελάτες με το ελάχιστο δυνατό κόστος εντός των χρονικών ορίων που ορίζει το κάθε πρόβλημα και των περιορισμών που επικρατούν.

Η μελέτη αυτή δεν διερευνά τη διαχείριση οχημάτων σε πραγματικό χρόνο κατά τη διάρκεια της εκτέλεσης των διανομών καθώς δεν είναι δυνατόν να συμπεριληφθούν στον πρόβλημα όλοι εκείνοι οι αστάθμητοι παράγοντες οι οποίοι μπορεί να προκύψουν κατά τη διάρκεια της διανομής. Τα προβλήματα αυτά μπορεί να επιδεινώσουν την αποδοτικότητα των προκαθορισμένων – στατικών αποφάσεων της δρομολόγησης. Ως αστάθμητοι παράγοντες μπορούν να ληφθούν γεγονότα που δημιουργούν δραματική αλλαγή της κατάστασης του προβλήματος όπως καταστάσεις αυξημένης κίνησης στους δρόμους των αστικών κέντρων, γεγονότα για τα οποία

ευθύνονται τα οχήματα (π.χ. μηχανικές βλάβες), γεγονότα που οφείλονται στη αγορά (π.χ. αλλαγή παραγγελίας πελατών ή διάρκεια διανομής) κλπ. τα οποία επιλύονται συνήθως εμπειρικά.

Επιπλέον μελετάται και το στοχαστικό πρόβλημα δρομολόγησης οχημάτων (Stochastic Vehicle Routing Problem, SVRP) στο οποίο η ζήτηση του κάθε πελάτη δεν είναι γνωστή εκ των προτέρων αλλά μόνο όταν το όχημα επισκεφτεί τον πελάτη. Έτσι τίθεται το ερώτημα αν το όχημα πριν επισκεφτεί τον κάθε πελάτη θα πρέπει να επιστρέψει στην αποθήκη για ανεφοδιασμό.

Ένα άλλο πρόβλημα που διερευνάται είναι το πρόβλημα του πλανόδιου πωλητή (Traveling Salesman Problem, TSP) το οποίο ανήκει στην κατηγορία των προβλημάτων βελτιστοποίησης. Το TSP συνίσταται στην προσπάθεια του πλανόδιου πωλητή να επισκεφτεί όλες τις πόλεις της περιοχής, επισκέπτοντας μία κάθε φορά, και να επιστρέψει στην πόλη από την οποία ξεκίνησε διανύοντας την μικρότερη απόσταση. Επέκταση του TSP είναι το πιθανοτικό πρόβλημα του πλανόδιου πωλητή (Probabilistic Traveling Salesman Problem, PTSP) στο οποίο απαιτείται να επισκεφτεί ο πωλητής την κάθε πόλη ή όχι βάση μίας πιθανότητας.

Στο Κεφάλαιο 2 παρουσιάζεται το γενικό πρόβλημα δρομολόγησης οχημάτων (VRP) καθώς και το στοχαστικό (SVRP) οι παράγοντες που μπορεί να επηρεάσουν το εκάστοτε πρόβλημα καθώς οι συνθήκες που μελετώνται στην παρούσα μελέτη. Δίνεται επίσης το γενικό πρόβλημα του πλανόδιου πωλητή (TSP) και το πιθανοτικό (PTSP) και οι συνθήκες που το διέπουν.

Στο Κεφάλαιο 3 παρουσιάζεται ο Γενετικός Αλγόριθμος με τον οποίο επιλύονται τα ανωτέρω προβλήματα (VRP, SVRP, TSP, STSP), η δομή του και τα αποτελέσματα που προκύπτουν από τη χρήση του.

Στο Κεφάλαιο 4 τα προβλήματα που προαναφέρθηκαν επιλύονται χρησιμοποιώντας έναν εξελικτικό αλγόριθμο, τον Differential Evolution και παρουσιάζονται τα αποτελέσματα που προκύπτουν από αυτόν.

Στο Κεφάλαιο 5 γίνεται σύγκριση των δύο αλγορίθμων που χρησιμοποιήθηκαν.

Στο Κεφάλαιο 6 παρουσιάζονται τα συμπεράσματα.

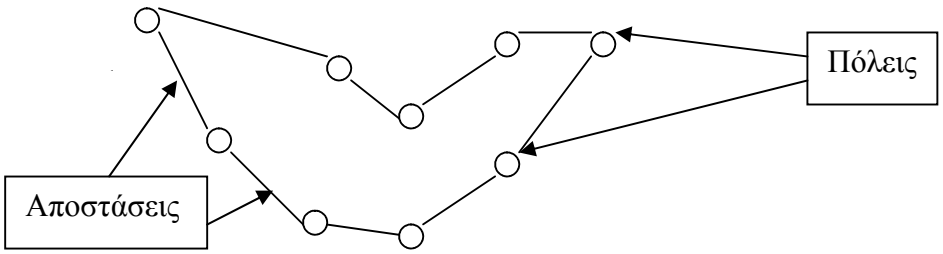
Κεφάλαιο 2
Το Πρόβλημα του Πλανόδιου Πωλητή και το Πρόβλημα Δρομολόγησης Οχημάτων

2.1 Το Πρόβλημα του Πλανόδιου Πωλητή

Στο Πρόβλημα του Πλανόδιου Πωλητή (TSP) στόχος είναι να βρεθεί η συντομότερη διαδρομή για όλες τις πόλεις που πρέπει να επισκεφτεί ο πωλητής. Το πρόβλημα του πλανόδιου πωλητή είναι πιθανώς το πιο γνωστό πρόβλημα και εκτενώς μελετημένο στο πεδίο της Συνδυαστικής Βελτιστοποίησης. Ευρετικές μέθοδοι για την επίλυση του TSP επικεντρώνονται στις μεθόδους δημιουργίας της διαδρομής και στις μεθόδους βελτίωσής της. Το πρόβλημα με τις ευρετικές μεθόδους είναι ότι παρόλο που είναι γρήγορες συνήθως δεν παράγουν πολύ καλές λύσεις. Οι πιο γνωστοί αλγόριθμοι βελτίωσης είναι ο 2-opt, όπου δύο σημεία της διαδρομής διαγράφονται και συνδέονται ξανά με διαφορετική σειρά ώστε να προκύψει μία νέα λύση. Ο πιο γνωστός ευρετικός αλγόριθμος που παράγει τα καλύτερα αποτελέσματα είναι ο Lin-Kernigham heuristic [S.Lin and B.W.Kernigham, (1973)]. Τα τελευταία δεκαπέντε χρόνια νέοι δρόμοι έχουν ανοιχτεί με τους ευρετικούς αλγόριθμους όπως οι simulating annealing, tabu search, γενετικοί αλγόριθμοι και νευρωνικά δίκτυα.

2.1.1. Περιγραφή του προβλήματος (TSP)

Το πρόβλημα του πλανόδιου πωλητή μπορεί να μοντελοποιηθεί ως ένα γράφημα, όπου οι πόλεις είναι οι κόμβοι, τα μονοπάτια που συνδέουν τις πόλεις είναι οι ακμές των κορυφών και η απόσταση των δύο πόλεων είναι το μήκος της ακμής που τις συνδέει. Ο πωλητής πρέπει να επισκεφτεί όλες τις πόλεις και να επιστρέψει στην πόλη από όπου ξεκίνησε διανύοντας τη μικρότερη διαδρομή.



Διάγραμμα 2.1: Απλό παράδειγμα του πλανόδιου πωλητή

Το πρόβλημα του πλανόδιου πωλητή απαιτεί τον καθορισμό ενός κύκλου ελαχίστου κόστους που περνά από κάθε κόμβο του συσχετιζόμενου γραφήματος ακριβώς μία φορά. Εάν το κόστος του ταξιδιού μεταξύ δύο τοποθεσιών δεν εξαρτάται από την κατεύθυνση του γραφήματος, τότε ενα συμμετρικό TSP και η απόσταση των δύο πόλεων είναι η ίδια από κάθε κατεύθυνση, δημιουργώντας ένα μη κατευθυνόμενο γράφημα.

Στο μη συμμετρικό TSP, τα μονοπάτια μπορεί να μην υπάρχουν και προς τις δύο κατευθύνσεις ή οι αποστάσεις μπορεί να είναι διαφορετικές, δημιουργώντας ένα κατευθυνόμενο γράφημα. Δρόμοι μονής κατεύθυνσης ή παρακαμπτήριοι δρόμοι είναι παραδείγματα ενός κατευθυνόμενου γραφήματος.

Το πρόβλημα έχει προτυποποιηθεί με πολλούς τρόπους. Μια απλή προτυποποίηση είναι η ακόλουθη:

Ένα γράφημα G για το TSP είναι ένα πλήρες σταθμισμένο, μη προσανατολισμένο γράφημα που καθορίζεται από ζεύγη (N,d) όπου N είναι το σύνολο των κόμβων, και d η συνάρτηση απόστασης των κορυφών (κόμβων) του γραφήματος και ικανοποιεί τις:

- d(i,j) = d(j,i) για όλα τα *i* και *j* που ανήκουν στο N
- d(i,j) ≥ 0 για όλα τα *i* και *j* που ανήκουν στο N
- d(i,j) + d(j,k) ≥ d(i,k) για όλα τα *i* και *j* που ανήκουν στο N

Η συνθήκη 3 ονομάζεται τριγωνική ανισότητα. Ο αριθμός d(i,j) ονομάζεται μήκος ή βάρος του τόξου (i,j).

2.1.2. Περιγραφή του προβλήματος

Στον πρόβλημα που θα επιλύσουμε στη συνέχεια υπάρχουν *n* πελάτες τους οποίους ο πωλητής πρέπει επισκεφτεί μία φορά τον καθένα. Η αντικειμενική συνάρτηση έχει ως σκοπό την ελαχιστοποίηση του κόστους διαδρομής.

Η αντικειμενική συνάρτηση είναι:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} * x_{i,j} \qquad (2.1)$$

υπό

$$x_{i,j} = \begin{cases} 1, \text{ εάν ο πωλητής επισκέπτεται τον πελάτη j αμέσως μετά τον πελάτη i} \\ 0 \text{ , αλλιώς} \end{cases} \quad (2.2)$$

όπου $c_{i,j}$ είναι η απόσταση από τον πελάτη i στον πελάτη j. Η απόσταση μεταξύ των πελατών είναι στη δική μας περίπτωση το κόστος μίας διαδρομής το οποίο καλούμαστε να ελαχιστοποιήσουμε.

2.2 Πρόβλημα Δρομολόγησης Οχημάτων

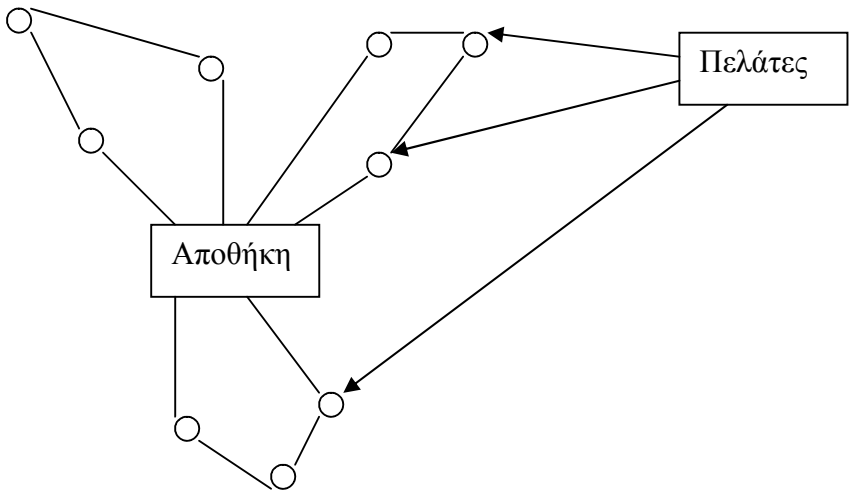
Το Πρόβλημα Δρομολόγησης Οχημάτων παρουσιάστηκε πρώτα από τους Dantzin και Ramser (1959). Καθώς είναι ένα NP-δυσκολίας πρόβλημα, μεγάλος αριθμός τεχνικών επίλυσης προτάθηκαν. Αυτές οι τεχνικές ταξινομούνται σε δύο μεγάλες κατηγορίες: Κλασσικοί Ευρετικοί οι οποίοι αναπτύχθηκαν κυρίως μεταξύ 1960 και 1990 (Altinkemer & Gavish 1991, Bodin & Golden 1981, Bodin et al. 1983, Christofides Mingozzi & Toth 1979, Clarke & Wright 1964, Destrochers & Verhoog 1989, Fisher & Jaikumar 1981, Foster & Ryan 1976, Gillett & Miller 1074. Lin 1965, Lin & Kernighan 1973, Mole & Jameson 1976, Wark & Holt 1994) και μεθευρετικοί οι οποίοι αναπτύχθηκαν τα τελευταία δεκαπέντε χρόνια. Οι μεθευρετικοί αλγόριθμοι ταξινομούνται σε κατηγορίες βασιζόμενοι στην στρατηγική που χρησιμοποιούν. Η μέθοδος Tabu Search χρησιμοποιείται πιο συχνά στο πρόβλημα του πλανόδιου πωλητή και πολλοί ερευνητές έχουν προτείνει αλγορίθμους βασιζόμενοι σε αυτήν την μέθοδο (Barbarosoglou & Ozgur 1999, Cordeau, Gendreau, Laporte, Potvin & Semete 2002, Gendreau, Hertz & Laporte 1994, Osman 1993, Rego 1998, Rego 2001, Taillard 1993, Toth & Vigo 2003, Xu & Kelly 1996). Πολλοί αποτελεσματικοί αλγόριθμοι βασίζονται στην ιδέα Adaptive Memory σύμφωνα με την οποία δημιουργούνται υψηλής απόδοσης VRP λύσεις και στη συνέχεια αντικαθίστανται από λύσεις που προήλθαν από τις μεθόδους που αναφέρθηκαν. (Rochat & Taillard 1995, Tarantillis 2005, Tarantillis & Kiranoudis 2002). Τα τελευταία δέκα χρόνια μεγάλος αριθμός μεθευρετικών αλγορίθμων, οι οποίοι εμπνέονται από τους νόμους της φύσης, απευθύνονται στο πρόβλημα του πλανόδιου πωλητή (Baker & Ayechew 2003, Berger & Barkaoui 2003, Marinakis, Migdalas & Pardalos 2007, Prins 2004), ant colony optimization (Bullnheimer, Hartl & Strauss 1999, Reimann, Stummer & Doerner

2002, 2004), honey bees mating optimization (Marinakis, Marinaki & Dounias 2008) και άλλες εξελικτικές μέθοδοι (Cordeau, Gendreau, Hertz, Laport & Sormany 2005, Mester & Braysy 2005)

2.2.1. Περιγραφή του προβλήματος (VRP)

Το πρόβλημα δρομολόγησης οχημάτων (VRP) παρουσιάζεται συνεχώς στην καθημερινή μας ζωή. Αφορά τη διανομή αγαθών από την αποθήκη στους πελάτες με ένα στόλο οχημάτων των οποίων η διαχείριση ταιριάζει με το μοντέλο δρομολόγησης των οχημάτων.

Ένα τυπικό πρόβλημα δρομολόγησης οχημάτων είναι ένα πρόβλημα σχεδιασμού βέλτιστης διαδρομής βάσει των διαθέσιμων οχημάτων, της χωρητικότητας του κάθε οχήματος και των πελατών που πρέπει να εξυπηρετηθούν. Οι διαδρομές πρέπει να σχεδιαστούν έτσι ώστε το όχημα να επισκέπτεται μία φορά τον κάθε πελάτη, οι διαδρομές να ξεκινούν και να καταλήγουν στην αποθήκη και η συνολική ζήτηση των πελατών της κάθε διαδρομής να μην ξεπερνάει την χωρητικότητα του οχήματος.



Διάγραμμα 2.2: Απλό παράδειγμα δρομολόγησης οχημάτων

Το πρόβλημα δρομολόγησης οχημάτων όπως αντιμετωπίζεται στην πραγματικότητα περιλαμβάνει πολλούς περιορισμούς στις διαδρομές διανομής τους οποίους το όχημα πρέπει να ακολουθήσει. Οι περιορισμοί αυτοί μπορούν να ταξινομηθούν σε δύο

κατηγορίες ανάλογα με το αν σχετίζονται με τα οχήματα ή τους πελάτες:

Οχήματα

- Κάθε όχημα έχει ένα όριο χωρητικότητας των αγαθών που μπορεί να μεταφέρει.
- Κάθε όχημα έχει ένα συγκεκριμένο χρόνο λειτουργίας από τη στιγμή που θα ξεκινήσει από την αποθήκη μέχρι να επιστρέψει, τυπικά δεν πρέπει να παραβιάζονται οι ώρες εργασίας του οδηγού.
- Κάθε όχημα έχει ένα χρονοδιάγραμμα μέσα στο οποίο πρέπει να φύγει από την αποθήκη, ώστε να είναι διαθέσιμη η αποθήκη να εφοδιάσει άλλα οχήματα
- Κάθε όχημα έχει συγκεκριμένη περίοδο κατά την οποία δεν κάνει τίποτα (ο οδηγός ξεκουράζεται) .
- Κάθε όχημα έχει ένα κόστος κατά τη λειτουργία του

Πελάτες

- Κάθε πελάτης περιμένει να του παραδοθεί μία συγκεκριμένη ποσότητα αγαθών
- Κάθε πελάτης μπορεί να δεχθεί το όχημα να ξεφορτώσει κατά τη διάρκεια συγκεκριμένων διαστημάτων.
- Κάθε πελάτης έχει ένα σύνολο οχημάτων που δεν μπορούν να χρησιμοποιηθούν για παράδοση.
- Κάθε πελάτης έχει προτεραιότητα κατά την παράδοση (σε περίπτωση που το όχημα δεν προλαβαίνει να τους εξυπηρετήσει όλους)
- Κάθε πελάτης μπορεί να δεχθεί την παραγγελία σε ξεχωριστές επισκέψεις (πχ από δύο οχήματα)

Άλλοι παράγοντες

- Πολλαπλές διαδρομές κατά τη διάρκεια μίας ημέρας όπου το όχημα επιστρέφει στην αποθήκη για ανεφοδιασμό.
- Διαδρομές του ίδιου οχήματος μεγαλύτερες της μίας ημέρας (με διανυκτέρευση)
- Οχήματα χωρισμένα εσωτερικά για τη μεταφορά διαφορετικών αγαθών
- Περισσότερες από μία αποθήκες όπου μπορούν να ανεφοδιαστούν τα

οχήματα.

Αντικειμενική Συνάρτηση

Κατά το σχεδιασμό των διαδρομών οι παραπάνω περιορισμοί πρέπει να ληφθούν υπόψη με αποτέλεσμα να μπορούν να υιοθετηθούν διαφορετικές αντικειμενικές συναρτήσεις κάθε φορά:

- Ελαχιστοποίησε το αριθμό των οχημάτων που χρησιμοποιούνται
- Ελαχιστοποίηση στη συνολική διαδρομή
- Ελαχιστοποίηση συνδυασμό του αριθμού των οχημάτων και τη συνολική διαδρομή.

Το κόστος λειτουργίας των οχημάτων συχνά θεωρείται σταθερό έτσι ώστε η πρώτη αντικειμενική συνάρτηση να αναφέρεται στην ελαχιστοποίηση των σταθερών κοστών, η δεύτερη στην ελαχιστοποίηση του κόστους των αποστάσεων και η τρίτη στην ελαχιστοποίηση του συνολικού κόστους.

2.2.2. Περιγραφή του προβλήματος

Στον πρόβλημα που θα επιλύσουμε στη συνέχεια υπάρχει μία αποθήκη και ένα όχημα το οποίο θα πρέπει να εξυπηρετήσει *n* πελάτες κάθε φορά. Το όχημα έχει περιορισμένη χωρητικότητα Q και η κάθε διαδρομή, από τη στιγμή που θα φύγει το όχημα από την αποθήκη μέχρι να επιστρέψει, έχει ένα συγκεκριμένο χρονικό περιθώριο. Επίσης το όχημα χρειάζεται ένα συγκεκριμένο χρονικό διάστημα για να ξεφορτώσει το εμπόρευμα στον κάθε πελάτη. Η αντικειμενική συνάρτηση έχει ως σκοπό την ελαχιστοποίηση του κόστους διαδρομής χωρίς να παραβιάζονται οι περιορισμοί που έχουν τεθεί.

Η αντικειμενική συνάρτηση είναι:

$$\min \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} * x_{i,j} \tag{2.3}$$

υπό τους περιορισμούς

$$\sum_{i=1}^{n} \sum_{j=1}^{n} d_j * x_{i,j} \le Q, d = \mathbf{1},2...Q \ , \forall \text{ k} \tag{2.4}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(c_{i,i+1}+service\_time\right)*x_{i,j}\le \max\_tour\_length\ ,\ \forall\ \mathrm{k} \qquad (2.5)$$

$$x_{i,j}=\begin{cases}1\ ,\ \text{εάν το όχημα επισκέπτεται τον πελάτη j αμέσως μετά τον πελάτη i}\\ 0\ ,\ \text{αλλιώς}\end{cases} \qquad (2.6)$$

όπου $c_{i,j}$ είναι η απόσταση από τον πελάτη i στον πελάτη j, d_j είναι η ζήτηση του επόμενου πελάτη και η συνολική ζήτηση των πελατών που θα εξυπηρετηθούν σε κάθε k διαδρομή δεν πρέπει να ξεπερνάει τη χωρητικότητα Q του οχήματος , και $service\_time$ είναι ο χρόνος που δαπανά το όχημα για να ξεφορτώσει στον κάθε πελάτη. Η απόσταση μεταξύ των πελατών είναι στη δική μας περίπτωση το κόστος μίας διαδρομής το οποίο καλούμαστε να ελαχιστοποιήσουμε.

2.3 Πιθανοτικό Πρόβλημα του Πλανόδιου Πωλητή

Το Probabilistic Traveling Salesman Problem (PTSP) είναι μία προέκταση του κλασσικού Traveling Salesman Problem (TSP) και έχει μελετηθεί πολύ στο πεδίο της Συνδυαστικής Βελτιστοποίησης. Το PTSP είναι πιθανόν το θεμελιώδες στοχαστικό πρόβλημα δρομολόγησης [Powell WB., Jaillet P., Odoni A., (1995)]. Πρωτοπαρουσιάστηκε το 1985 από τον Jaillet στο PhD του [Jaillet P. (1985)]. Μερικές θεωρητικές ιδιότητες του PTSP ειπώθηκαν από τον Jaillet P. [Jaillet P. (1985)] και δημοσιεύτηκαν το 1988 [Jaillet P. (1988)]. Στο PTSP η ζήτηση σε κάθε κόμβο προκύπτει με πιθανότητα p ή δεν προκύπτει με πιθανότητα 1-p κατά τη διάρκεια μιας διαδρομής. Η κυρίως διαφορά μεταξύ του PTSP και TSP είναι ότι ενώ στο TSP η αντικειμενική συνάρτηση είναι να βρεθεί η συντομότερη διαδρομή και ο πωλητής να επισκεφτεί μία φορά την κάθε πόλη και να επιστρέψει από αυτήν που ξεκίνησε, στο PTSP η αντικειμενική συνάρτηση θέλει να ελαχιστοποιήσει το αναμενόμενο μήκος της προκαθορισμένης διαδρομής όπου ο κάθε πελάτης απαιτεί να τον επισκεφτεί ο πωλητής με μία συγκεκριμένη πιθανότητα. Η προκαθορισμένη διαδρομή μπορεί να μεταφραστεί σε μία πρότυπη διαδρομή επίσκεψης του κάθε πελάτη. Σε μία δοσμένη περίπτωση, ο πωλητής πρέπει να επισκεφτεί τους πελάτες σύμφωνα με τη προκαθορισμένη διαδρομή ενώ θα πρέπει να παρακάμψει τους πελάτες που δεν απαιτούν επίσκεψη [Liu YH. (2007)]. Το PTSP ανήκει στο NP-

δυσκολίας προβλήματα [Bertsimas DJ (1988)]. Έτσι υπάρχει ανάγκη για ισχυρούς ευρετικούς οι οποίοι θα βρουν υποβέλτιστες λύσεις σε λογικά χρονικά πλαίσια.

2.3.1. Περιγραφή του προβλήματος (PTSP)

Ένας τρόπος επίλυσης του προβλήματος είναι ο ακόλουθος [Bianchi L. (2006), Jaillet P. (1985)]: έστω ότι έχουμε ένα συνδεδεμένο γράφημα στο οποίο οι κόμβοι σημειώνονται ως V={1,2…,*n*}. Κάθε κόμβος *i* (=1,2…*n*) συνδέεται με μία πιθανότητα p_i η οποία αντιπροσωπεύει την πιθανότητα ο κόμβος *i* να παρόν στην παρούσα περίπτωση. Δοσμένης μίας διαδρομής τ, αν η προκαθορισμένη διαδρομή περιλαμβάνει *S* πελάτες, και η πιθανότητα επίσκεψης του κάθε πελάτη είναι $p(S)$, ο πωλητής θα πρέπει να καλύψει τη συνολική απόσταση $L_\tau(S)$ για να επισκεφτεί το σύνολο S των πελατών. Αυτό το πρόβλημα λαμβάνει βάρος $p(S)L_\tau(S)$ στον υπολογισμό του αναμενόμενου μήκους. Αν δηλώσουμε το μήκος της διαδρομής τ με L_τ τότε το πρόβλημα είναι να βρεθεί η προκαθορισμένη διαδρομή μέσο όλων των *n* πιθανών πελατών η οποία ελαχιστοποιεί τη συνάρτηση

$$E[L_\tau] = \sum_{S \subseteq V} p(S) L_\tau(S) \tag{2.7}$$

Η πιθανότητα για όλα τα υποσύνολα των πελατών S να απαιτεί επίσκεψη είναι

$$p(S) = \prod_{i \in S} p_i \prod_{i \in V - S} (1 - p_i) \tag{2.8}$$

Στην πραγματικότητα σε μία προκαθορισμένη διαδρομή τ = (1,2...,*n*) το αναμενόμενο μήκος της είναι:

$$E[L_\tau] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1 - p_k) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} p_i p_j \prod_{k=i+1}^{n} (1 - p_k) \prod_{l=1}^{j-1} (1 - p_l) \tag{2.9}$$

όπου d_{ij} είναι η απόσταση μεταξύ του κόμβου i και του κόμβου j.

Αυτή η συνάρτηση προκύπτει από την πιθανότητα του κάθε τόξου σε ένα ολοκληρωμένο γράφημα να χρησιμοποιηθεί, όταν σε μία προκαθορισμένη διαδρομή

κάποιοι πελάτες οι οποίοι δεν απαιτούν επίσκεψη παρακάμπτονται.

2.4 Στοχαστικό Πρόβλημα Δρομολόγησης Οχημάτων

Το Vehicle Routing Problem with Stochastic Demand ανήκει στην κατηγορία των προβλημάτων γνωστά ως Stochastic VRPs (SVRPs) [Leonora Bianchi, Monaldo Mastrolilli, Mauro Birattari, Max Manfrin, Marco Chiarabdini, Luis Paquete, Olivia Rossi-Doria]. Σε αυτήν την κατηγορία προβλημάτων στοιχεία του προβλήματος όπως το σύνολο των πελατών, η ζήτηση των πελατών ή ο χρόνος της διαδρομής είναι στοχαστικές μεταβλητές. Χαρακτηριστικό αυτών των προβλημάτων είναι ότι έχουν ένα στοιχείο ντετερμινιστικό.

Στο δικό μας πρόβλημα η στοχαστική μεταβλητή είναι η ζήτηση των πελατών. Η δυσκολία σε αυτού του είδους τα προβλήματα είναι η αντικειμενική συνάρτηση που θα χρησιμοποιηθεί για την εύρεση του κόστους της κάθε διαδρομής.

2.4.1. Περιγραφή του προβλήματος (SVRP)

Το VRPSD [Leonora Bianchi, Monaldo Mastrolilli, Mauro Birattari, Max Manfrin, Marco Chiarabdini, Luis Paquete, Olivia Rossi-Doria] καθορίζεται από $G\{V, A, D\}$, όπου $V\{0,1...n\}$ είναι το σύνολο των κόμβων-πελατών, $A\{(i,j): i,j \in V, i \neq j\}$ είναι τα τόξα που συνδέουν τους κόμβους-πελάτες και $D\{d_{i,j}: i,j \in V, i \neq j\}$ είναι το κόστος από τον i κόμβο-πελάτη στον j κόμβο-πελάτη. Ένα όχημα χωρητικότητας Q πρέπει να παραδώσει τα προϊόντα στους πελάτες, ανάλογα με τη ζήτησή τους, ελαχιστοποιώντας το συνολικό κόστος της διαδρομής. Θεωρούμε ότι η ζήτηση του κάθε πελάτη είναι στοχαστική μεταβλητή $\xi_i, i = 1,...,n$. Η πραγματική ζήτηση του κάθε πελάτη γίνεται γνωστή μόνο όταν το όχημα φτάνει στον εκάστοτε πελάτη και δεν υπερβαίνει τη χωρητικότητα (Q) του οχήματος και ακολουθεί διακριτή κατανομή $p_{ij} = Prob(\xi_i = k), k = 0,1,2,...,K \leq Q$.

Μία πιθανή λύση του προβλήματος είναι $s = (s(0), s(1), ...s(n))$ ξεκινώντας από την αποθήκη $s(0) = 0$ και ονομάζεται εκ των προτέρων διαδρομή (a priori tour). Το όχημα επισκέπτεται τους πελάτες βάσει της a priori διαδρομής και ανάλογα με την ζήτηση του πελάτη τίθεται το ερώτημα αν θα προχωρήσει στον επόμενο πελάτη ή θα

γυρίσει στην αποθήκη για ανεφοδιασμό. Μερικές φορές η επιλογή του ανεφοδιασμού είναι η καλύτερη επιλογή ακόμα και αν η αναμενόμενη ζήτηση του επόμενου πελάτη είναι μικρότερη από το απόθεμα του φορτηγού. Αυτή η πράξη ονομάζεται 'προληπτικός ανεφοδιασμός' και στόχος του είναι να αποφευχθεί το ρίσκο το όχημα να συνεχίσει τη διαδρομή του στον επόμενο πελάτη χωρίς να έχει αρκετό απόθεμα στη διάθεσή του για να τον εξυπηρετήσει.

Η αναμενόμενη απόσταση που θα διανύσει το όχημα για να εξυπηρετήσει όλους τους πελάτες προκύπτει από την αντικειμενική συνάρτηση η οποία προκύπτει ως εξής:

$$f_j(q) = Minimun\begin{Bmatrix} f_j^p(q) \\ f_j^r(q) \end{Bmatrix} \tag{2.10}$$

$$f_j^p(q) = c_{j,j+1} + \sum_{k:k\le q} f_{j+1}(q-k)p_{j+1,k} + \sum_{k:k>q}\left[b + 2c_{j+1,0} + f_{j+1}(q+Q-k)\right]p_{j+1,k} \tag{2.11}$$

$$f_j^r(q) = c_{j,0} + c_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k)p_{j+1,k} \tag{2.12}$$

υπό τον περιορισμό

$$f_n(q) = c_{n,0}, q \in S_n \tag{2.13}$$

όπου $f_j^p(q)$ είναι το αναμενόμενο κόστος στην περίπτωση που επιλεγεί το όχημα να συνεχίσει στον επόμενο πελάτη και $f_j^r(q)$ είναι το αναμενόμενο κόστος στην περίπτωση που επιλεγεί να γίνει ανεφοδιασμός και το όχημα επιστρέψει στην αποθήκη.

Κεφάλαιο 3
Γενετικός Αλγόριθμος

3.1 Εισαγωγή

Η ιδέα των γενετικών αλγορίθμων προέρχεται από την βιολογία και τον κανόνα που επικρατεί στη φύση ‘ο πιο ικανός επιβιώνει’. Ο Δαρβίνος ήταν ο πρώτος που ανέπτυξε την ιδέα ότι οι οργανισμοί, με το πέρασμα του χρόνου, εξελίσσονται και προσαρμόζονται στο περιβάλλον ώστε να μπορέσουν να επιβιώσουν. Οι ‘γονείς’ ενός οργανισμού ζευγαρώνουν μεταξύ τους και οι απόγονοι διατηρούν στοιχεία και από τους δύο γονείς. Τα παιδιά που θα κληρονομήσουν τα πιο ικανά χαρακτηριστικά θα μπορέσουν να επικρατήσουν και να αποκτήσουν με τη σειρά τους απογόνους. Λογικό είναι ότι οι απόγονοι που θα προκύψουν από γονείς με καλύτερα χαρακτηριστικά έχουν περισσότερες πιθανότητες επιβίωσης. Πολλές φορές χαρακτηριστικά που αποκτούν τα παιδιά οφείλονται σε τυχαίες μεταλλάξεις οι οποίες τα κάνουν περισσότερο ή λιγότερο προσαρμόσιμα στο περιβάλλον. Συνεπώς οι απόγονοι που θα αποκτήσουν είτε με διασταύρωση είτε με μετάλλαξη τα πιο ικανά χαρακτηριστικά θα μπορέσουν να προσαρμοστούν καλύτερα στο εκάστοτε περιβάλλον και να επιβιώσουν. Οι γενετικοί αλγόριθμοι είναι υπολογιστικές προσομοιώσεις οι οποίες βασίζονται σε αυτές τις θεωρίες και προσομοιώνουν τη φυσική επιλογή των βιολογικών συστημάτων.

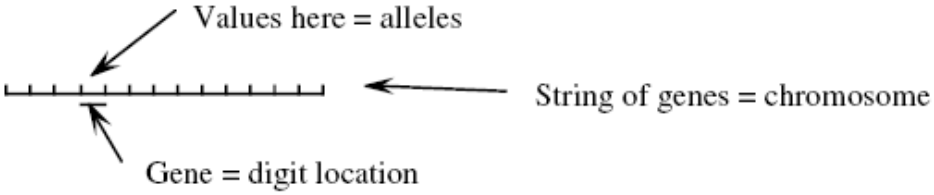
3.2 Αναπαριστώντας τον πληθυσμό των ατόμων

Η ‘αντικειμενική συνάρτηση’ υπολογίζει την ικανότητα ενός ατόμου να αναπαραχθεί και να δημιουργήσει απογόνους σε ένα περιβάλλον. Ο γενετικός αλγόριθμος προσπαθεί να μεγιστοποιήσει ή να ελαχιστοποιήσει, ανάλογα με το πρόβλημα, την τιμή της αντικειμενικής συνάρτησης $\overline{J}(\theta)$ επιλέγοντας τα πιο ικανά άτομα τα οποία συμβολίζονται με θ.

3.2.1. Κωδικοποίηση των χρωμοσωμάτων

Για να παρουσιάσουμε το γενετικό αλγόριθμο σε ένα Η/Υ, μετατρέπουμε το άτομο θ, σε μια αλληλουχία. Η αλληλουχία αντιπροσωπεύει ένα χρωμόσωμα σε ένα

βιολογικό σύστημα.



Σχήμα 3.1: Αλληλουχία αναπαράστασης ενός ατόμου

Χρωμόσωμα είναι μια αλληλουχία γονιδίων που μπορούμε να πάρουμε με διαφορετικούς συνδυασμούς. Στον Η/Υ, χρησιμοποιούμε κάποιο αριθμητικό σύστημα για να κωδικοποιήσουμε την αλληλουχία, όπως το δυαδικό {0,1} ή το δεκαδικό {0,1,2,3,4,5,6,7,8,9}. πχ. Ένα δυαδικό χρωμόσωμα μπορεί να είναι: 1011110001010, το οποίο έχει μήκος 13.

3.2.2. Ο πληθυσμός των ατόμων

Στη συνέχεια γίνεται η αναπαράσταση για ένα ολόκληρο σύνολο από άτομα. Έστω ότι k είναι ο αριθμός της γενιάς, $\theta_i^j(k)$ είναι ένα άτομο στο χρόνο k. Ο δείκτης j αναφέρεται στο j-οστό χρωμόσωμα. Επίσης μετράμε τα 'γνωρίσματα' σε κάθε χρωμόσωμα. Ο δείκτης i στο θ_i^j αναφέρεται στο i-οστό γνώρισμα του j-οστού χρωμοσώματος. Υποθέτοντας ότι το χρωμόσωμα j αποτελείται από p από αυτές τις παραμέτρους προκύπτει

$$\theta_i^j(k) = [\theta_1^j(k), \theta_2^j(k).....\theta_p^j(k)]$$

που είναι το j-οστό χρωμόσωμα.

Ο πληθυσμός των ατόμων σε δεδομένο χρόνο k είναι

$$P(k) = \{\theta^j(k) : j = 1,2.....S\}$$

και ο αριθμός των ατόμων στον πληθυσμό δίνεται από το S. Θέλουμε το μέγεθος του πληθυσμού (S) να είναι πολύ μεγάλο ώστε τα στοιχεία του πληθυσμού να μπορούν να καλύψουν το διάστημα της έρευνας. Αλλά δε θέλουμε το μέγεθος του πληθυσμού (S) να είναι υπερβολικά μεγάλο αφού αυτό αυξάνει τον αριθμό των υπολογισμών που πρέπει να εκτελεστούν.

Θα μπορούσε το μέγεθος του πληθυσμού να ποικίλει με το χρόνο, όπως γίνεται και στη φύση, καθώς και να εξαρτάται από τους πόρους του περιβάλλοντος και τον ανταγωνισμό μεταξύ των ατόμων. Το μέγεθος του πληθυσμού παίζει σημαντικό ρόλο

στη φύση. Η φύση ελέγχει τον αριθμό των ατόμων και δεν επιτρέπει μεγάλη αύξηση αυτού, ώστε το μέγεθος του πληθυσμού να ποικίλει σε ορισμένα όρια. Οι δικές μας προσομοιώσεις της εξέλιξης συχνά δεν το εκμεταλλεύονται αυτό εξαιτίας της έλλειψης υπολογιστικών πόρων.

3.2.3. Γενετικές Διαδικασίες

Ο πληθυσμός P(k) τη στιγμή k συχνά αναφέρεται ως 'γενιά' των ατόμων τη στιγμή k. Βασικά, σύμφωνα με το Δαρβίνο, τα πιο ικανά άτομα επιβιώνουν ώστε να ζευγαρώσουν και αποκτήσουν απογόνους. Βαθμολογούμε τα άτομα ως 'πιο ικανά' με τη συνάρτησης $\overline{J}(\theta^j(k))$ τη στιγμή k. Για την επιλογή δημιουργούμε ένα *'χώρο ζευγαρώματος'* τη στιγμή k,

$$M(k) = \{m^j(k) : j = 1,2.....S\}$$

Ο *'χώρος ζευγαρώματος'* είναι ένα σύνολο χρωμοσωμάτων τα οποία έχουν επιλεγεί για αναπαραγωγή. Γίνεται επιλογή των ατόμων που θα μπουν στο *'χώρο ζευγαρώματος'* , τα άτομα ζευγαρώνουν και στη συνέχεια παράγονται οι μεταλλάξεις. Μετά τη μετάλλαξη μπορούμε να έχουμε ένα τροποποιημένο *'χώρο ζευγαρώματος'* τη στιγμή k, M(k).

Η επόμενη γενιά προκύπτει από το χώρο ζευγαρώματος M(k) όπως αυτός έχει τροποποιηθεί

$$P(k+1) = M(k)$$

Η εξέλιξη εμφανίζεται καθώς πάμε από μία γενιά τη στιγμή k στην επόμενη γενιά τη στιγμή k+1.

3.2.4. Επιλογή

Υπάρχουν πολλοί τρόποι για να επιτευχθεί η επιλογή, αλλά η πιο συνηθισμένη που χρησιμοποιείται στην πράξη είναι η τεχνική Επιλογή Αναλογικής Καταλληλότητας (*Fitness-Proportionate Selection)*.

Επιλογή Αναλογικής Καταλληλότητας

Σε αυτήν την περίπτωση, επιλέγουμε ένα άτομο (το *i*-οστό χρωμόσωμα) για ζευγάρωμα αφήνοντας το m^j (k) να είναι ίσο με το θ^i (k)

$$p_i = \frac{\overline{J}(\theta^i(k))}{\sum_{j=1}^{S} \overline{J}(\theta^j(k))}$$

Για να διευκρινίσουμε τη σημασία αυτής της φόρμουλας και έπειτα την επιλογή της στρατηγικής, χρησιμοποιούμε έναν τροχό ο οποίος διαιρείται όπως μία πίτα σε S περιοχές. Η *i*-οστή περιοχή σχετίζεται με το *i*-οστό άτομο του P(k). Κάθε περιοχή συγκρίνεται με τη συνολική 'πίτα' και η τιμή της δίνεται από το p_i . Αρχικά παίρνουμε τυχαία μία πιθανότητα. Περιστρέφουμε τον τροχό και αν η τιμή (p_i) της περιοχής που θα δείξει ο δείκτης *i* είναι μεγαλύτερη από την τιμή της πιθανότητας τότε το άτομο τοποθετείται στο *'χώρο ζευγαρώματος'* M(k). Ο τροχός περιστρέφεται S φορές έτσι ώστε να προκύψουν S στοιχεία στο *'χώρο ζευγαρώματος'* και το μέγεθος του πληθυσμού να μείνει σταθερό.

Άτομα τα οποία είναι πιο υγιή καταλήγουν με περισσότερα αντίγραφα στο *'χώρο ζευγαρώματος'* , επιπλέον χρωμοσώματα με μεγαλύτερα από το μέσο όρο επίπεδα υγείας θα δώσουν περισσότερα στοιχεία στην επόμενη γενιά. Την ίδια στιγμή χάρη στην πιθανολογική φύση της διαδικασίας επιλογής είναι πιθανό ότι κάποια σχετικά μη υγιή άτομα μπορεί να καταλήξουν στο *'χώρο ζευγαρώματος'* .

3.2.5. Διασταύρωση

Ως διασταύρωση εννοείται το ζευγάρωμα δυο ατόμων , η οποία σε βιολογικό επίπεδο περιλαμβάνει τη διαδικασία του συνδυασμού χρωμοσωμάτων.

Για την προσομοίωση της εξέλιξης στον υπολογιστή η λειτουργία της διασταύρωσης λειτουργεί στο *'χώρο ζευγαρώματος',* M(k). Πρώτα ορίζεται η πιθανότητα του διασταύρωσης p_c (συνήθως επιλέγεται να είναι κοντά στο 1 γιατί όταν προκύπτει ζευγάρωμα σε βιολογικά συστήματα, γενετικό υλικό εναλλάσσεται σίγουρα μεταξύ των γονέων). Υπάρχουν πολλοί τύποι διασταύρωσης αλλά ο πιο απλός είναι ο '*single-point crossover'*.

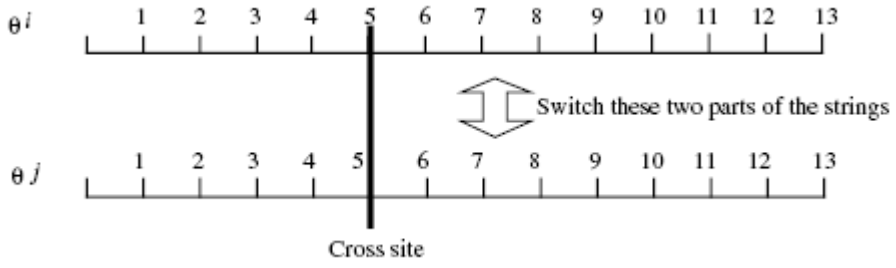
Single –point Crossover

Η διαδικασία της *single-point crossover* περιλαμβάνει τα ακόλουθα βήματα.

1. Τυχαία ζευγάρια ατόμων μπαίνουν στο *'χώρο ζευγαρώματος'* M(k). Υπάρχουν πολλοί τρόποι για να γίνει αυτό. Μπορεί απλά να επιλεγεί ένα

άτομο από το 'χώρο ζευγαρώματος' και μετά τυχαία να επιλεγεί ένα διαφορετικό άτομο για να ζευγαρώσουν μαζί. Ή όλα τα άτομα μπορούν να ζευγαρώσουν το καθένα με το άτομο που έχει δίπλα του.(όπου 'ακριβώς δίπλα' καθορίζεται από το πως είναι τα άτομα αριθμημένα). Με αυτήν την προσέγγιση, αν ο αριθμός των ατόμων είναι μονός, το τελευταίο άτομο μπορεί να γίνει ζευγάρι με ένα άλλο άτομο το οποίο έχει γίνει ήδη ζευγάρι (ή θα μπορούσε να ζευγαρώσει με ένα από τα πιο υγιή άτομα).

2. Έστω τα ζευγάρια χρωμοσωμάτων θ^j, θ^i που δημιουργήθηκαν στο βήμα 1. Γέννησε ένα τυχαίο αριθμό r Є [0,1]
    a) Αν $r<p_c$ τότε διασταύρωσε το θ^j και θ^i . Για να διασταυρώσεις τα χρωμοσώματα διάλεξε ένα '*σημείο διασταύρωσης'* στην τύχη και αντάλλαξε τα ψηφία στη δεξιά πλευρά μεταξύ των δύο αλληλουχιών. Αυτή η διαδικασία φαίνεται στο σχήμα 14.2. Σε αυτό το παράδειγμα το '*σημείο διασταύρωσης'* είναι στην θέση 5 στην αλληλουχία και ανταλλάσσονται τα τελευταία οχτώ ψηφία μεταξύ των δύο αλληλουχιών.



Σχήμα 3.2: Παράδειγμα διασταύρωσης

    b) Αν $r>p_c$ τότε δε θα γίνει διασταύρωση και δε θα τροποποιηθεί η αλληλουχία.
3. Επανέλαβε το βήμα 2 για κάθε ζευγάρι των αλληλουχιών στο '*σημείο διασταύρωσης'* M(k).

3.2.6. Μετάλλαξη

Όπως η διασταύρωση, η μετάλλαξη τροποποιεί το *'χώρο ζευγαρώματος'* (αφού η επιλογή έχει γίνει). Η διαδικασία της μετάλλαξης κανονικά εκτελείται στα στοιχεία του *'χώρο ζευγαρώματος'* αφού η διασταύρωση έχει πραγματοποιηθεί.

Μετάλλαξη γονιδίων

Για να εκτελεστεί μια μετάλλαξη στον υπολογιστή, πρώτα επιλέγεται η πιθανότητα μετάλλαξης p_m. Κάθε γονίδιο που εντοπίζεται σε κάθε χρωμόσωμα μεταλλάσσεται με πιθανότητα p_m ανάλογα με το αριθμητικό σύστημα που χρησιμοποιείται. Για παράδειγμα στο δυαδικό σύστημα θα μπορούσε να μεταλλαχθεί το

1010111 σε

1011111

όπου το τέταρτο στοιχείο μεταλλάχτηκε σε μονάδα. Για μία βάση του δεκαδικού συστήματος, θα μπορούσε απλά να επιλεγεί ένας αριθμός στην τύχη και να αντικατασταθεί με ένα ψηφίο, αν πρόκειται να μεταλλαχθεί απλά μία ψηφιακή θέση (κανονικά η αντικατάσταση δεν είναι έγκυρη αν αντικατασταθεί ένα ψηφίο με την ίδια αξία).

Στον αλγόριθμο που αναπτύσσεται στη συνέχεια, επειδή τα χρωμοσώματα αναπαρίστανται στο δεκαδικό σύστημα, η αντικατάσταση ενός στοιχείου του ατόμου με έναν οποιοδήποτε αριθμό μπορεί να μην δώσει έγκυρη λύση καθώς το άτομο που θα προκύψει μπορεί να περιλαμβάνει δύο ίδιους αριθμούς. Στο πρόβλημα που εξετάζουμε αυτό σημαίνει ότι το όχημα ή ο πλανόδιος πωλητής θα επισκεφτεί δύο φορές την ίδια πόλη και μία θα την αφήσει εκτός διαδρομής. Για να αποφύγουμε αυτήν την περίπτωση, βάση της πιθανότητα μετάλλαξης επιλέγουμε το κάθε στοιχείο του ατόμου αλλάζει θέση με ένα άλλο στοιχείο του ίδιου ατόμου. Για παράδειγμα θα μπορούσε να μεταλλαχθεί το άτομο

1 3 5 2 6 4 σε

1 6 5 2 3 4

όπου το δεύτερο στοιχείο άλλαξε θέση με το πέμπτο. Στο ίδιο άτομο μπορούν να γίνουν περισσότερες από μία μεταλλάξεις, στην ουσία αντιμεταθέσεις των στοιχείων, καθώς σαρώνονται όλα τα στοιχεία του ατόμου

3.2.7. Τερματισμός του GA

Σε προηγούμενες ενότητες είδαμε πως παράγουμε αποτελεσματικά γενιές και τελικά πως προσομοιώνουμε την εξέλιξη. Καθώς η βιολογική επαναστατική διαδικασία συνεχίζεται, πιθανώς κατά τρόπο αόριστο, υπάρχουν πολλές στιγμές που θα θέλαμε να τερματίσουμε τη διαδικασία και να βρούμε τα ακόλουθα:

- Το άτομο του πληθυσμού – έστω $\theta^*(k)$ – το οποίο μεγιστοποιεί ή ελαχιστοποιεί καλύτερα την συνάρτηση προσαρμογής (δεν χρησιμοποιούμε το θ^* καθώς επιφυλασσόμαστε για το μέγιστο σημείο αν υπάρχει (υπάρχουν)). Για να το καθορίσουμε αυτό χρειαζόμαστε επίσης να γνωρίζουμε τον αριθμό της γενιάς k όπου το ποιο υγιές άτομο υπήρχε (δεν είναι απαραίτητο να υπήρχε στην τελευταία γενιά). Ο κώδικας μπορεί να σχεδιαστεί ώστε ο γενετικός αλγόριθμος να κρατάει συνεχώς τα ίχνη του $\overline{J}$ με την υψηλότερη τιμή, και ο αριθμός της γενιάς και των ατόμων έχουν επιτύχει αυτή την τιμή του $\overline{J}$.
- Η τιμή της συνάρτησης ικανότητας $\overline{J}(\theta^*(k))$. Ενώ για μερικές εφαρμογές αυτή η τιμή μπορεί να μην είναι σημαντική για άλλες μπορεί να είναι χρήσιμη (σε πολλά προβλήματα μεγιστοποίησης συνάρτησης).
- Ο μέσος όρος των υγιών τιμών στον πληθυσμό
- Πληροφορίες σχετικά με τον τρόπο που ο πληθυσμός εξελίσσεται, ποιες περιοχές του χώρου έρευνας επισκέφτηκαν και πόσο εξελίχθηκε η συνάρτηση προσαρμογής με το χρόνο. Ο κώδικας μπορείς να σχεδιαστεί ώστε ο γενετικός αλγόριθμος να παρέχει διαγράμματα ή έντυπα με όλα τα σχετικά δεδομένα του γενετικού αλγορίθμου.

Υπάρχει τότε η ερώτηση του πως θα τερματίσουμε το γενετικό αλγόριθμο. Υπάρχουν πολλοί τρόποι για να τερματίσουμε τον γενετικό αλγόριθμο, πολλοί από αυτούς είναι παρόμοιοι με τους όρους τερματισμού που χρησιμοποιούνται για συμβατικούς αλγόριθμους μεγιστοποίησης-ελαχιστοποίησης. Για να εισάγουμε μερικούς από αυτούς, έστω $\epsilon > 0$ ένας πολύ μικρός αριθμός και $M_1 > 0$ και $M_2 > 0$ να είναι ακέραιοι:

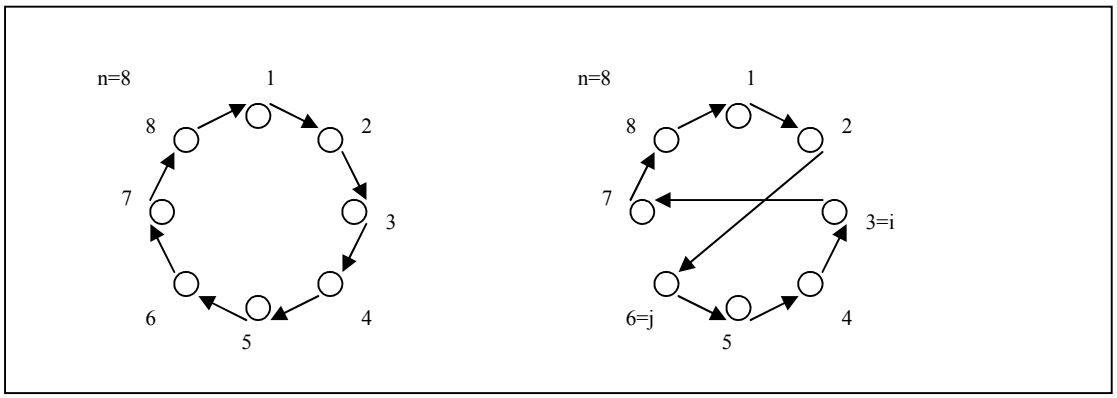
- Σταμάτα τον αλγόριθμο μετά την παραγωγή της γενιάς $P(M_1)$ – η οποία είναι μετά τη γενιά M_1.
- Σταμάτα τον αλγόριθμο αφού έχουν παραχθεί τουλάχιστον M_1 γενιές και τουλάχιστό M_2 βήματα έχουν περάσει όπου η μέγιστη (ή μέσος όρος) τιμή του $\overline{J}$ για όλα τα μέλη του πληθυσμού έχει αυξηθεί αλλά όχι περισσότερο από ϵ.
- Σταμάτα τον αλγόριθμο μόλις το $\overline{J}$ πάρει μία συγκεκριμένη τιμή

3.3 Αλγόριθμοι Τοπικής Αναζήτησης

Για να αποφύγουμε το πρόβλημα εγκλωβισμού σε τοπικό ελάχιστο που συναντάται συχνά στους γενετικούς αλγορίθμους αλλά και για τον πιο γρήγορο εντοπισμό των βέλτιστων πιθανών λύσεων εφαρμόζουμε μία μέθοδο τοπικής αναζήτησης σε κάθε άτομο του πληθυσμού. Στη συνέχεια αναλύεται η μέθοδος 2-opt και η 3-opt. Στον αλγόριθμο θα εφαρμοστεί η 3-opt.

3.3.1. 2-opt

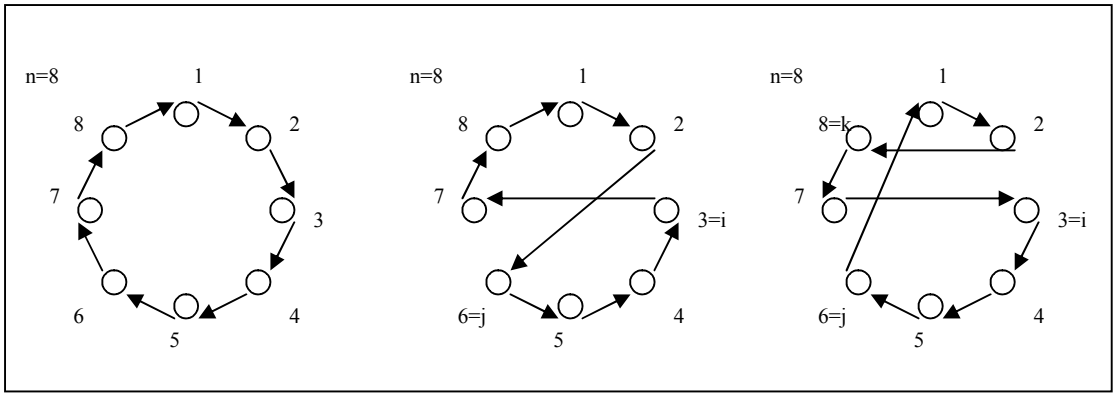
Στη μέθοδο 2-opt τοπικής αναζήτησης [Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella (2003)] , θεωρούμε μία εκ των προτέρων διαδρομή λ, η οποία αποτελείται από συνεχόμενους κόμβους. Επιλέγονται δύο τυχαίοι κόμβοι και αντιστρέφεται η αλληλουχία της διαδρομής μεταξύ των δύο διατηρώντας την υπόλοιπη διαδρομή αναλλοίωτη όπως παρουσιάζεται στην Εικόνα 2.1.



Εικόνα 3.1: Διαδρομή λ=1,2,...*i*,*i*+1,…,*j*,…,*n* (αριστερά) και διαδρομή λ΄=1,2,...,*i*-1,*j*,*j*-1,…,*i*,*j*+1,…,*n* (δεξιά). Η διαδρομή λ΄ προκύπτει από την αντιστροφή μέρους της διαδρομής (*i*,*i*+1,…,*j*), με *n*=8,*i*=3,*j*=6.

3.3.2. 3-opt

Η μέθοδος 3-opt είναι όμοια με την 2-opt, αλλά εδώ επιλέγονται τρεις κόμβοι, γίνεται αντιστροφή της διαδρομής μεταξύ των δύο πρώτων, και από τη διαδρομή που θα προκύψει γίνεται αντιστροφή του δεύτερου με τον τρίτο επιλεγμένο κόμβο.



Εικόνα 3.2: Διαδρομή λ=1,2,...*i*,*i*+1,…,*j*,…,*n* (αριστερά) και διαδρομή λ΄=1,2,...,*i*-1,*j*,*j*-1,…,*i*,*j*+1,…,*n* (κέντρο), διαδρομή λ΄΄=1,2,*i*-1,k,k-1,…*i*,*i*+1,…*j*,*j*+1,…*n*.

Για *i*=3, *j*=6,*k*=8 η αρχική διαδρομή είναι: λ=1,2,**3**,4,5,**6**,7,8,1 με αντιστροφή της διαδρομής μεταξύ των κόμβων 3 και 6 προκύπτει: λ΄=1,2,**6**,5,4,3,7,**8**,1και με επιπλέον αντιστροφή των κόμβων 6 και 8 προκύπτει η τελική διαδρομή:λ΄΄=1,2,8,7,3,4,5,6,1

3.4 Σχεδιασμός και Παράμετροι του Γενετικού Αλγορίθμου

Ο γενετικός αλγόριθμος που χρησιμοποιήθηκε για να επιλυθεί αυτό το πρόβλημα κωδικοποιήθηκε στη Matlab χρησιμοποιώντας δεκαδική κωδικοποίηση. Το χρωμόσωμα είναι ένα διάνυσμα *n* διαστάσεων [1,2..,*n*], το οποίο αναπαριστά μία πιθανή λύση του προβλήματος μας (τη σειρά με την οποία εξυπηρετήθηκε ο κάθε πελάτης) και το κάθε στοιχείο του διανύσματος αντιπροσωπεύει έναν κόμβο-πελάτη.

Ο αρχικός πληθυσμός δίνεται τυχαία. Κάθε πιθανή διαδρομή ξεκινάει από τον κόμβο 1 (αποθήκη) και η σειρά επίσκεψη των πελατών είναι τυχαία. Ο πληθυσμός κάθε γενιάς αποτελείται από 40 άτομα. Την πιθανότητα διασταύρωση (p_c) την έχουμε ορίσει 0.8 και την πιθανότητα μετάλλαξης (p_m) 0.2. Επίσης χρησιμοποιούμε τον αλγόριθμο 3-opt για τοπική αναζήτηση και πραγματοποιείται για 50 επαναλήψεις σε κάθε άτομο. Αν βρεθεί καλύτερη λύση πριν τελειώσει ο αριθμός των επαναλήψεων η τοπική αναζήτηση διακόπτεται και προχωράμε στο επόμενο άτομο.

Χρησιμοποιείται ελιτισμός και το καλύτερο άτομο της γενιάς μεταφέρεται κάθε φορά στην επόμενη χωρίς να μπει στη διαδικασία της διασταύρωσης και της μετάλλαξης εξασφαλίζοντας έτσι ότι η καλύτερη λύση θα είναι πάντα διαθέσιμη.

Στο τέλος κάθε γενιάς σε περίπτωση που προκύψουν περισσότερα από ένα άτομα τα οποία θα έχουν την ίδια λύση με τη βέλτιστη χρησιμοποιείται η συνάρτηση 3-opt

ώστε να τροποποιηθούν οι παρούσες λύσεις και να διαφοροποιηθούν από τη βέλτιστη ώστε να διερευνηθεί το πεδίο των λύσεων και να προκύψουν νέες λύσεις στην επόμενη γενιά.

Χρησιμοποιούμε *fitness-proportionate selection* για την επιλογή των ατόμων, *single-point crossover* για τη διασταύρωση και η μετάλλαξη γίνεται με αντιμετάθεση των στοιχείων του ατόμου. Επίσης μετά τη διαδικασία της διασταύρωσης χρησιμοποιείται μία συνάρτηση επιδιόρθωσης καθώς είναι πολύ πιθανό η λύση που θα προκύψει να περιλαμβάνει ορισμένους κόμβους δύο φορές ενώ κάποιοι να απουσιάζουν τελείως από την πιθανή λύση.

Για τερματικό κριτήριο επιτρέπουμε όχι παραπάνω από ένα καθορισμένο μέγιστο αριθμό επαναλήψεων (M_1). Επειδή εξετάζονται διαφορετικού μεγέθους προβλήματα ο μέγιστος αριθμός των επαναλήψεων (M_1) δεν είναι σταθερός, σε μεγαλύτερα προβλήματα επιτρέπουμε περισσότερες επαναλήψεις. Παρόλα αυτά προσθέτουμε επίσης και ένα άλλο τερματικό κριτήριο που μπορεί να σταματήσει τον αλγόριθμο πριν ο μέγιστος αριθμός επαναλήψεων επιτευχθεί. Ειδικότερα τερματίζουμε το πρόγραμμα αν το καλύτερο άτομο στο πληθυσμό δεν έχει αλλάξει περισσότερο από $\epsilon = 0.01$ στις τελευταίες 300 γενιές (M_2=300).

Αλγόριθμος 3.1 : Genetic Algorithm

Βήμα 1. Προσδιόρισμος των παραμέτρων του GA

Βήμα 2. Αρχικοποίηση του πληθυσμού P(0)

Βήμα 3. Για k=1 μέχρι N_{ga}

a) Υπολόγισε τη συνάρτηση προσαρμογής για κάθε άτομο

b) **Επιλογή**: Χρησιμοποιώντας τη *fitness-proportionate selection* επέλεξε από τον πληθυσμό τα άτομα που θα μπουν στο χώρο του ζευγαρώματος (M(k)).

c) **Διασταύρωση:** Με πιθανότητα p_c διασταύρωσε το τρέχον άτομο με ένα τυχαίο από το 'χώρο ζευγαρώματος'

d) **Επιδιόρθωση:** Γίνεται επιδιόρθωση της πιθανής λύσης

e) **Μετάλλαξη:** Με πιθανότητα p_m αντιμετάθεσε το τρέχον στοιχείο με ένα άλλο τυχαίο του διανύσματος.

f) **Τοπική Αναζήτηση:** Για κάθε άτομο που πέρασε στην επόμενη γενιά χρησιμοποίησε τη μέθοδο 3-opt για τη εύρεση καλύτερης λύσης.

Βήμα 4. Αποτελέσματα

3.5 Επίλυση VRP με χρήση του GA

Σε αυτήν την ενότητα επιλύεται το VRP με τον *αλγόριθμο 3.1*. Η αντικειμενική συνάρτηση προκύπτει από την Εξ. (2.3) υπό τους περιορισμούς των Εξ. (2.4), (2.5), (2.6).

Ο αλγόριθμος που αναλύθηκε εκτελέστηκε για 14 διαφορετικά προβλήματα στα οποία δίνονται οι θέσεις των κόμβων στο καρτεσιανό επίπεδο, το πλήθος το κόμβων *n*, η χωρητικότητα του οχήματος διανομής Q, η μέγιστη διαδρομή που μπορεί να διανύσει το φορτηγό από τη στιγμή που θα εγκαταλείψει την αποθήκη. Τα στοιχεία αυτά παρουσιάζονται στον επόμενο πίνακα:

File-n	***n***	***Q***	***max_tour_length***	***service_time***
par1-51	51	160	999999	0
par2-76	76	140	999999	0
par3-101	101	200	999999	0
par4-151	151	200	999999	0
par5-200	200	200	999999	0
par6-51	51	160	200	10
par7-76	76	140	160	10
par8-101	101	200	230	10
par9-151	151	200	200	10
par10-200	200	200	200	10
par11-121	121	200	999999	0
par12-101	101	200	999999	0
par13-121	121	200	720	50
par14-101	101	200	1040	90

Πίνακας 3.1: Δεδομένα αρχείων

Στα προβλήματα που εξετάζουμε, όπου το max_tour_length είναι 999999 ουσιαστικά δεν λαμβάνεται υπόψιν ο περιορισμός της διάρκειας της διαδρομής και το μόνο που μας ενδιαφέρει είναι να μην υπερβεί η συνολική ζήτηση των πελατών τη χωρητικότητα του οχήματος σε μία διαδρομή και όπου το service_time είναι 0 δεν απαιτείται κάποιος χρόνος για την εξυπηρέτηση του πελάτη, συνεπώς και η διάρκεια

της διαδρομής (το κόστος) είναι μικρότερο. Στο σύνολο των κόμβων *n* συμπεριλαμβάνεται και η αποθήκη.

Τα δεδομένα των προβλημάτων είναι όμοια ανά δύο, δηλαδή οι θέσεις των πελατών και η χωρητικότητα των φορτηγών είναι ίδιες. Με διαφορά ότι στο ένα πρόβλημα υπάρχει περιορισμός για τη μέγιστη διαδρομή (max_tour_length) και χρόνος που δαπανάται στην εξυπηρέτηση του πελάτη (service_time).

3.5.1. Αποτελέσματα VRP με χρήση GA

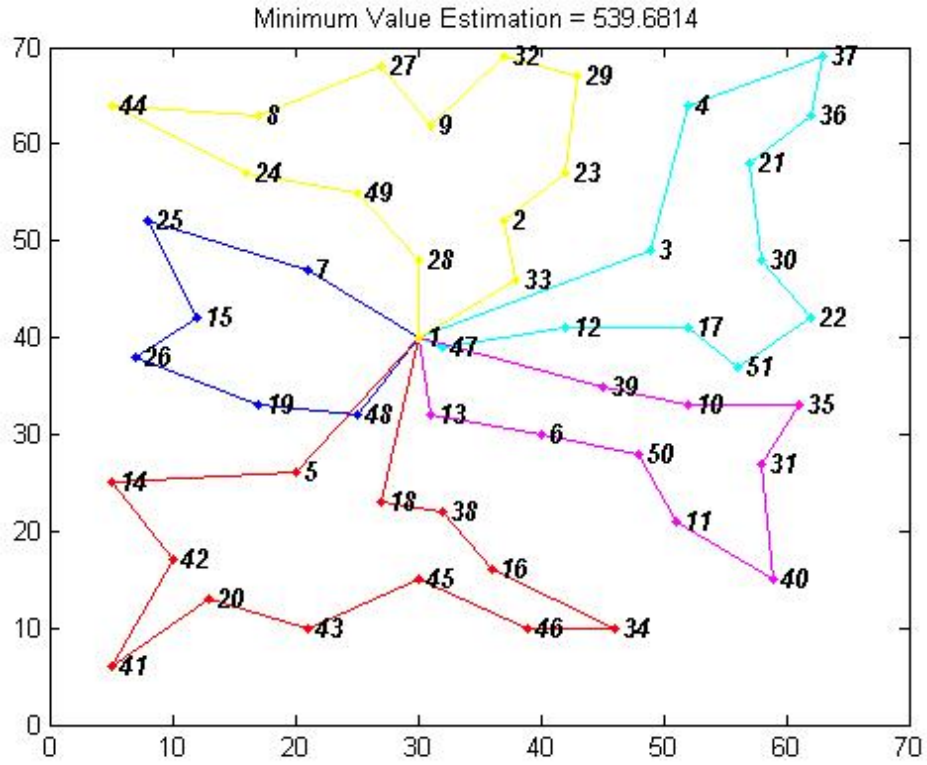
Σε αυτήν την ενότητα θα παρουσιαστούν τα αποτελέσματα που προέκυψαν από την επίλυση του προβλήματος δρομολόγησης οχημάτων με χρήση γενετικού αλγόριθμου και τοπική αναζήτηση 3-opt.

Στα διαγράμματα που ακολουθούν φαίνεται η διαδρομή που πρέπει να κάνει το όχημα για εξυπηρετήσει τους πελάτες καθώς και πόσες φορές επιστρέφει στην αποθήκη για ανεφοδιασμό. Το όχημα επιστρέφει στην αποθήκη είτε γιατί το εμπόρευμα που έχει στη διάθεση του δεν επαρκεί για να καλύψει τη ζήτηση του επόμενου πελάτη είτε γιατί παραβιάζεται ο περιορισμός service_time, δηλαδή ο χρόνος που μπορεί να βρίσκεται σε λειτουργία το όχημα ή οι ώρες εργασίας του οδηγού.

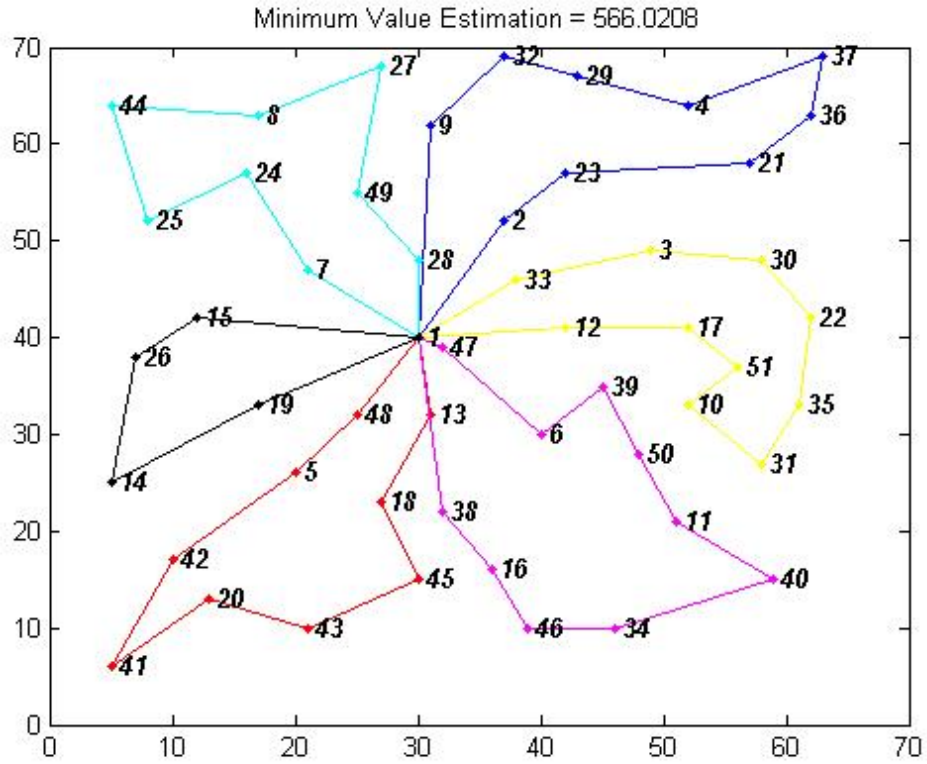
Τα πακέτα δεδομένων για τα οποία εκτελείται ο αλγόριθμος έχουν τα ίδια δεδομένα ανά δύο όσο αφορά τον αριθμό των πελατών, τις θέσεις αυτών και τη χωρητικότητα του φορτηγού. Έτσι το par_1 και το par_6 έχουν το ίδιο *n*, το ίδιο Q και το ίδιο κόστος μετάβασης από τον έναν πελάτη στον επόμενο. Στο par_1 δεν έχουμε service_time και ο περιορισμός της μέγιστης διαδρομής (max_tour_length) είναι 999999, δηλαδή δεν παραβιάζεται ποτέ συνεπώς είναι σαν μην υπάρχει. Στο par_6 υπάρχει service_time καθώς και μέγιστη τιμή της κάθε διαδρομής (max_tour_length). Το ίδιο ισχύει και για τα υπόλοιπα πακέτα δεδομένων και η αντιστοιχία είναι η εξής: par_2 –par_7, par_3 – par_8, par_4 – par_9, par_5 – par_10, par_11 – par_13, par_12– par_14[1].

Τα δεδομένα των αρχείων παρουσιάζονται στον πίνακα 3.1 και τα αποτελέσματα δίνονται όπως και η ανωτέρω διάταξη.

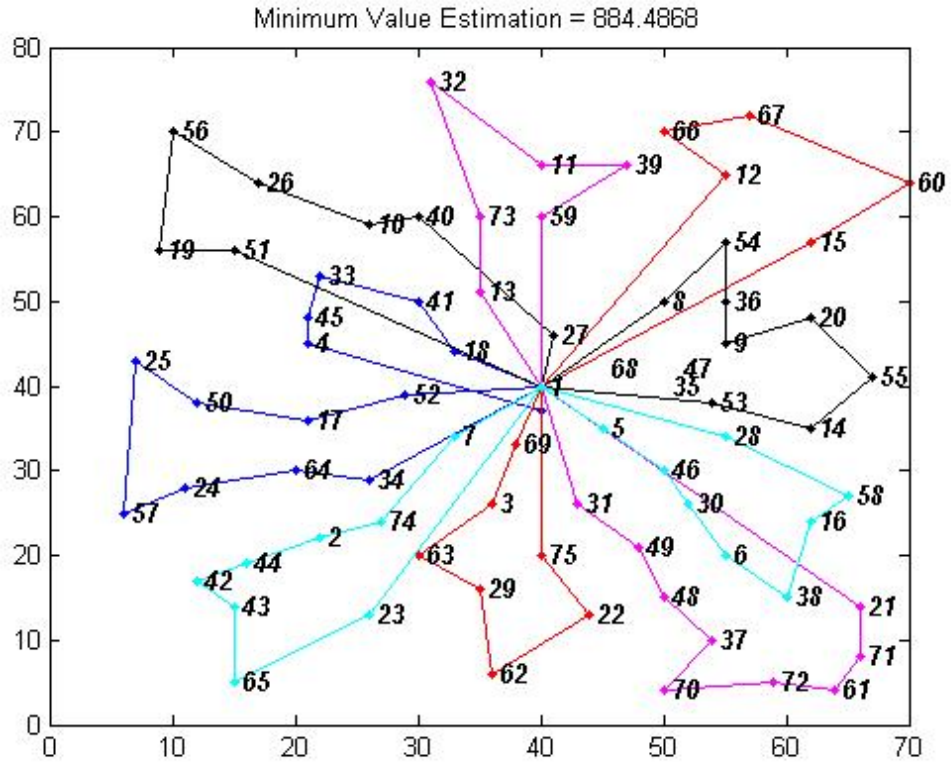
[1] Στα προβλήματα όπου ο αριθμός των πελατών είναι μεγάλος, παραλείπεται η αρίθμηση των κόμβων στα διαγράμματα για καλύτερη απεικόνιση της διαδρομής



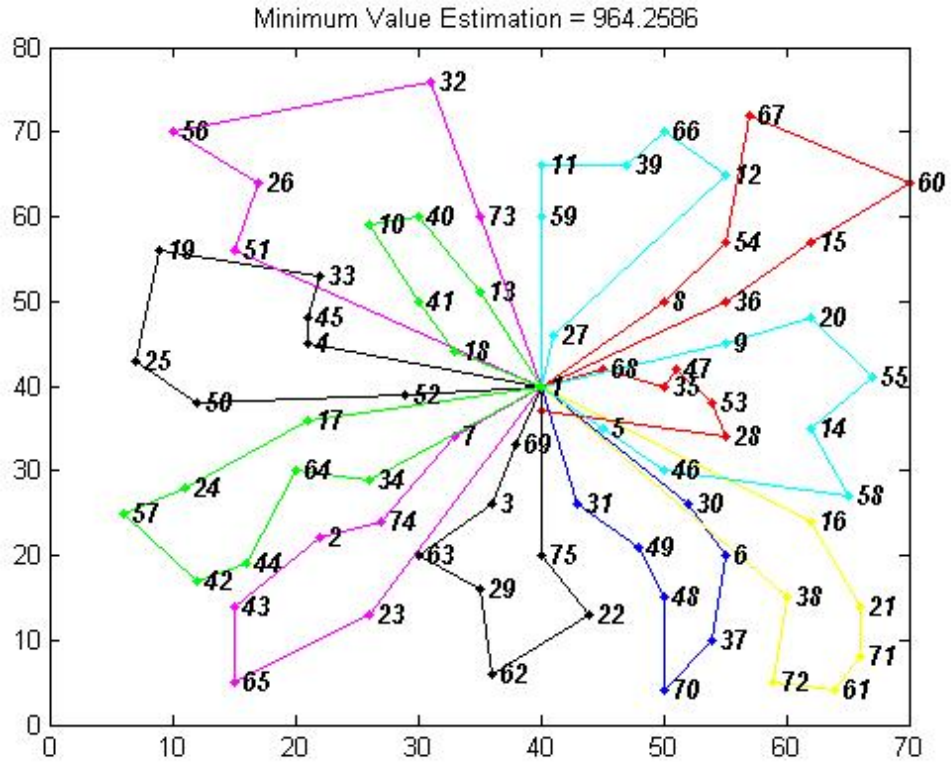
Διάγραμμα 3.1: par_1 : n=51, Q=160, max_tour_length=99999 ,service_time=0



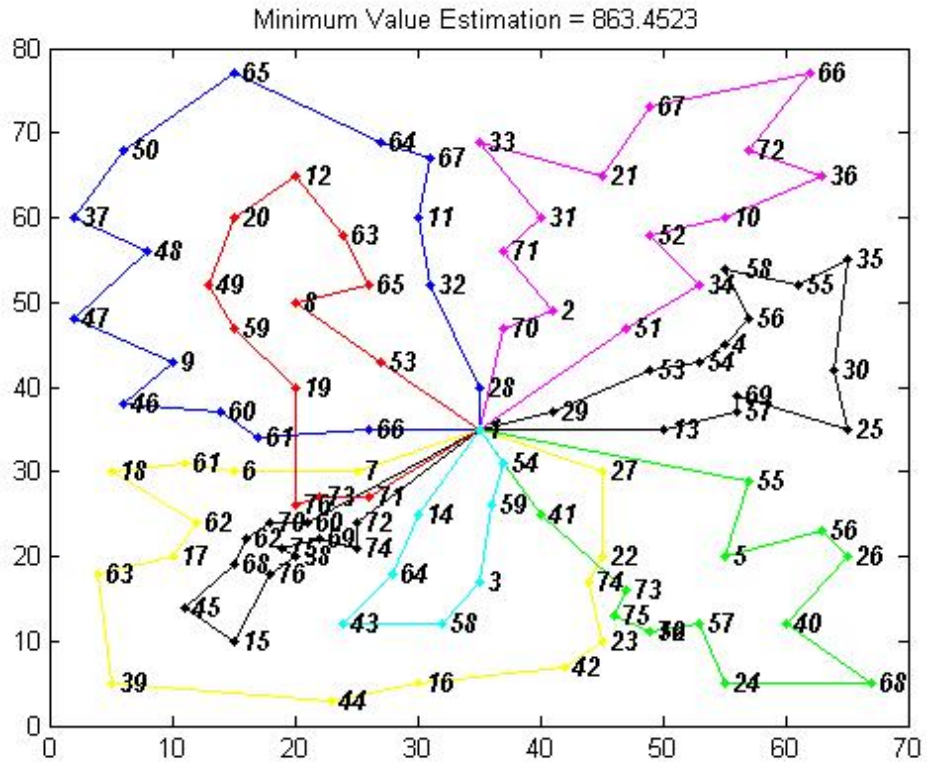
Διάγραμμα 3.2: par_6 : n=51, Q=160, max_tour_length=200, service_time=10



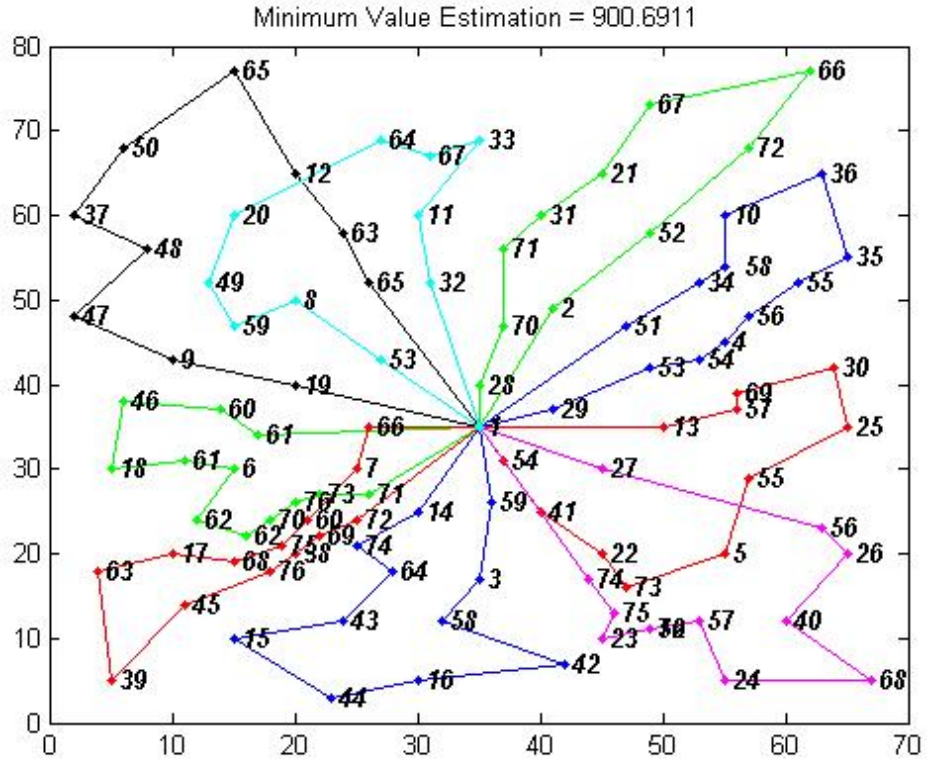
Διάγραμμα 3.3: par_2: n=76, Q=140, max_tour_length=99999, service_time=0



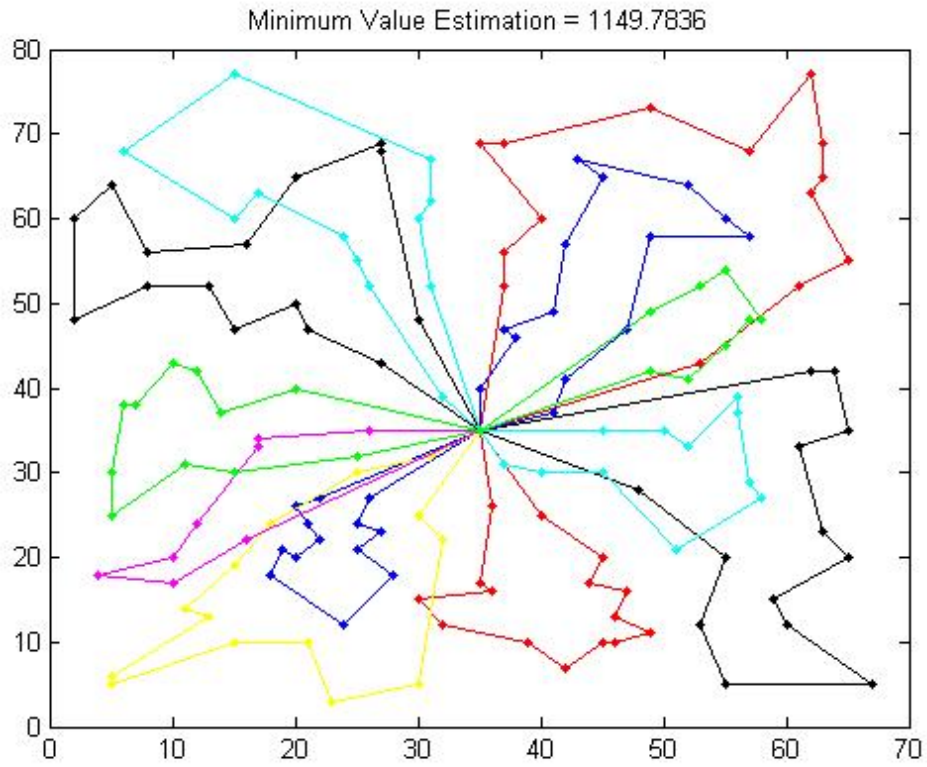
Διάγραμμα 3.4: par_7: n=76, Q=140, max_tour_length=160, service_time=10



Διάγραμμα 3.5: par_3: n=101, Q=200, max_tour_length=99999, service_time=0



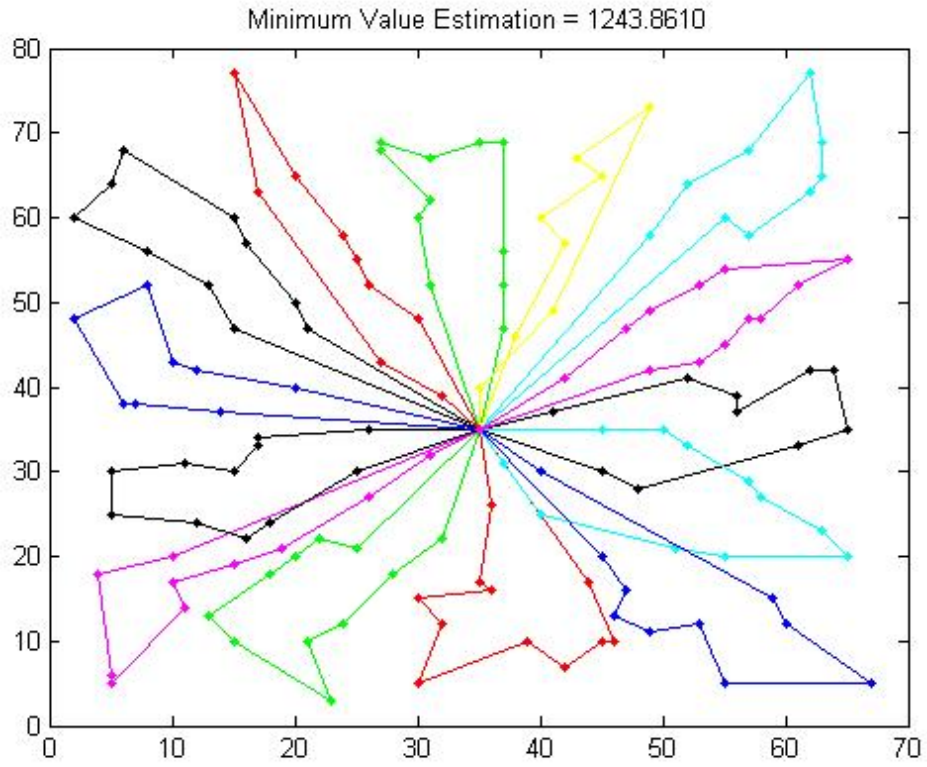
Διάγραμμα 3.6: par_8: n=101, Q=200, max_tour_length=230, service_time=10



Διάγραμμα 3.7: par_4: n=151, Q=200, max_tour_length=99999, service_time=0

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

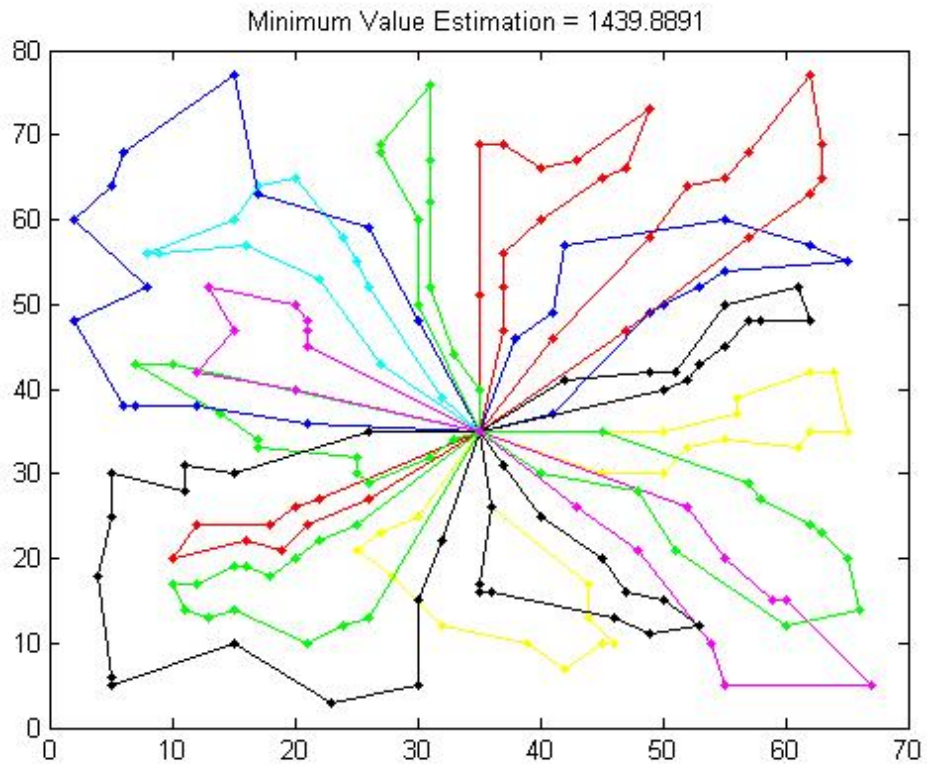
Πίνακας 3.2: Διαδρομή - par_4: n=151



Διάγραμμα 3.8: par_9: n=151, Q=200, max_tour_length=200, service_time=10

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

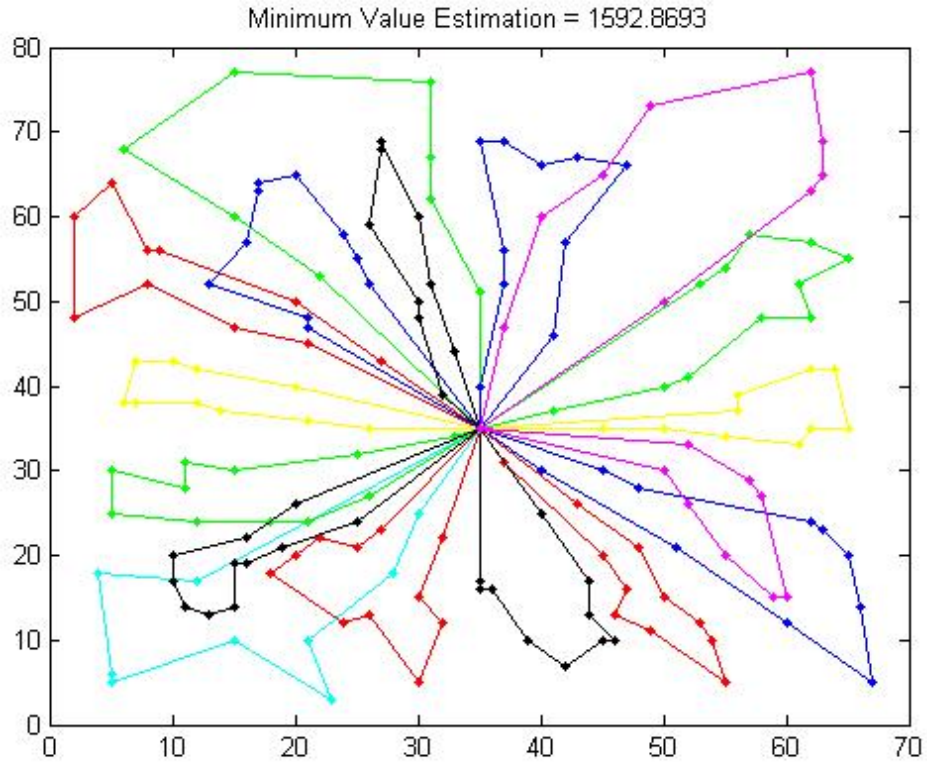
Πίνακας 3.3: Διαδρομή - par_9: n=151



Διάγραμμα 3.9: par_5: n=200, Q=200, max_tour_length=99999, service_time=0

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

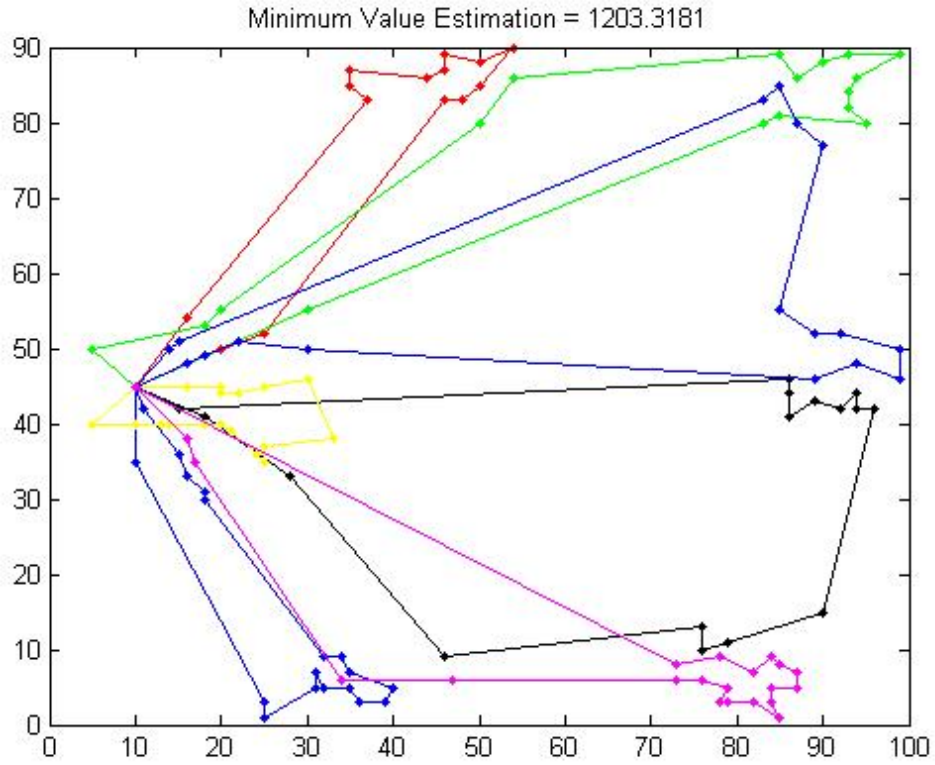
Πίνακας 3.4: Διαδρομή - par_5: n=201



Διάγραμμα 3.10: par_10: n=200, Q=200, max_tour_length=200, service_time=10

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

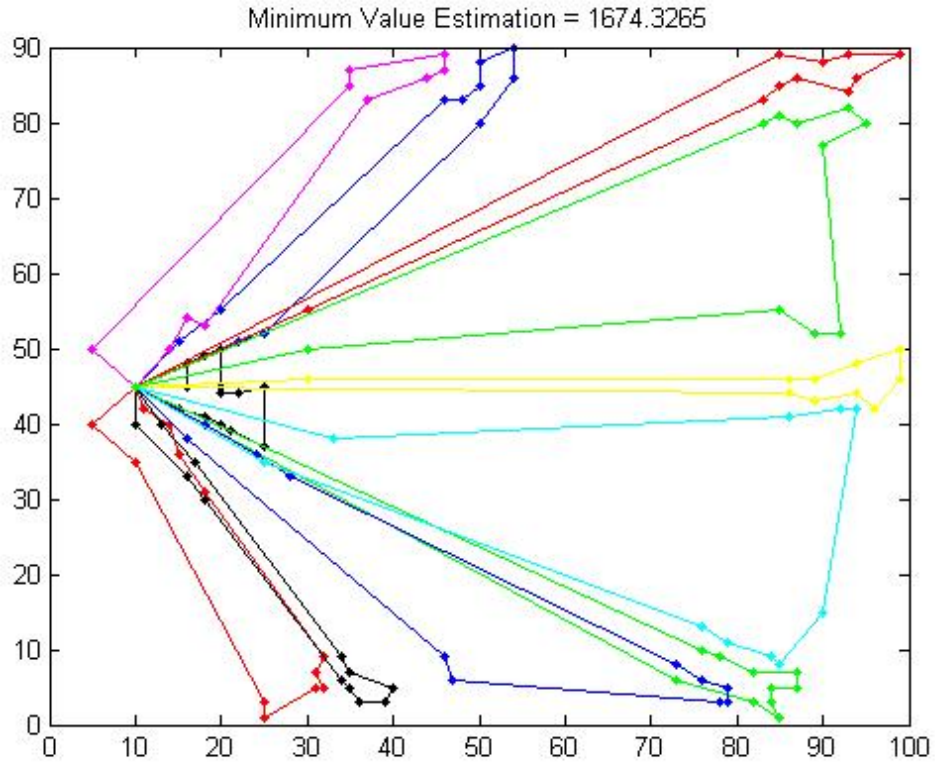
Πίνακας 3.5: Διαδρομή - par_10: n=201



Διάγραμμα 3.11: par11: n=121, Q=200, max_tour_length=99999, service_time=0

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

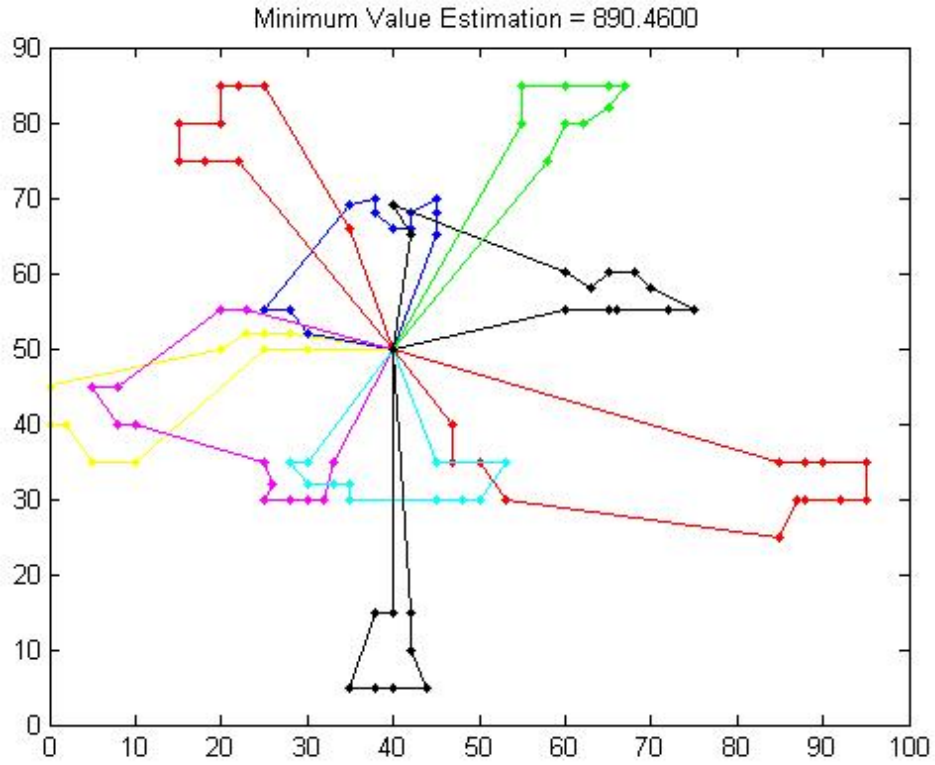
Πίνακας 3.6: Διαδρομή **- par11:** n=121



Διάγραμμα 3.12: par_13: n=121, Q=200, max_tour_length=720, service_time=50

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

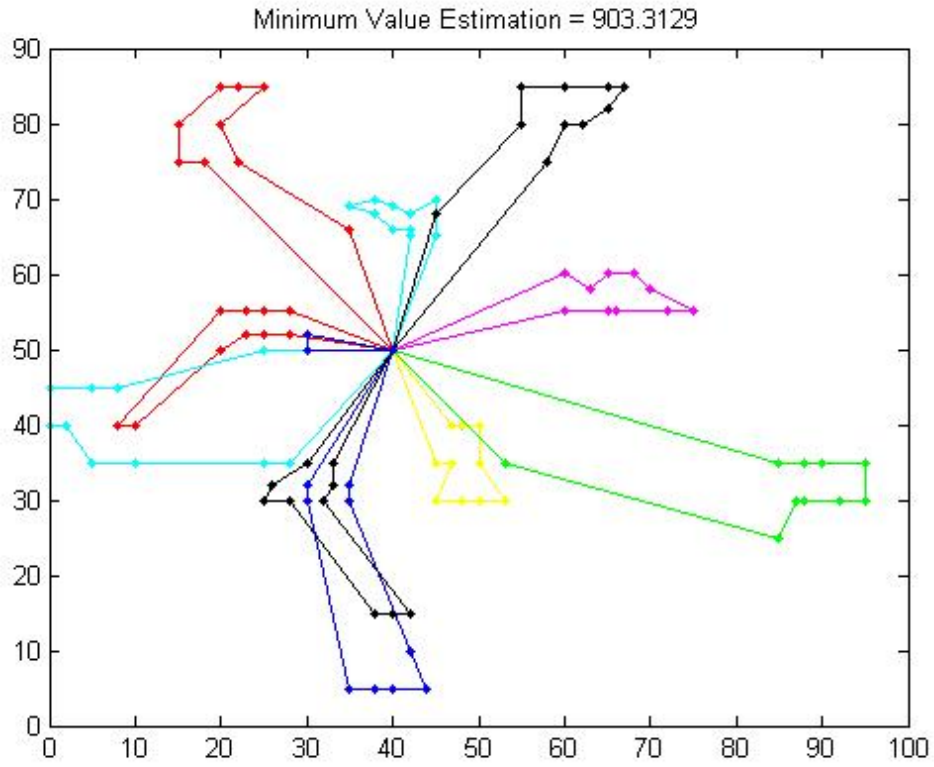
Πίνακας 3.7: Διαδρομή - par_13: n=121



Διάγραμμα 3.13: par_12: n=100, Q=200, max_tour_length=99999, service_time=0

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

Πίνακας 3.8: Διαδρομή - par_12: n=100



Διάγραμμα 3.14: par_14 – n=100, Q=200, max_tour_length=1040, service_time=90

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

Πίνακας 3.9: Διαδρομή - par_14 – n=100

File-n	*Best Cost*	*GA*	*iter*	*Απόκλιση*
par1-51	**524,61**	539,68	1388	2,87%
par2-76	**835,26**	884,48	2151	5,89%
par3-101	**826,14**	856,86	3850	3,72%
par4-151	**1028,42**	1149,80	9927	11,80%
par5-200	**1291,29**	1439,90	29758	11,51%
par6-51	**555,43**	566,02	1429	1,91%
par7-76	**909,68**	954,34	2464	4,91%
par8-101	**865,94**	900,69	4977	4,01%
par9-151	**1162,55**	1243,90	12969	7,00%
par10-200	**1395,55**	1592,90	37616	14,14%
par11-121	**1042,11**	1203,00	3226	15,44%
par12-101	**819,56**	890,46	2747	8,65%
par13-121	**1541,19**	1674,30	5793	8,64%
par14-101	**866,37**	903,31	2177	4,26%

Πίνακας 3.10: Αποτελέσματα VRP με GA

Όπως προαναφέρθηκε τα αποτελέσματα παρουσιάζονται ανά δύο σύμφωνα με τα δεδομένα τους. Έτσι στο πρώτο γράφημα δεν ισχύει ο περιορισμός του max_tour_length καθώς όχημα δεν έχει κάποιο χρονικό όριο στο οποίο πρέπει να εκτελέσει τη διαδρομή και δεν υπάρχει service_time ενώ στο δεύτερο γράφημα ισχύει ο περιορισμός αυτός.

Συγκρίνοντας τα δύο γραφήματα αυτό που είναι εμφανές είναι ότι στην περίπτωση όπου ισχύει ο περιορισμός το όχημα θα πρέπει να επιστρέψει περισσότερες φορές στην αποθήκη. Αυτό συμβαίνει γιατί παραβιάζεται ο περιορισμός της μέγιστης διαδρομής και όχι της χωρητικότητας του φορτηγού. Ο περιορισμός της μέγιστης διαδρομής μπορεί να μεταφραστεί είτε ως το χρονικό όριο εργασίας του οδηγού είτε ως τον ανεφοδιασμό καυσίμων του οχήματος.

Το βέλτιστο κόστος στην περίπτωση όπου ισχύει ο περιορισμός της μέγιστης διαδρομής είναι λογικά μεγαλύτερο αφού το όχημα εκτελεί περισσότερες διαδρομές.

Ο αλγόριθμος παρουσιάζει καλύτερα αποτελέσματα στα μικρά προβλήματα, η απόκλιση από τη βέλτιστη λύση είναι μικρότερη, ανεξάρτητα από το αν ισχύει ο περιορισμός ή όχι. Βέβαια στα προβλήματα όπου ισχύει ο περιορισμός της βέλτιστης διαδρομής (par_6, par_7, par_8, par_9, par_10, par_12, par_14) ο αλγόριθμος χρειάζεται να εκτελέσει περισσότερες επαναλήψεις για να πλησιάσει τη βέλτιστη τιμή.

3.6 Επίλυση SVRP με χρήση GA

Σε αυτήν την ενότητα επιλύεται το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση με χρήση του *αλγόριθμου 3.1* και τοπική αναζήτηση 3-opt. Η αντικειμενική συνάρτηση προκύπτει από τις Εξ. (2.10), (2.11), (2.12) τροποποιημένη ώστε να υπολογίζει το κόστος για συγκεκριμένη χωρητικότητα του οχήματος. Επίσης η αντικειμενική συνάρτηση δεν υπολογίζει το κόστος για όλες τις πιθανές τιμές της ζήτησης (0<d<ζήτηση πελάτη), αλλά για συγκεκριμένες τιμές που ορίζουμε.

Για να είναι δυνατή η σύγκριση με το απλό πρόβλημα δρομολόγησης οχημάτων αρχικά επιλύουμε το στοχαστικό πρόβλημα αλλά με συγκεκριμένη ζήτηση του κάθε πελάτη όπως αυτή δίνεται στο απλό πρόβλημα. Στη συνέχεια ορίζουμε τη μεταβλητή dem η οποία μας δίνει την απόκλιση από τη ζήτηση του κάθε πελάτη. Επιλύουμε τα προβλήματα για dem=±1, ±2. Επίσης δεν επιτρέπουμε η ζήτηση να πάρει την τιμή 0, γιατί σε αυτήν την περίπτωση το όχημα δεν θα πρέπει να επισκεφτεί τον πελάτη και ο συγκεκριμένος πελάτης θα πρέπει να βγει από τη διαδρομή.

Ο γενετικός αλγόριθμος που αναλύθηκε στις προηγούμενες ενότητες εκτελέστηκε για 5 διαφορετικά προβλήματα στα οποία δίνονται οι θέσεις των κόμβων στο καρτεσιανό επίπεδο, το πλήθος το κόμβων n, η χωρητικότητα του οχήματος διανομής Q, η μέγιστη διαδρομή που μπορεί να διανύσει το φορτηγό από τη στιγμή που θα εγκαταλείψει την αποθήκη. Τα στοιχεία αυτά παρουσιάζονται στον επόμενο πίνακα:

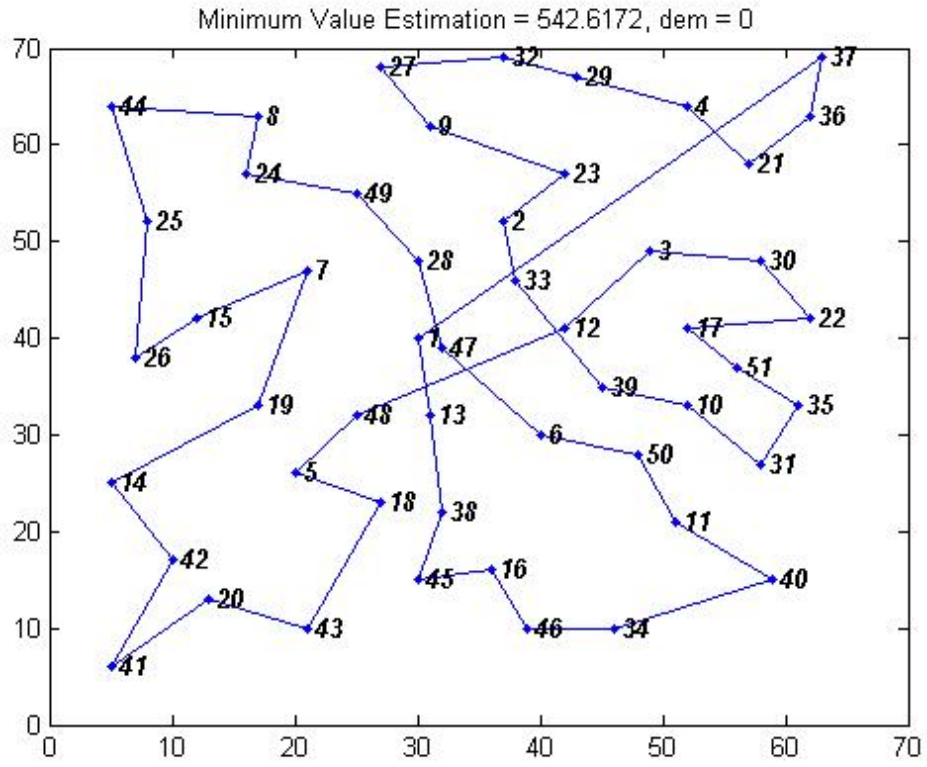
File-n	***n***	***Q***	***max_tour_length***	***service_time***
par1-51	51	160	999999	0
par2-76	76	140	999999	0
par3-101	101	200	999999	0

Πίνακας 3.11: Δεδομένα αρχείων

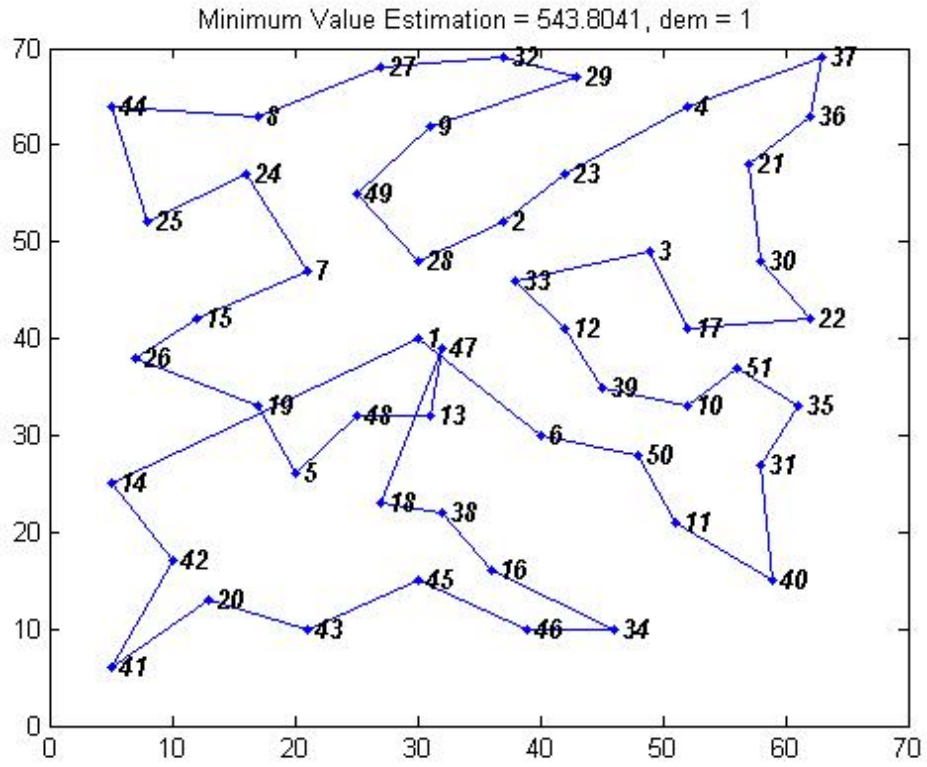
Στα προβλήματα που εξετάζουμε, όπου το max_tour_length είναι 999999 ουσιαστικά δεν λαμβάνεται υπόψιν ο περιορισμός της διάρκειας της διαδρομής και το μόνο που μας ενδιαφέρει είναι να μην υπερβεί η συνολική ζήτηση των πελατών τη χωρητικότητα του οχήματος σε μία διαδρομή και όπου το service_time είναι 0 δεν απαιτείται κάποιος χρόνος για την εξυπηρέτηση του πελάτη, συνεπώς και η διάρκεια της διαδρομής (το κόστος) είναι μικρότερο. Στο σύνολο των κόμβων n συμπεριλαμβάνεται και η αποθήκη.

3.6.1. Αποτελέσματα SVRP με χρήση GA

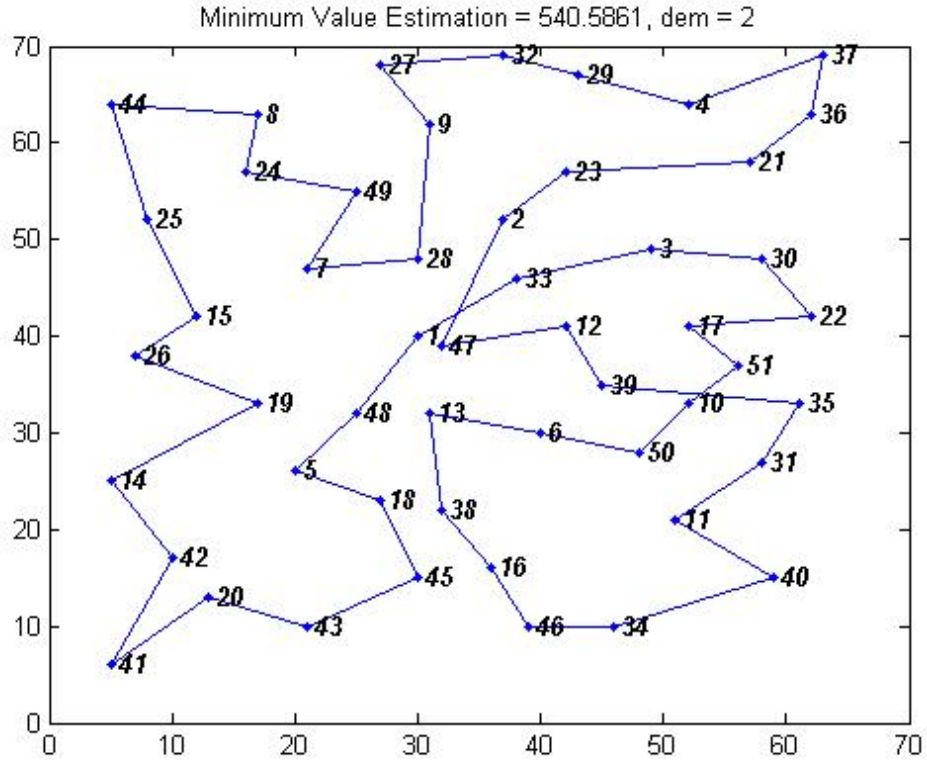
Στα διαγράμματα που ακολουθούν το όχημα δεν επιστρέφει στην αποθήκη και η διαδρομή είναι συνεχόμενη. Αυτό συμβαίνει γιατί όταν η ζήτηση δεν είναι γνωστή εκ των προτέρων και ενδέχεται να μεταβληθεί οποιαδήποτε στιγμή δεν είναι δυνατόν να γνωρίζουμε πότε ακριβώς το όχημα θα επιστρέψει στην αποθήκη για ανεφοδιασμό.



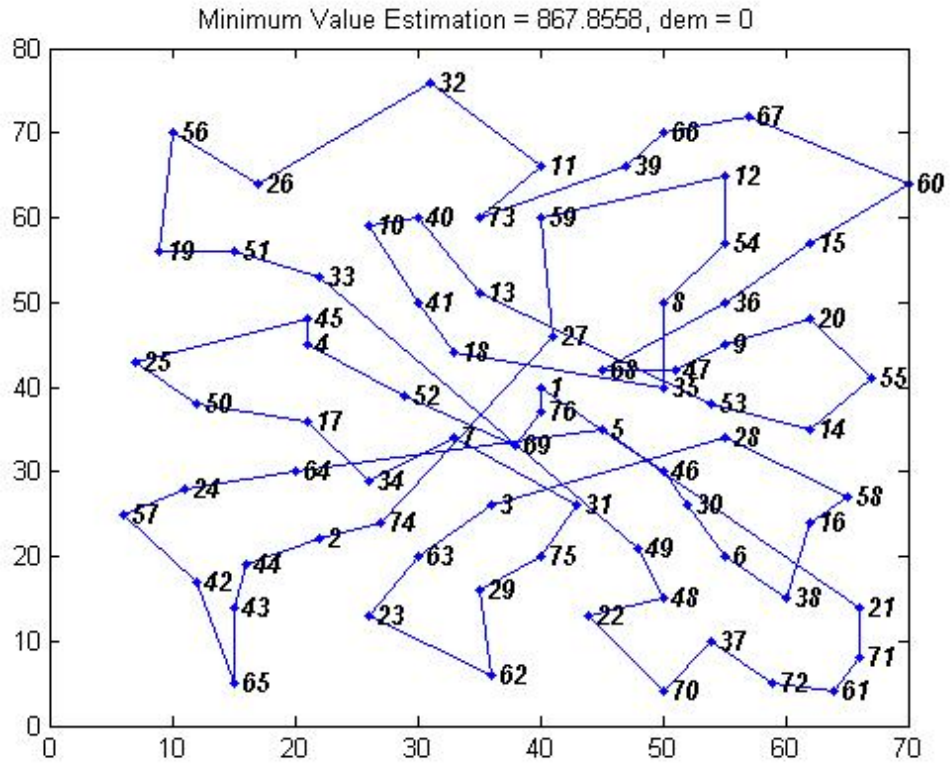
Διάγραμμα 3.15: par_1: n=51, dem=0, Q=160



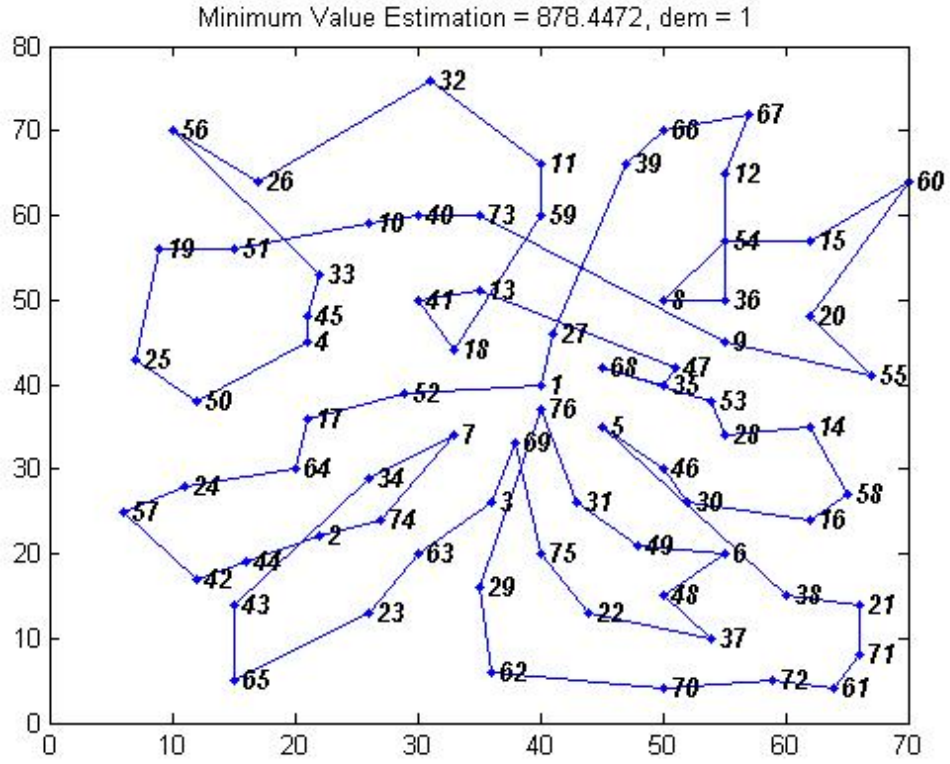
Διάγραμμα 3.16: par_1: n=51, dem=1, Q=160



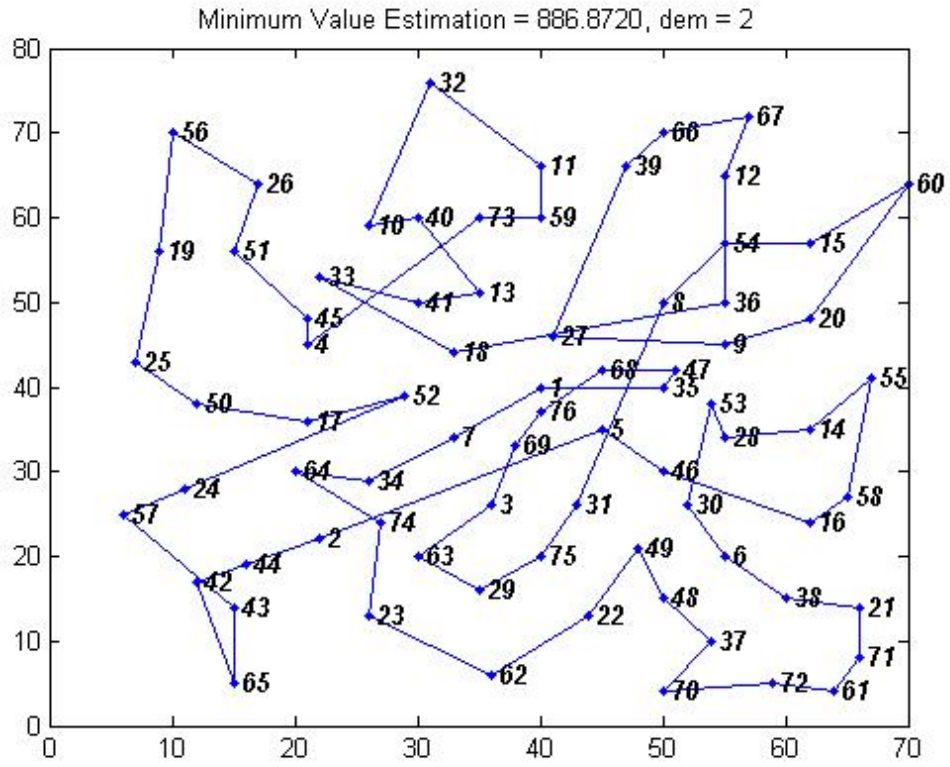
Διάγραμμα 3.17: par_1: n=51, dem=2, Q=160



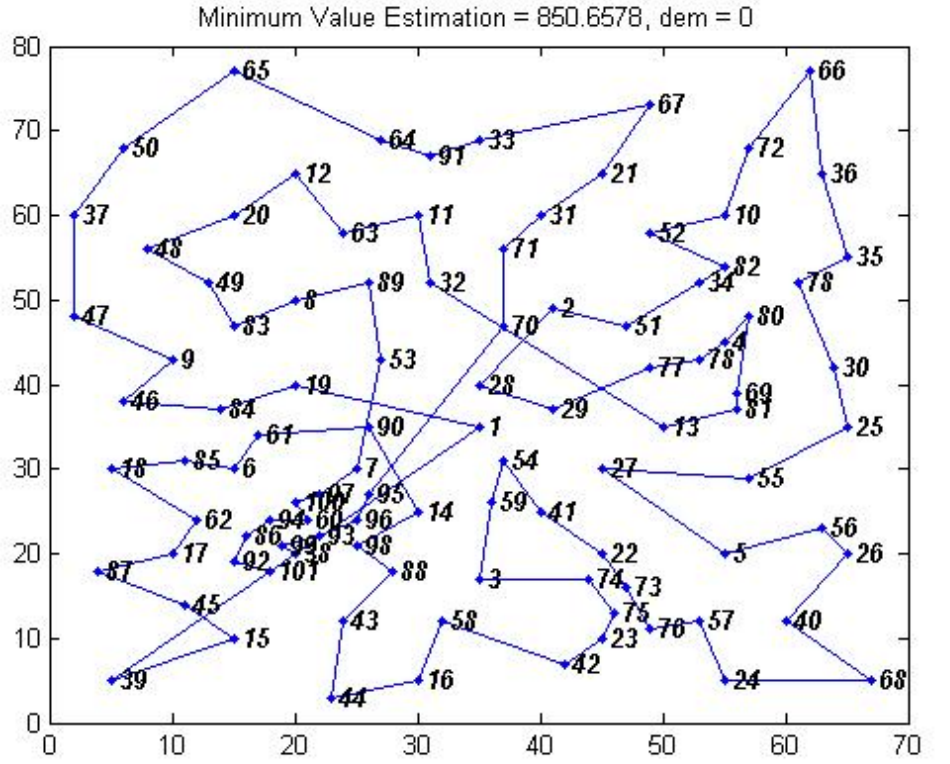
Διάγραμμα 3.18: par_2: n=76, dem=0, Q=150



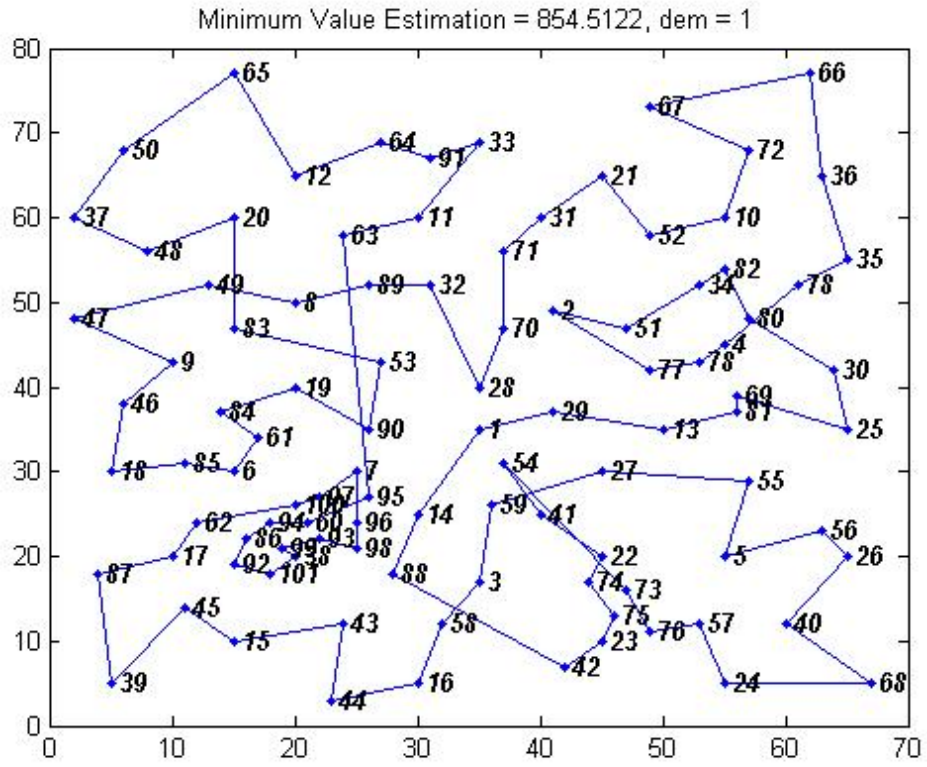
Διάγραμμα 3.19: par_2: n=76, dem=1, Q=150



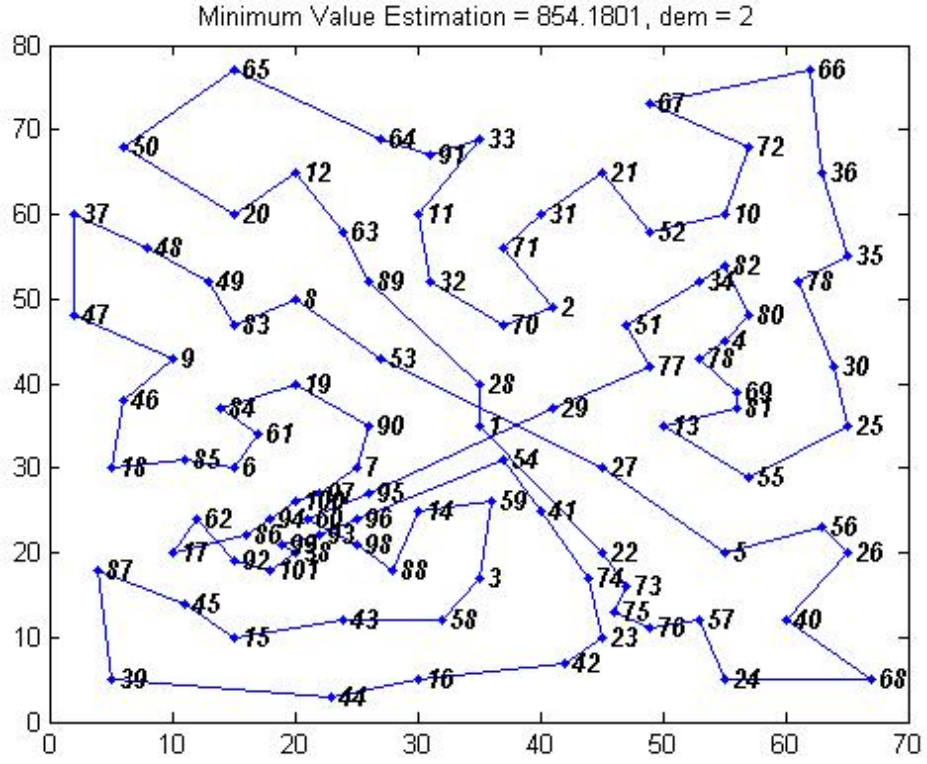
Διάγραμμα 3.20: par_2: n=76, dem=2, Q=150



Διάγραμμα 3.21: par_3: n=101, dem=0, Q=200



Διάγραμμα 3.22: par_3: n=101, dem=1, Q=200



Διάγραμμα 3.23: par_3: n=101, dem=2, Q=200

File-n	*dem*	*Best Cost*	*GA*	*iter*	*απόκλιση*
par1 - 51	0	**524,61**	542,6172	489	**3,43%**
	1		543,8041	643	**3,66%**
	2		540,5861	486	**3,05%**
par2 -76	0	**835,26**	867,8558	1158	**3,90%**
	1		878,4472	2364	**5,17%**
	2		886,872	1194	**6,18%**
par3 - 101	0	**826,14**	850,6578	2269	**2,97%**
	1		854,5122	2401	**3,43%**
	2		854,1801	1658	**3,39%**

Πίνακας 3.12: Αποτελέσματα SVRP με GA

Να επισημάνουμε ότι η λύση που εντοπίζεται από αυτόν τον αλγόριθμο, όταν η ζήτηση είναι συγκεκριμένη (dem=0), μέσο της αντικειμενικής συνάρτησης του στοχαστικού προβλήματος, το αποτέλεσμα είναι ίδιο με αυτό που θα έδινε η συγκεκριμένη διαδρομή μεσώ του απλού προβλήματος δρομολόγησης οχημάτων. Με αυτό τον τρόπο επαληθεύεται η ορθότητα της αντικειμενικής συνάρτησης του στοχαστικού προβλήματος. Επίσης να πούμε ότι στο στοχαστικό πρόβλημα η επιλογή του οχήματος για το αν θα συνεχίσει στον επόμενο πελάτη ή θα επιστρέψει στην αποθήκη για ανεφοδιασμό εξαρτάται από το κόστος μετάβασης στην κάθε περίπτωση και όχι από το απόθεμα του οχήματος. Αυτό δε σημαίνει ότι παραβιάζεται ο περιορισμός της χωρητικότητας του οχήματος. Για παράδειγμα αν η ζήτηση του επόμενου πελάτη είναι μεγαλύτερη από το απόθεμα, το όχημα θα χρεωθεί την απόσταση μεταξύ των δύο πελατών αλλά και τη διαδρομή πελάτης-αποθήκη-πελάτης καθώς όταν φτάσει στον πελάτη θα χρειαστεί να επιστρέψει στην αποθήκη για ανεφοδιασμό. Στην περίπτωση αυτή βάση του κόστους το όχημα θα επιλέξει να επιστρέψει στην αποθήκη πριν επισκεφτεί τον επόμενο πελάτη.

3.7 Επίλυση TSP & PTSP με χρήση GA

Σε αυτήν την ενότητα επιλύεται το στοχαστικό πρόβλημα του πλανόδιου πωλητή με χρήση του *αλγόριθμου 3.1* και τοπική αναζήτηση 3-opt Η αντικειμενική συνάρτηση προκύπτει από την Εξ. (2.7), (2.8) και (2.9).

Ο γενετικός αλγόριθμος που αναλύθηκε στις προηγούμενες ενότητες δοκιμάστηκε για 5 διαφορετικά προβλήματα στα οποία δίνονται οι θέσεις των κόμβων στο καρτεσιανό επίπεδο και το πλήθος το κόμβων n. Το κάθε πρόβλημα επιλύθηκε για διαφορετική πιθανότητα επίσκεψης του κάθε κόμβου και στην περίπτωση που η πιθανότητα είναι 1 ουσιαστικά επιλύεται το TSP ενώ στις υπόλοιπες περιπτώσεις το PTSP. Τα στοιχεία αυτά παρουσιάζονται στον επόμενο πίνακα:

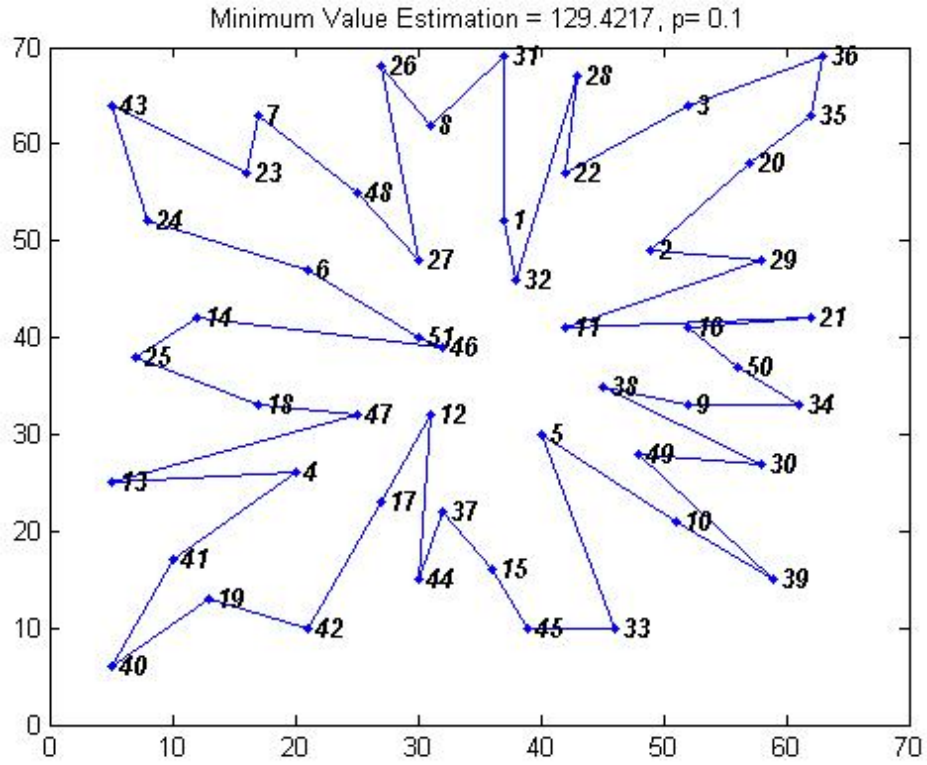
File-n	***n***
eil 51	51
eil101	101
kroA100	100
ch150	150
d198	198

Πίνακας 3.13: Δεδομένα αρχείων

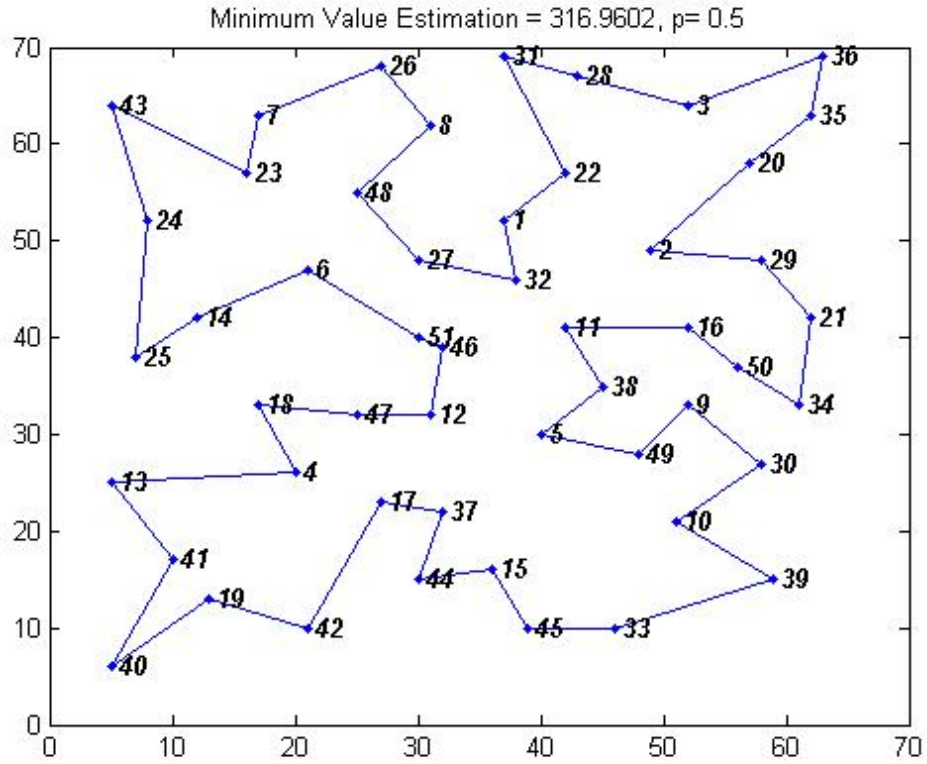
3.7.1. Αποτελέσματα TSP & PTSP με χρήση GA

Σε αυτήν την ενότητα θα παρουσιαστούν τα αποτελέσματα που προέκυψαν από την επίλυση του απλού προβλήματος του πλανόδιου πωλητή (όταν p=1) και του στοχαστικού (για p=0.1, p=0.5, p=1) με χρήση γενετικού αλγόριθμου.

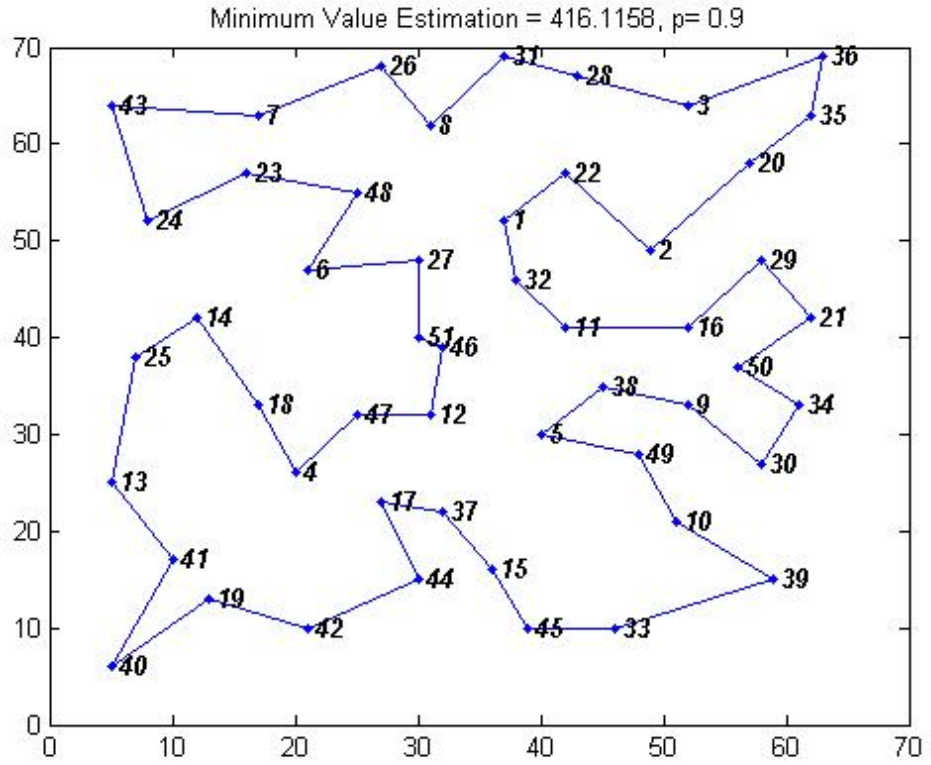
Στα διαγράμματα που ακολουθούν φαίνεται η διαδρομή που πρέπει να κάνει ο πωλητής ώστε να επισκεφτεί όλες τις πόλεις με το μικρότερο δυνατό κόστος.



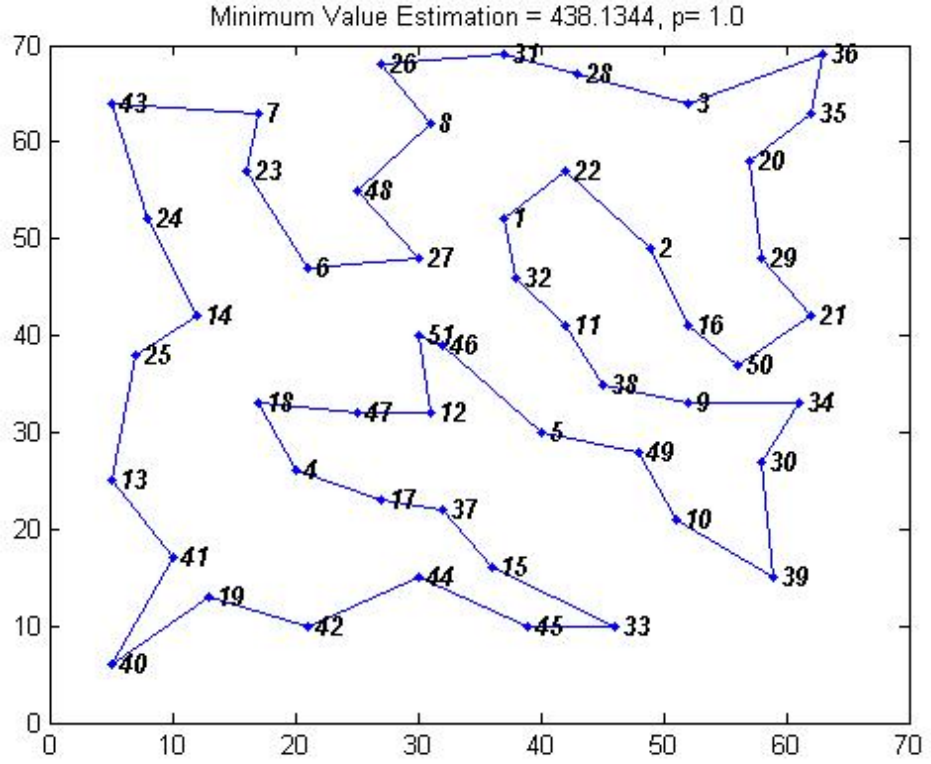
Διάγραμμα 3.24: eil51 - n=51, p=0.1



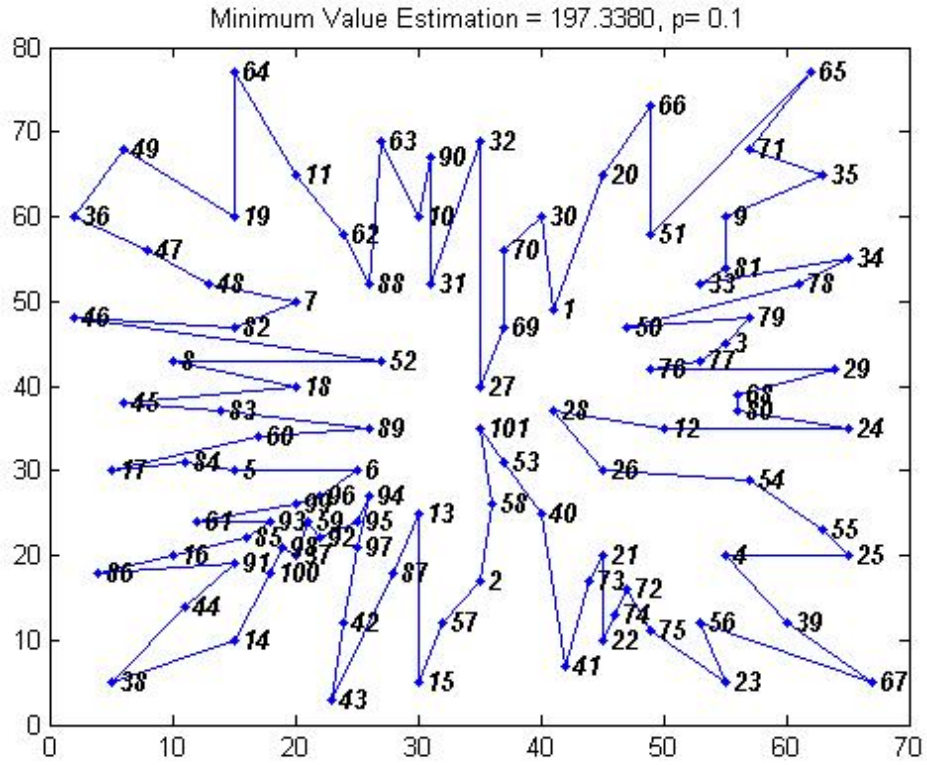
Διάγραμμα 3.25: eil51 - n=51, p=0.5



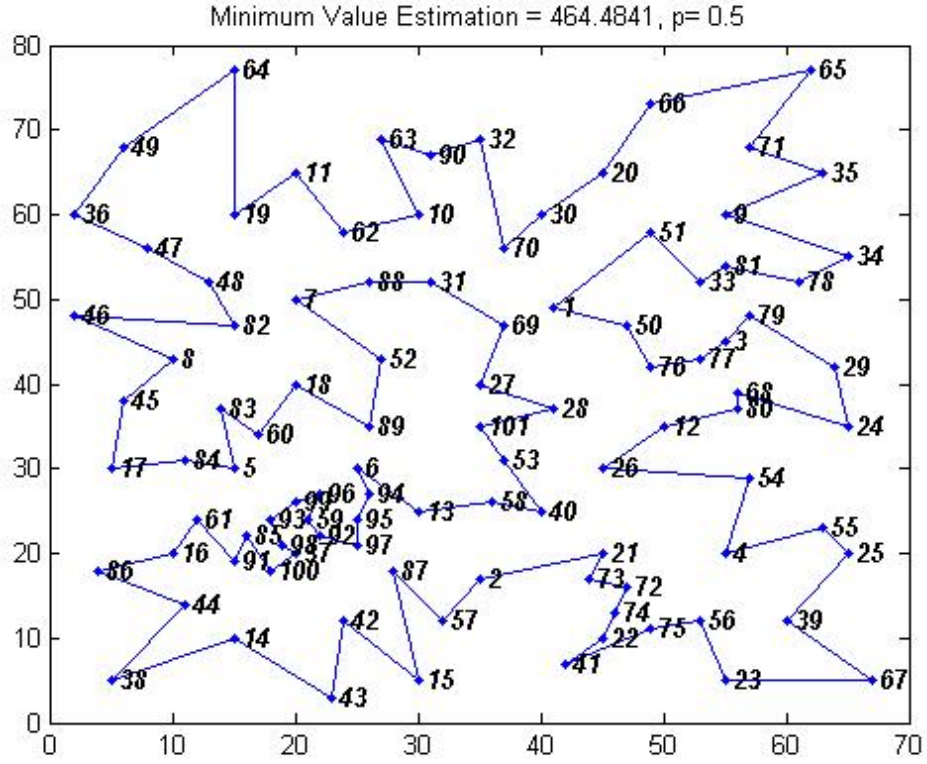
Διάγραμμα 3.26: eil51 - n=51, p=0.9



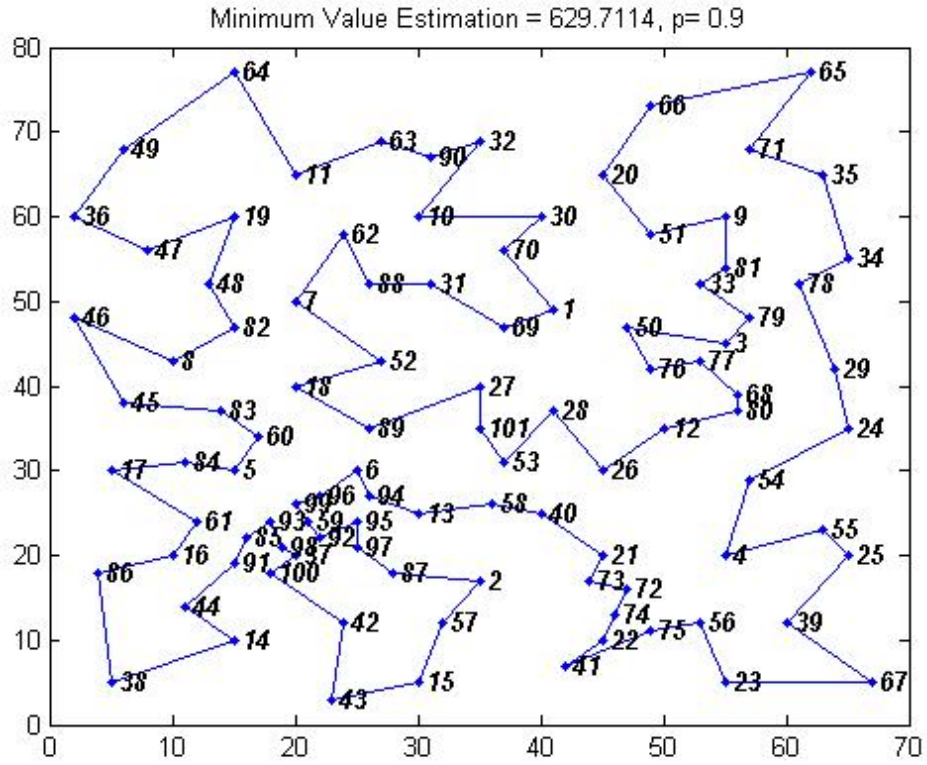
Διάγραμμα 3.27: eil51 - n=51, p=1.0



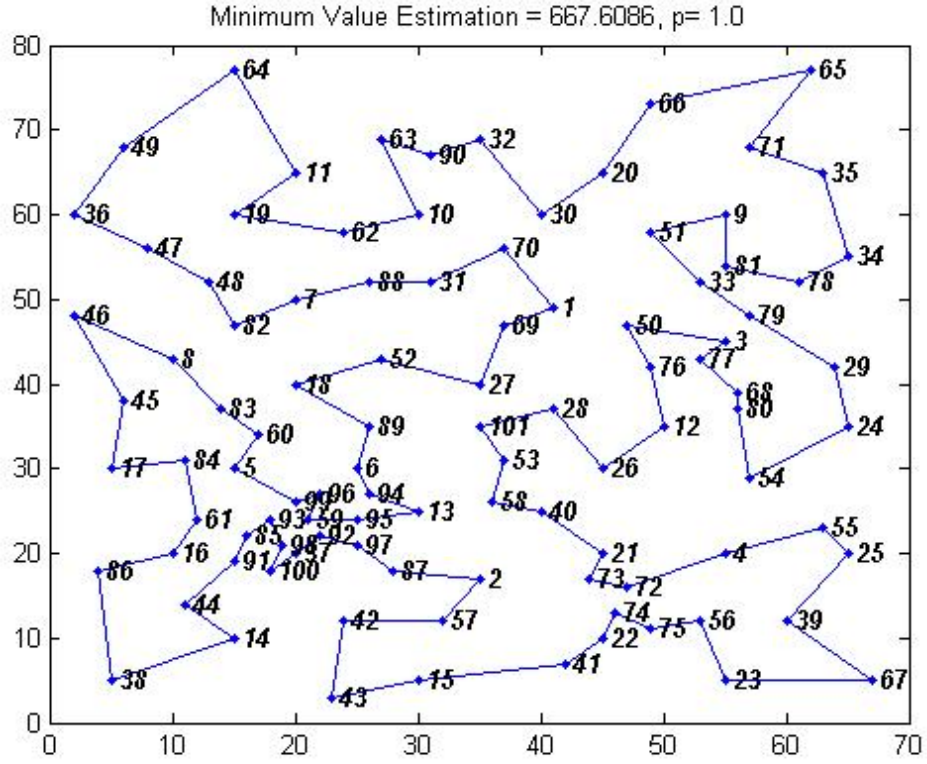
Διάγραμμα 3.28: eil101 - n=101, p=0.1



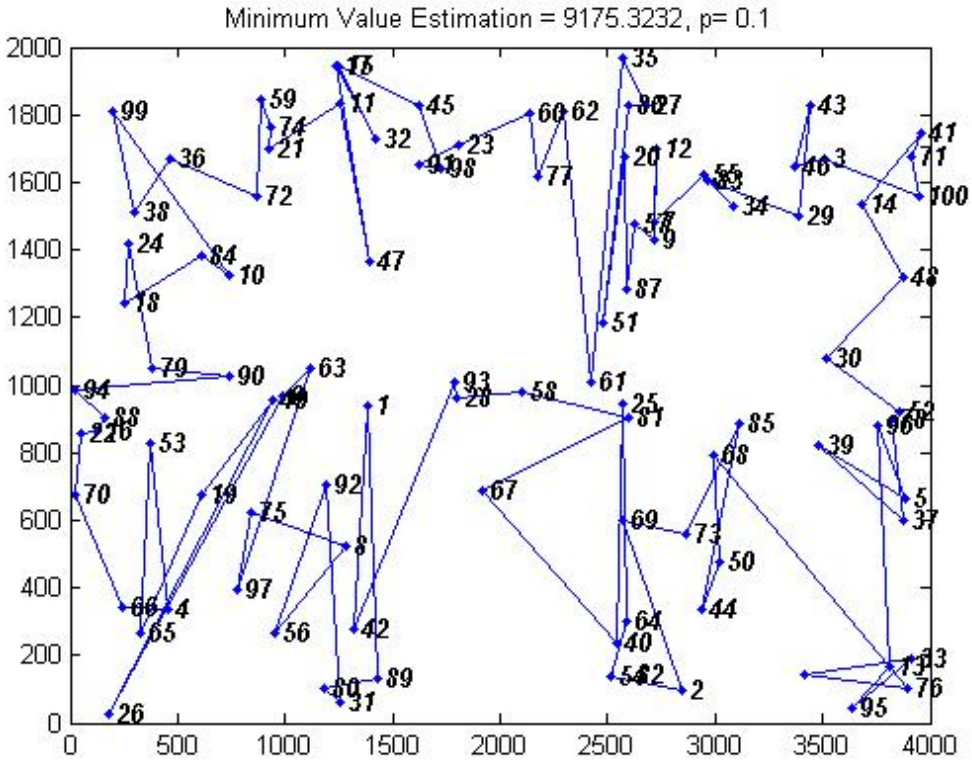
Διάγραμμα 3.29: eil101 - n=101, p=0.5



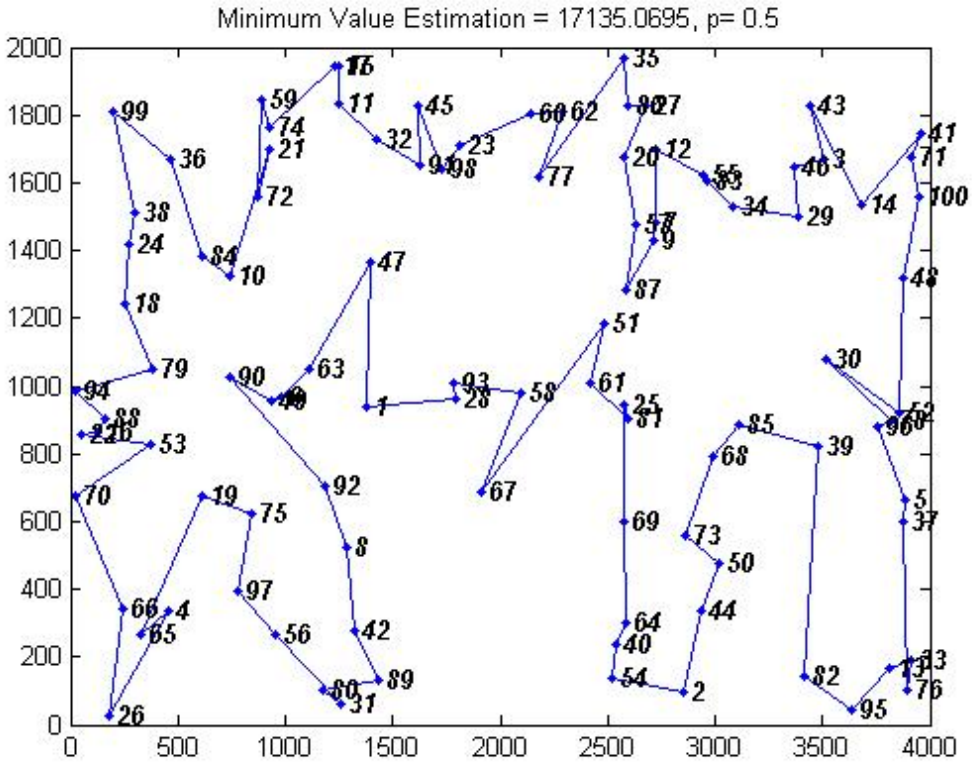
Διάγραμμα 3.30: eil101 - n=101, p=0.9



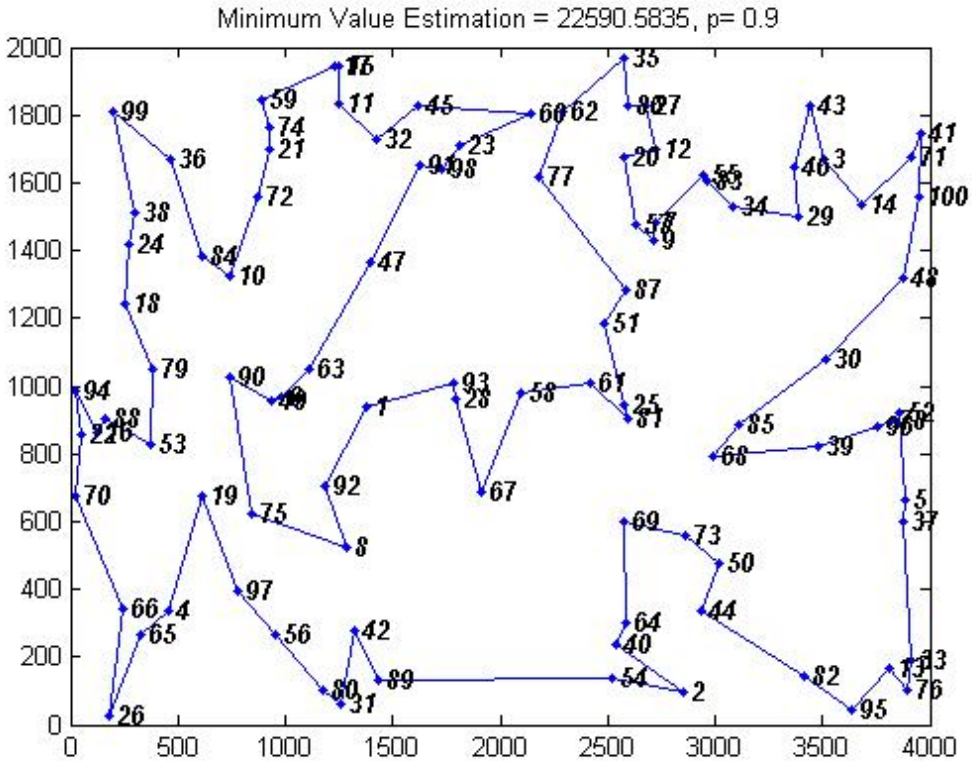
Διάγραμμα 3.31: eil101 - n=101, p=1.0



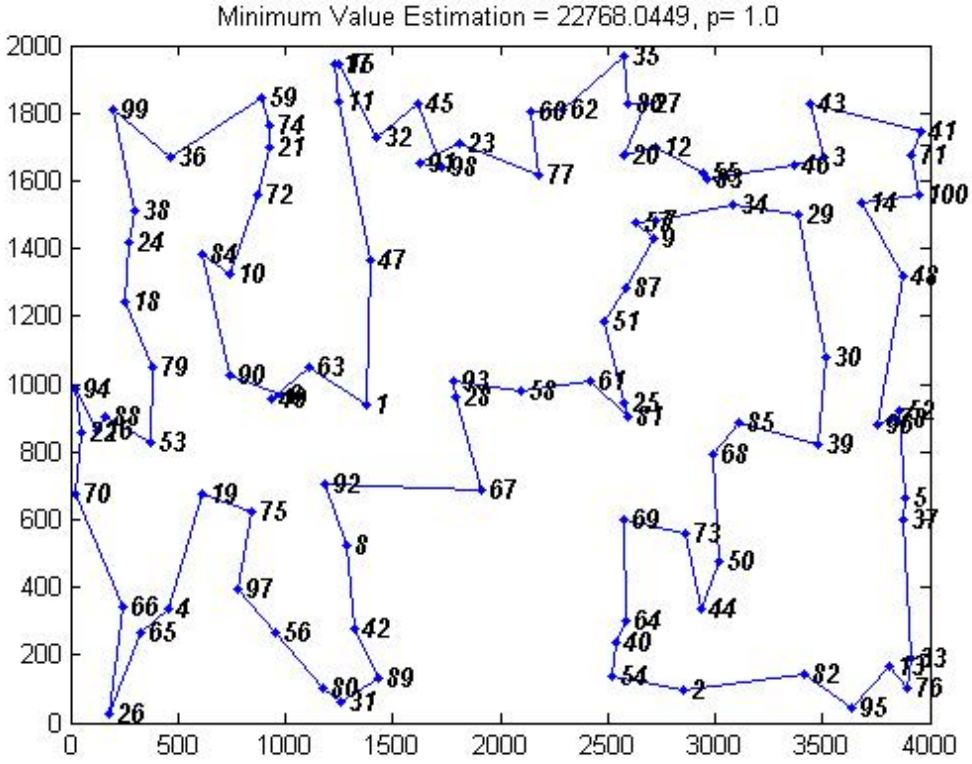
Διάγραμμα 3.32: kroA100 – n=100, p=0.1



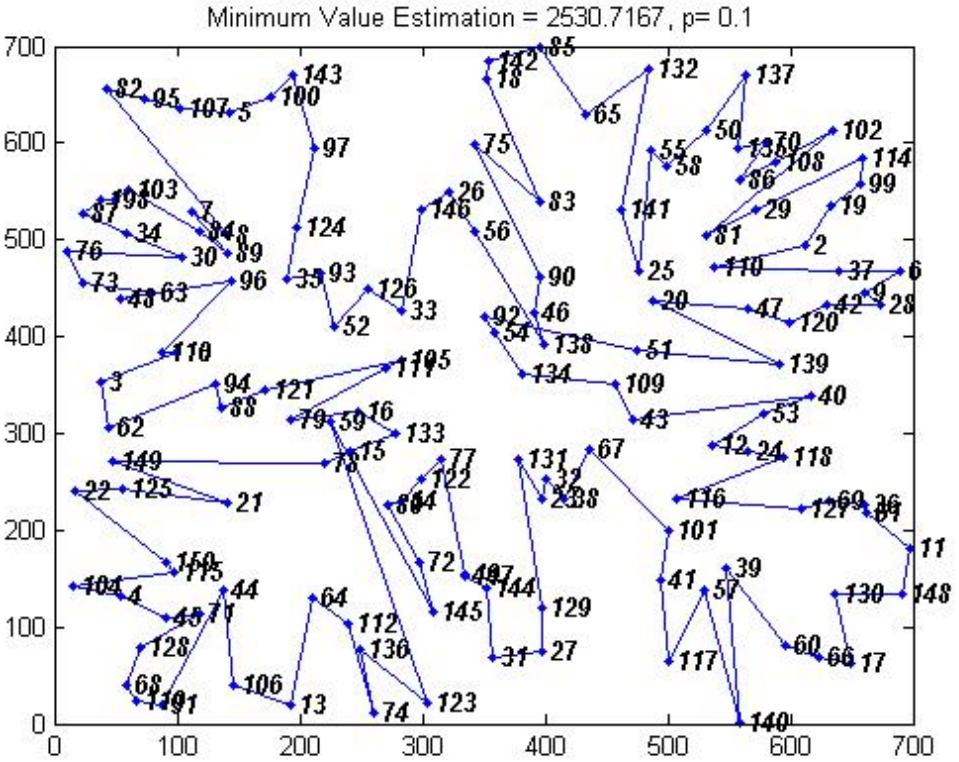
Διάγραμμα 3.33: kroA100 – n=100, p=0.5



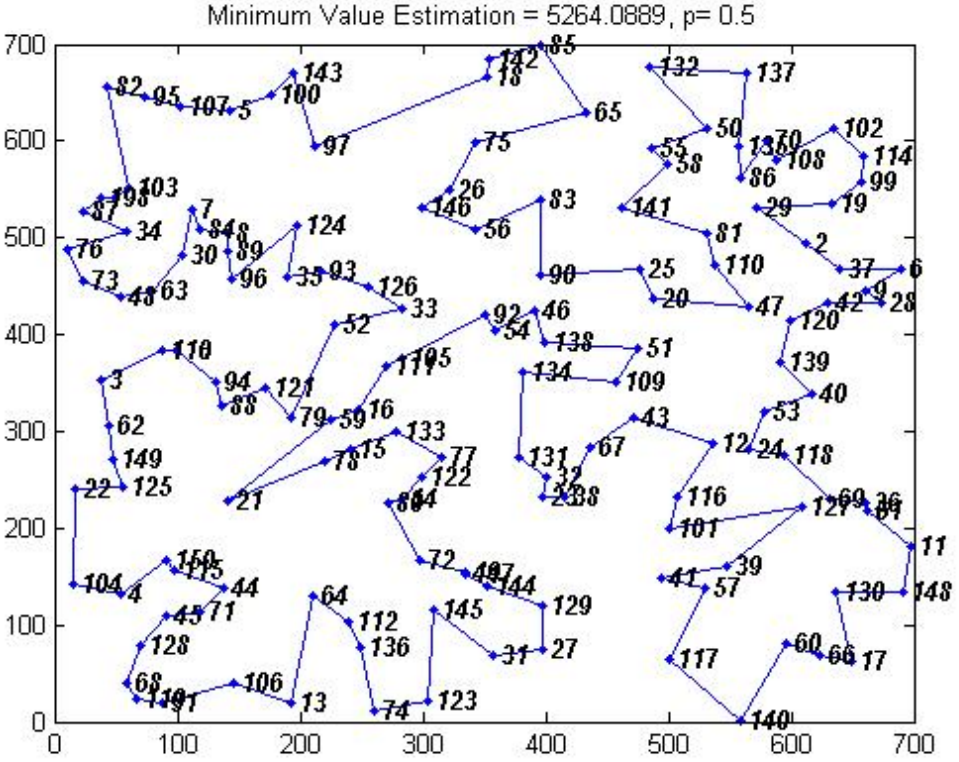
Διάγραμμα 3.34: kroA100 – n=100, p=0.9



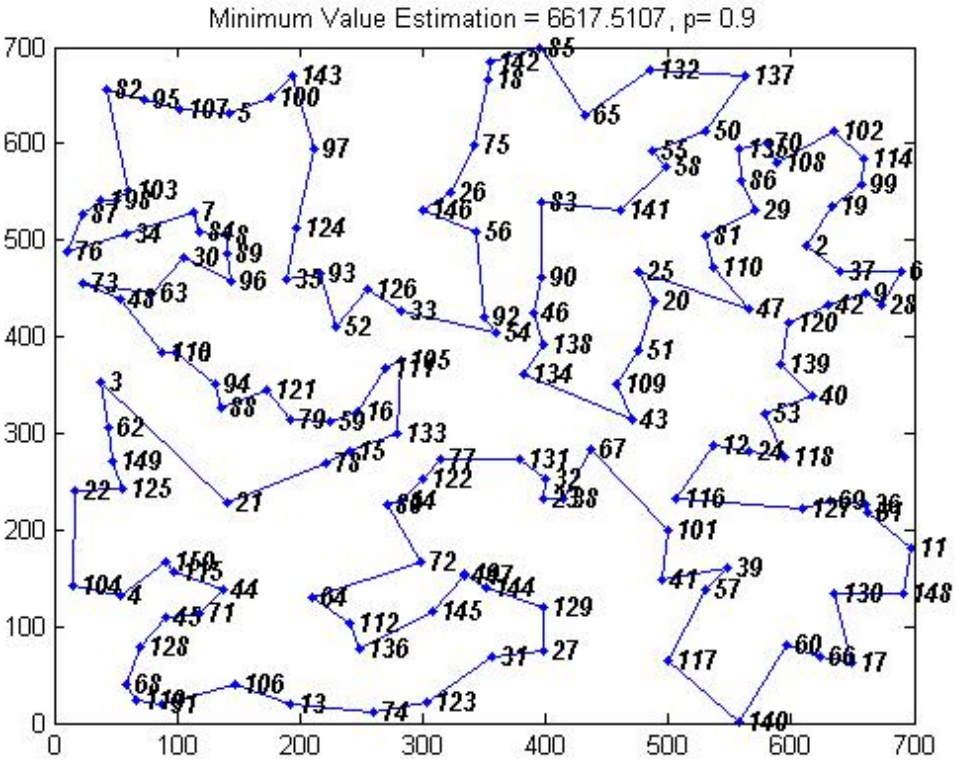
Διάγραμμα 3.35: kroA100 – n=100, p=1.0



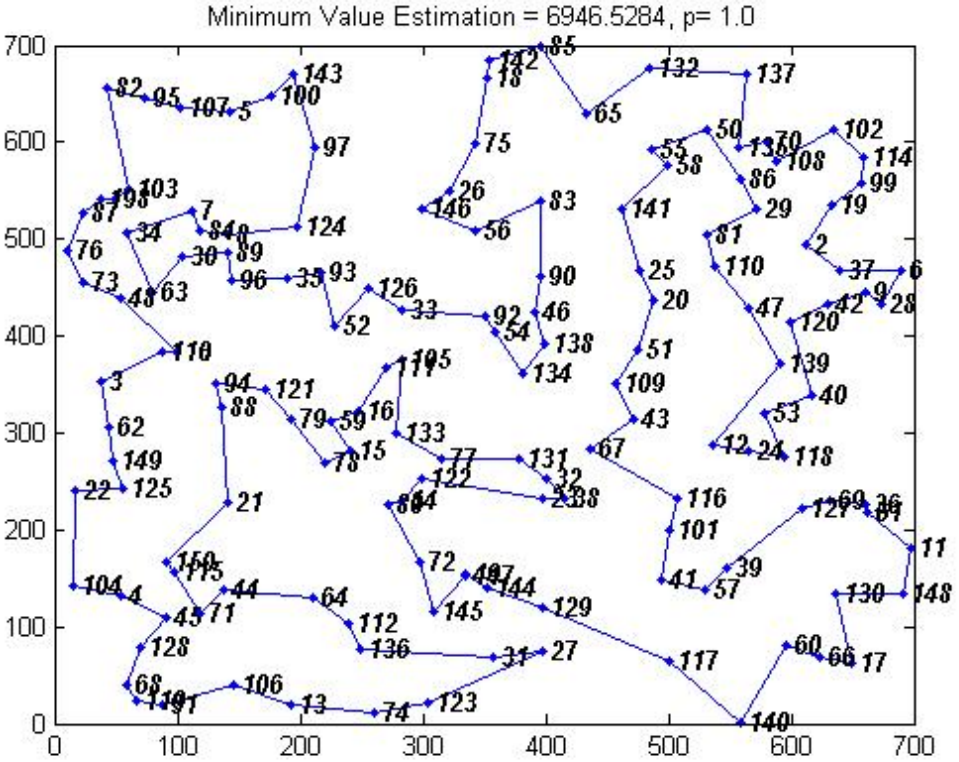
Διάγραμμα 3.36: ch150 – n=150, p=0.1



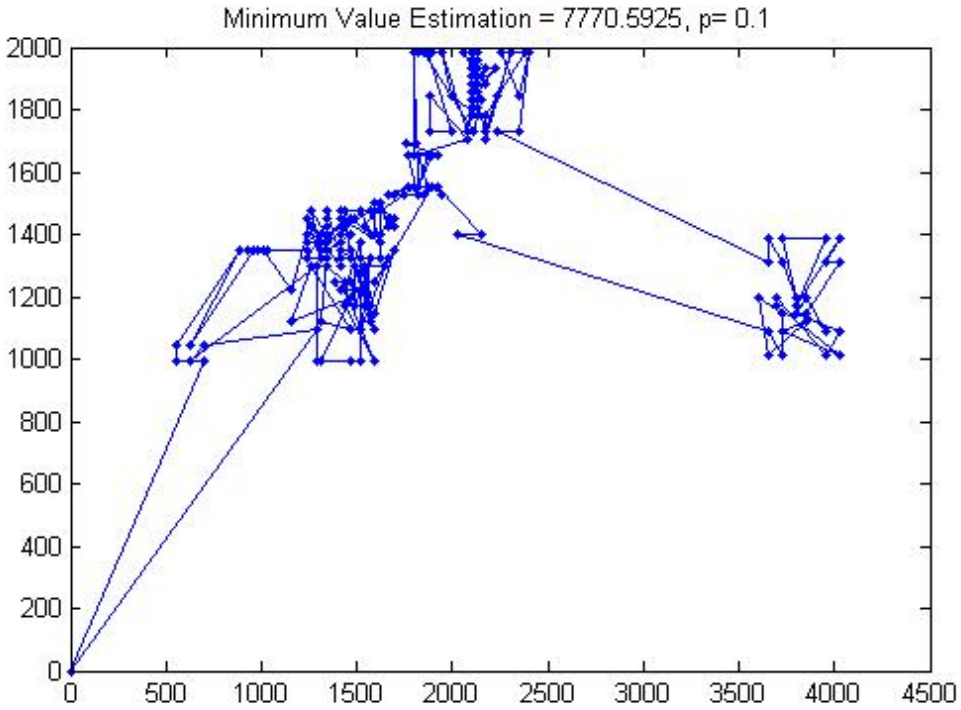
Διάγραμμα 3.37: ch150 – n=150, p=0.5



Διάγραμμα 3.38: ch150 – n=150, p=0.9



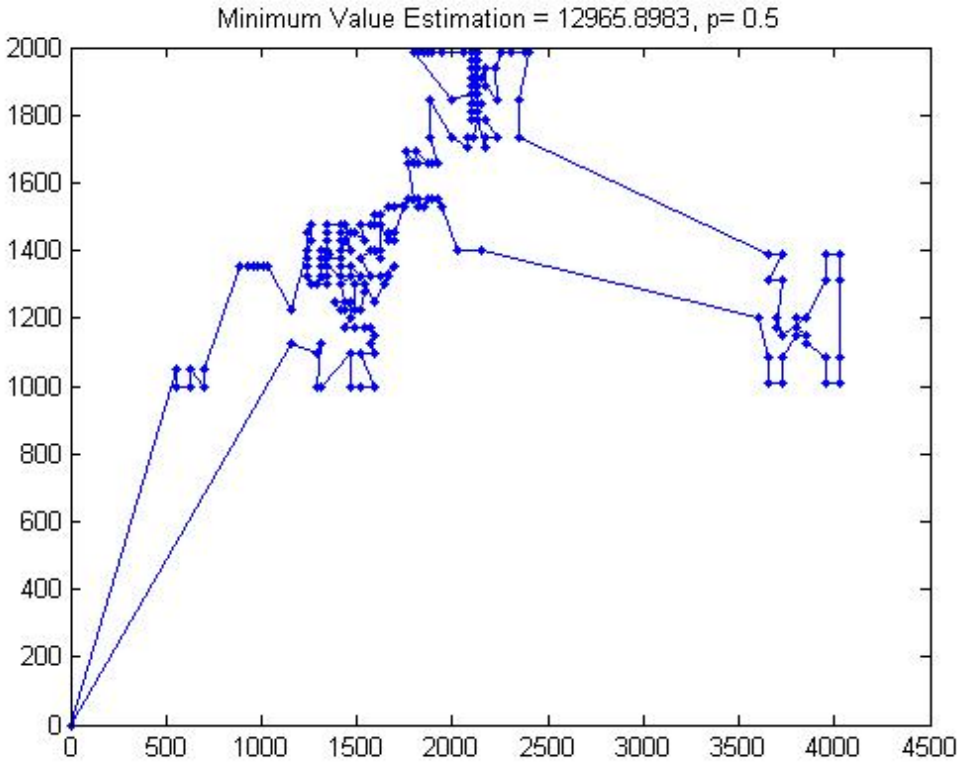
Διάγραμμα 3.39: ch150 – n=150, p=1.0



Διάγραμμα 3.40: d198 – n=198, p=0.1

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

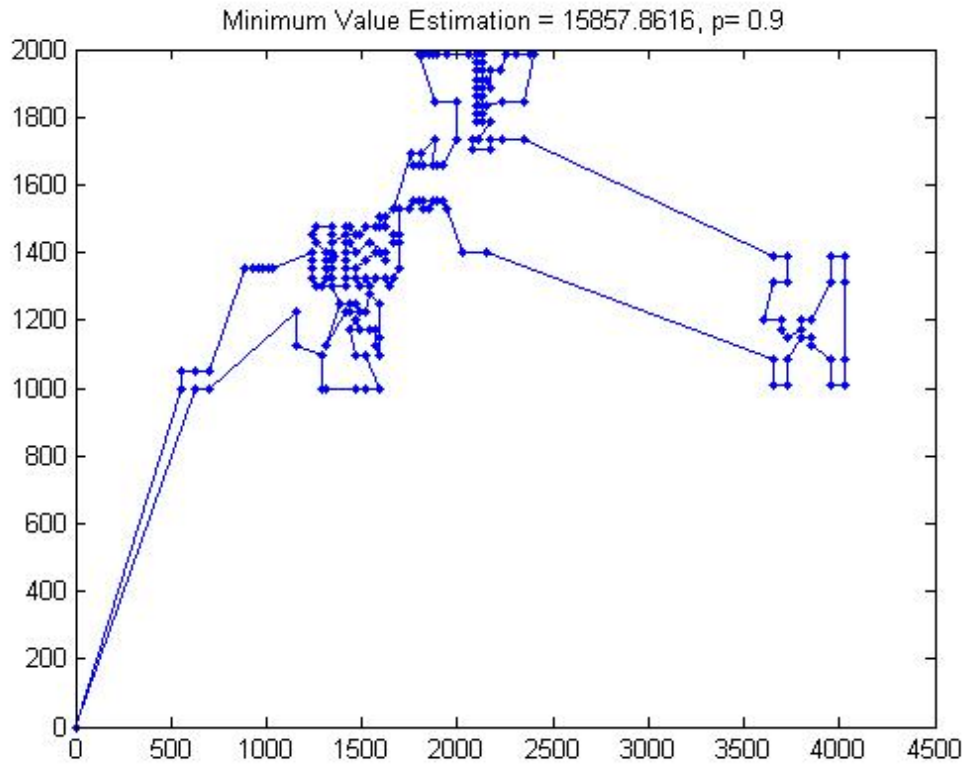
Πίνακας 3.14: Διαδρομή – d198, n=198, p=0.1



Διάγραμμα 3.41: d198 – n=198, p=0.5

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

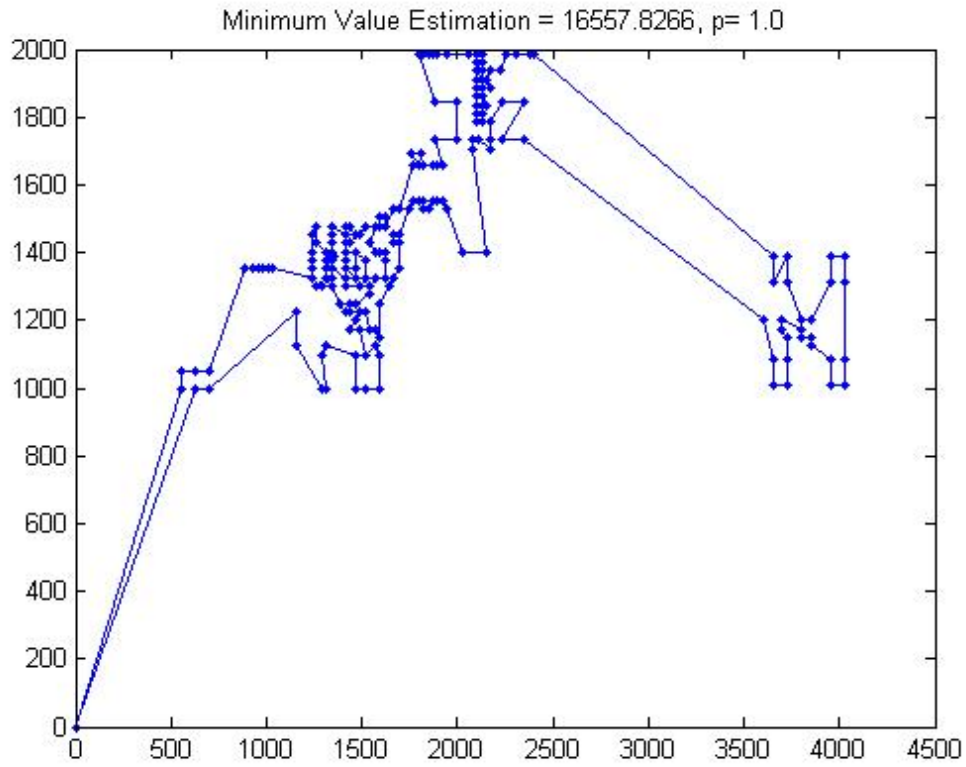
Πίνακας 3.15: Διαδρομή – d198, n=198, p=0.5



Διάγραμμα 3.42: d198 – n=198, p=0.9

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

Πίνακας 3.16: Διαδρομή – d198, n=198, p=0.9



Διάγραμμα 3.43: d198 – n=198, p=1.0

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

Πίνακας 3.17: Διαδρομή – d198, n=198, p=1.0

File-n	*p*	*Best*	*GA*	*iter*	*απόκλιση*
eil 51	0,1	**130,12**	**129,42**	424	**-0,54%**
	0,5	**310,75**	316,51	293	1,86%
	0,9	**407,92**	416,11	610	2,01%
	1	**426**	438,13	370	2,85%
eil101	0,1	**200,03**	**197,33**	1355	**-1,35%**
	0,5	**455,65**	464,48	1479	1,94%
	0,9	**601,5**	629,71	1199	4,69%
	1	**629**	667,60	1359	6,14%
kroA100	0,1	**9074,94**	9175,32	992	1,11%
	0,5	**16581,64**	17135,06	991	3,34%
	0,9	**20508,77**	22590,58	926	10,15%
	1	**21282**	22768,04	1486	6,98%
ch150	0,1	**2510,11**	2530,71	1324	0,82%
	0,5	**5016,85**	5264,08	2337	4,93%
	0,9	**6292,01**	6617,50	3664	5,17%
	1	**6528**	6946,53	3190	6,41%
d198	0,1	**7504,94**	7770,59	1987	3,54%
	0,5	**12527,56**	12965,89	2198	3,50%
	0,9	**15216,61**	15857,86	1585	4,21%
	1	**15780**	16557,83	1583	4,93%

Πίνακας 3.18: Αποτελέσματα TSP & PTSP με GA

Ο αλγόριθμος φαίνεται να λειτουργεί καλύτερα στα μικρά προβλήματα ενώ στα μεγάλα εμφανίζει καλύτερα αποτελέσματα όταν η πιθανότητα επίσκεψης του επόμενου κόμβου είναι μικρή. Στα προβλήματα μάλιστα eil51, eil101 εντοπίζει καλύτερη λύση από την έως τώρα βέλτιστη 129,42 και 197,34 αντίστοιχα. Ο αλγόριθμος καταφέρνει να βελτιώσει τη λύση κατά 0,54% από τη μέχρι στιγμής βέλτιστη στο πρόβλημα eil51, ενώ στο πρόβλημα kroA100 έχουμε βελτίωση κατά 1,35%.

Κεφάλαιο 4

Αλγόριθμος Διαφορικής Εξέλιξης

4.1 Εισαγωγή

Η Διαφορική Εξέλιξη (Differential Evolution (DE)) είναι ένας στοχαστικός, βασιζόμενος σε πληθυσμό, αλγόριθμος που προτάθηκε από τους Storm και Price [Kenneth V.Price, Rainer M.Storn, Jouni A.Lampinen (2005)]. Η Διαφορική Εξέλιξη έχει τα βασικά χαρακτηριστικά των εξελικτικών αλγορίθμων καθώς είναι ένας εξελικτικός αλγόριθμος. Έχει όμως και ένα αριθμό από διαφορές όπως το γεγονός ότι η μέθοδος αυτή εστιάζει στην απόσταση μεταξύ των μελών του πληθυσμού και στις διαφορετικές κατευθύνσεις που μπορεί να κινηθεί κάποιο μέλος του πληθυσμού. Για να εφαρμοστεί η διαφορική εξέλιξη θα πρέπει να έχουμε κωδικοποιήσει τις λύσεις με αναπαράσταση πραγματικού αριθμού ώστε να μπορούν να εφαρμοστούν οι τελεστές μετάλλαξης που θα περιγράψουμε στη συνέχεια. Στους εξελικτικούς αλγορίθμους όταν χρησιμοποιείται ο τελεστής διασταύρωσης εφαρμόζεται αρχικά πάνω σε δύο ή περισσότερους γονείς και στη συνέχεια στον απόγονο ή στους απογόνους που θα δημιουργηθούν εφαρμόζεται ένας τελεστής μετάλλαξης ο οποίος συνήθως μετακινεί τη λύση από ένα σημείο σε κάποιο άλλο με τη χρήση κάποιου βήματος που στηρίζεται σε κάποια κατανομή πιθανότητας. Υπάρχουν δύο βασικές διαφορές στη διαφορική εξέλιξη:

1. Ο τελεστής μετάλλαξης χρησιμοποιείται αρχικά για να παραχθεί ένα δοκιμαστικό διάνυσμα, το οποίο στην συνέχεια χρησιμοποιείται με κάποιον τελεστή διασταύρωσης για τη δημιουργία ενός απογόνου, και
2. Τα βήματα που γίνονται με τον τελεστή μετάλλαξης δεν υπόκεινται σε κάποια γνωστή κατανομή πιθανοτήτων αλλά επηρεάζονται από τις διαφορετικές τιμές στα γονίδια ανάμεσα στα μέλη του πληθυσμού.

4.2 Αναπαριστώντας τον πληθυσμό των ατόμων

Ο τρέχον πληθυσμός συμβολίζεται με P_x, και αποτελείται από τα διανύσματα $x_{i,g}$ τα οποία έχουν προκύψει είτε ως αρχικά σημεία είτε μετά από σύγκριση με άλλα διανύσματα:

$$P_{x,g} = (x_{i,g}), i = 0,1..., N-1, g = 0,1..., g_{max} \quad (4.1)$$

$$x_{i,g} = (x_{j,i,g}), j = 0,1...,D-1 \quad (4.2)$$

Ο δείκτης $g = 0,1,...g_{max}$, δηλώνει τη γενιά στην οποία ένα διάνυσμα ανήκει, ο δείκτης i δηλώνει το άτομο του πληθυσμού μεγέθους N και ο δείκτης j δηλώνει το στοιχείο του ατόμου μεγέθους D.

Ο πληθυσμός αρχικοποιείται τυχαία και στη συνέχεια ο DE μεταλλάσσει τυχαία επιλεγμένα άτομα ώστε να παραχθεί ένας ενδιάμεσος πληθυσμός P_v από N μεταλλαγμένα άτομα $v_{i,g}$

$$P_{v,g} = (v_{i,g}), i = 0,1...,N-1, g = 0,1...,g_{max} \quad (4.3)$$

$$v_{i,g} = (v_{j,i,g}), j = 0,1...,D-1 \quad (4.4)$$

Το κάθε άτομο της τρέχουσας γενιάς διασταυρώνεται με ένα μεταλλαγμένο άτομο ώστε να παραχθεί ο πληθυσμός P_u, με N άτομα, $u_{i,g}$

$$P_{u,g} = (u_{i,g}), i = 0,1...,N-1, g = 0,1...,g_{max} \quad (4.5)$$

$$u_{i,g} = (u_{j,i,g}), j = 0,1...,D-1 \quad (4.6)$$

4.2.1. Αρχικοποίηση

Πριν γίνει αρχικοποίηση του πληθυσμού ορίζονται το κάτω και άνω όριο, b_L και b_U αντίστοιχα. Στη συνέχεια μία γεννήτρια τυχαίων αριθμών καθορίζει κάθε παράμετρο για κάθε διάνυσμα από ένα ορισμένο διάστημα.

$$x_{i,o} = randperm(n) \quad (4.7)$$

Η συνάρτηση randperm(n) επιστρέφει σε τυχαία σειρά όλους τους ακέραιους από το 1 μέχρι το n. Στη συνέχεια εντοπίζεται ο κόμβος με την τιμή 1 (αποθήκη) και μεταφέρεται στην αρχή του διανύσματος. Η μεταβλητή i δηλώνει το άτομο του πληθυσμού.

4.2.2. Μετάλλαξη

Κατά την μετάλλαξη ο DE συνδυάζει τρία άτομα της τρέχουσας γενιάς ώστε να προκύψει το μεταλλαγμένο άτομο $v_{i,g}$:

$$v_{i,g} = x_{r0,g} + F\left(x_{r1,g} - x_{r2,g}\right) \tag{4.8}$$

Η μεταβλητή $F \in (0,1+)$ είναι ένας θετικός πραγματικός αριθμός και τα διανύσματα $x_{r0,g}$, $x_{r1,g}$, $x_{r2,g}$ είναι τυχαία επιλεγμένα άτομα του πληθυσμού.

4.2.3. Διασταύρωση

Η διασταύρωση πραγματοποιείται μεταξύ ενός ατόμου του τρέχοντος πληθυσμού και ενός μεταλλαγμένου ατόμου για κάθε άτομο του τρέχοντος πληθυσμού βάσει της πιθανότητας διασταύρωσης.

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & if\left(rand_j(0,1)\right) \le p_C \\ x_{j,i,g} & else \end{cases} \tag{4.9}$$

Η πιθανότητα διασταύρωσης $Cr \in [0,1]$, ελέγχει τα στοιχεία του μεταλλαγμένου ατόμου που θα αντιγραφούν στο νέο άτομο. Αν ο τυχαίος αριθμός είναι μικρότερος από την πιθανότητα μετάλλαξης τότε το στοιχείο j του i ατόμου θα κληρονομηθεί στο άτομο της επόμενης γενιάς διαφορετικά θα αντιγραφεί το στοιχείο του ατόμου $x_{j,i,g}$.

4.2.4. Επιλογή

Αν η τιμή της αντικειμενικής συνάρτησης του άτομο $u_{i,g}$ είναι ίση ή μικρότερη από την τιμή του ατόμου $x_{i,g}$, τότε αντικαθιστά το άτομο $x_{i,g}$ στην επόμενη γενιά διαφορετικά το άτομο $x_{i,g}$ μεταβιβάζεται ως έχει στην επόμενη γενιά.

$$u_{i,g+1} = \begin{cases} u_{i,g} & if \quad f(u_{i,g}) \le f(x_{i,g}) \\ x_{i,g} & else \end{cases} \tag{4.10}$$

Μόλις η νέα γενιά επιλεγεί, η διαδικασία της μετάλλαξης, διασταύρωσης και επιλογής επαναλαμβάνονται μέχρι να εντοπιστεί το βέλτιστο ή κάποιο κριτήριο τερματισμού να ικανοποιείται πχ ο αριθμός των γενεών που έχουν παραχθεί έχει φτάσει το μέγιστο που αρχικά έχει οριστεί g_{max} .

4.3 Σχεδιασμός και παράμετροι του Differential Evolution

Ο αλγόριθμος που χρησιμοποιήθηκε για να επιλυθεί αυτό το πρόβλημα κωδικοποιήθηκε στην Matlab χρησιμοποιώντας δεκαδική κωδικοποίηση. Το χρωμόσωμα είναι ένα διάνυσμα *n* διαστάσεων [*1,2..,n*], το οποίο αναπαριστά μία πιθανή λύση του προβλήματος μας (τη σειρά με την οποία εξυπηρετήθηκε ο κάθε πελάτης) και το κάθε στοιχείο του διανύσματος αντιπροσωπεύει έναν κόμβο-πελάτη.
Τα στοιχεία που χρησιμοποιούνται είναι όμοια με του γενετικού αλγορίθμου ώστε να είναι εφικτή και η σύγκριση τους,
Ο αρχικός πληθυσμός δίνεται τυχαία. Κάθε πιθανή διαδρομή ξεκινάει από τον κόμβο 1 (αποθήκη) και η σειρά επίσκεψη των πελατών είναι τυχαία. Ο πληθυσμός κάθε γενιάς αποτελείται από 40 άτομα. Την πιθανότητα διασταύρωση (p_c) την έχουμε ορίσει 0.8. Η πιθανότητα μετάλλαξης (p_m) είναι στην πραγματικότητα μονάδα καθώς κάθε άτομο μεταλλάσσεται και ουσιαστικά προκύπτει από τρία τυχαία επιλεγμένα άτομα με σκοπό να αντικατασταθεί από ένα καλύτερο. Η παράμετρος F στην διαδικασίας μετάλλαξης έχει οριστεί ίση με 5.
Επίσης χρησιμοποιούμε τον αλγόριθμο 3-opt για τοπική αναζήτηση και πραγματοποιείται για 50 επαναλήψεις σε κάθε άτομο. Αν βρεθεί καλύτερη λύση πριν τελειώσει ο αριθμός των επαναλήψεων η τοπική αναζήτηση διακόπτεται και προχωράμε στο επόμενο άτομο.
Χρησιμοποιείται ελιτισμός και το καλύτερο άτομο της γενιάς μεταφέρεται κάθε φορά στην επόμενη χωρίς να μπει στη διαδικασία της διασταύρωσης και της μετάλλαξης εξασφαλίζοντας έτσι ότι η καλύτερη λύση θα είναι πάντα διαθέσιμη. Στο τέλος κάθε γενιάς σε περίπτωση που προκύψουν περισσότερα από ένα άτομα τα οποία θα έχουν την ίδια λύση με τη βέλτιστη χρησιμοποιείται η συνάρτηση 3-opt ώστε να

τροποποιηθούν οι παρούσες λύσεις και να διαφοροποιηθούν από τη βέλτιστη ώστε να διερευνηθεί το πεδίο των λύσεων και να προκύψουν νέες λύσεις στην επόμενη γενιά. Χρησιμοποιούμε *fitness-proportionate selection* για την επιλογή των ατόμων που θα συμμετέχουν στην διαδικασία της διασταύρωσης και της μετάλλαξης. Επίσης μετά τη διαδικασία της διασταύρωσης χρησιμοποιείται μία συνάρτηση επιδιόρθωσης καθώς είναι πολύ πιθανό η λύση που θα προκύψει να περιλαμβάνει ορισμένους κόμβους δύο φορές ενώ κάποιοι να απουσιάζουν τελείως από την πιθανή λύση.

Για τερματικό κριτήριο επιτρέπουμε όχι παραπάνω από ένα καθορισμένο μέγιστο αριθμό επαναλήψεων (M_1). Επειδή εξετάζονται διαφορετικού μεγέθους προβλήματα ο μέγιστος αριθμός επαναλήψεων (M_1) δεν είναι σταθερός, σε μεγαλύτερα προβλήματα επιτρέπουμε περισσότερες επαναλήψεις. Παρόλα αυτά προσθέτουμε επίσης και ένα άλλο τερματικό κριτήριο που μπορεί να σταματήσει τον αλγόριθμο πριν ο μέγιστος αριθμός επαναλήψεων επιτευχθεί. Ειδικότερα τερματίζουμε το πρόγραμμα αν το καλύτερο άτομο στο πληθυσμό δεν έχει αλλάξει περισσότερο από $\epsilon = 0.01$ στις τελευταίες 300 γενιές (M_2=300).

4.3.1. Ψευδοκώδικας του Differential Evolution

Αλγόριθμος 4.1: Differential Evolution

Βήμα 1. Προσδιόρισε τις παραμέτρους του DE

Βήμα 2. Αρχικοποίηση του πληθυσμού P(0)

Βήμα 3. Για k=1 μέχρι N_{ga}

a) Υπολόγισε τη συνάρτηση προσαρμογής για κάθε άτομο

b) **Επιλογή**: Χρησιμοποιώντας τη *fitness-proportionate selection* επέλεξε από τον πληθυσμό τα άτομα που θα μπουν στο χώρο του ζευγαρώματος (M(k)).

c) **Μετάλλαξη:** Επέλεξε τρία άτομα για να γίνει η μετάλλαξη.

a) **Διασταύρωση:** Με πιθανότητα p_c διασταύρωσε το τρέχον άτομο με ένα τυχαίο από το 'χώρο ζευγαρώματος'

b) **Επιδιόρθωση:** Γίνεται επιδιόρθωση της πιθανής λύσης

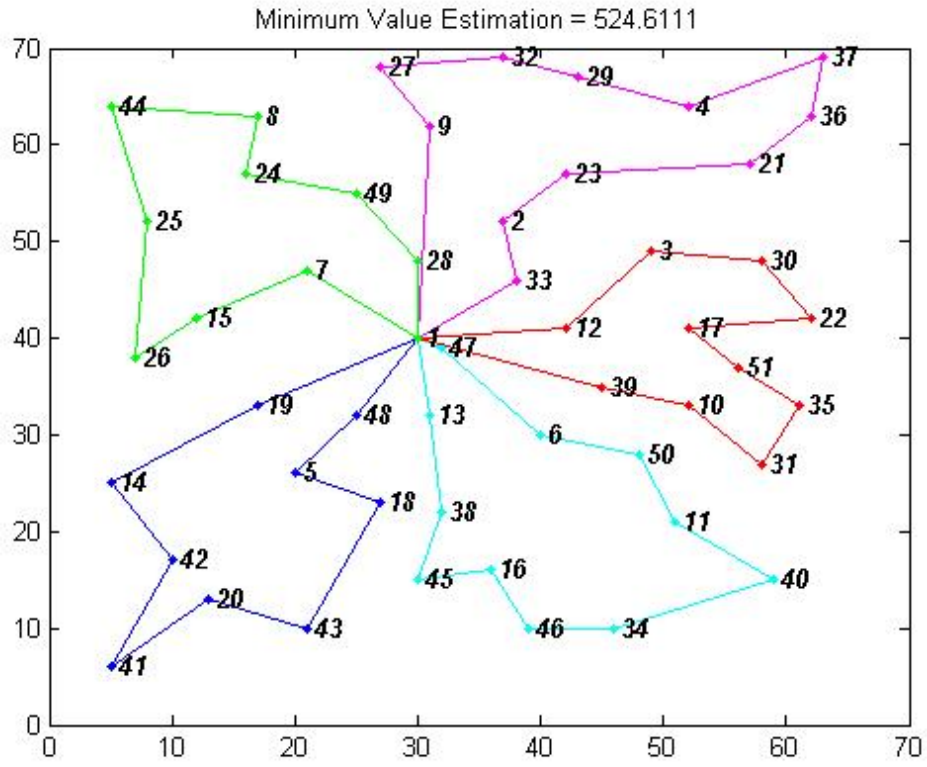
d) **Τοπική Αναζήτηση:** Για κάθε άτομο που πέρασε στην επόμενη γενιά χρησιμοποίηση την μέθοδο 3-opt για τη εύρεση καλύτερης λύσης.

Βήμα 4. Αποτελέσματα

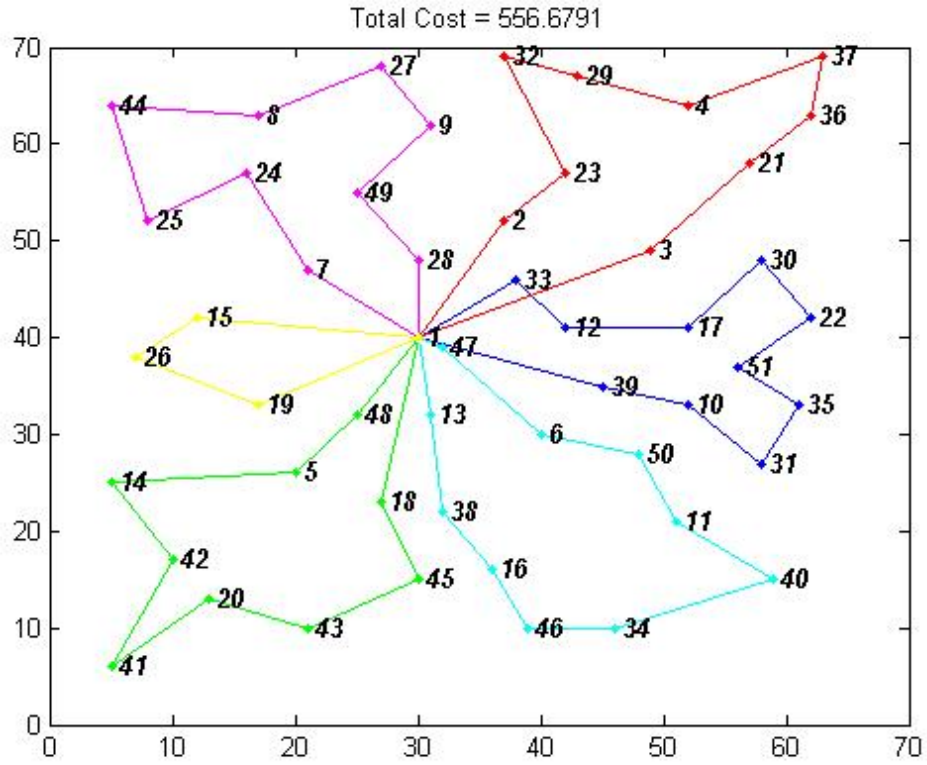
4.4 Επίλυση του VRP με χρήση Differential Evolution

Σε αυτήν την ενότητα επιλύεται το πρόβλημα δρομολόγησης οχημάτων με χρήση του αλγόριθμου *αλγόριθμος 4.1* και τοπική αναζήτηση 3-opt όπως παρουσιάστηκε στο Κεφάλαιο 3. Η αντικειμενική συνάρτηση προκύπτει από την Εξ. (2.3) υπό τους περιορισμούς των Εξ. (2.4), (2.5), (2.6) και τα πακέτα δεδομένων είναι ίδια με την Παράγραφο 3.5.

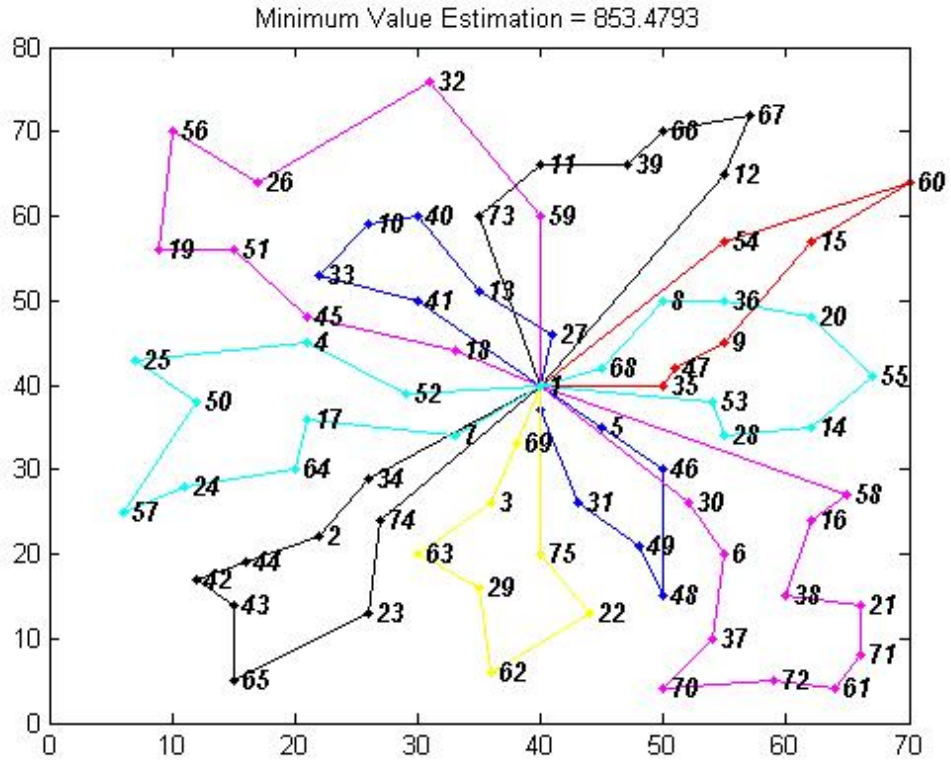
4.4.1. Αποτελέσματα VRP με χρήση DE



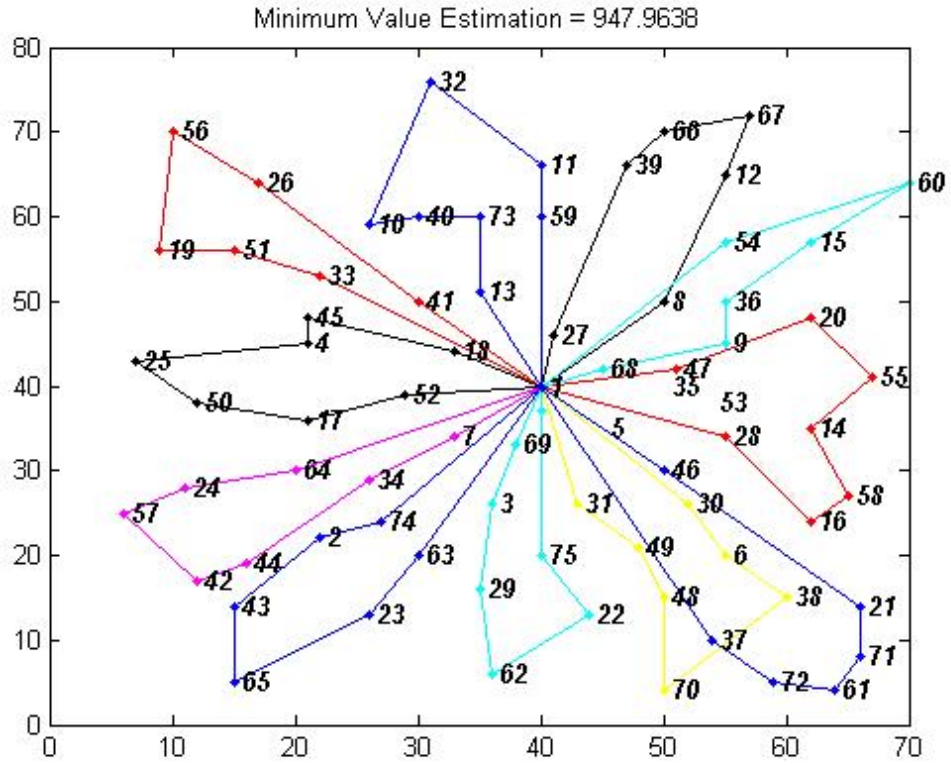
Διάγραμμα 4.1: par_1: n=51, Q=160, max_tour_length=999999, service_time=0



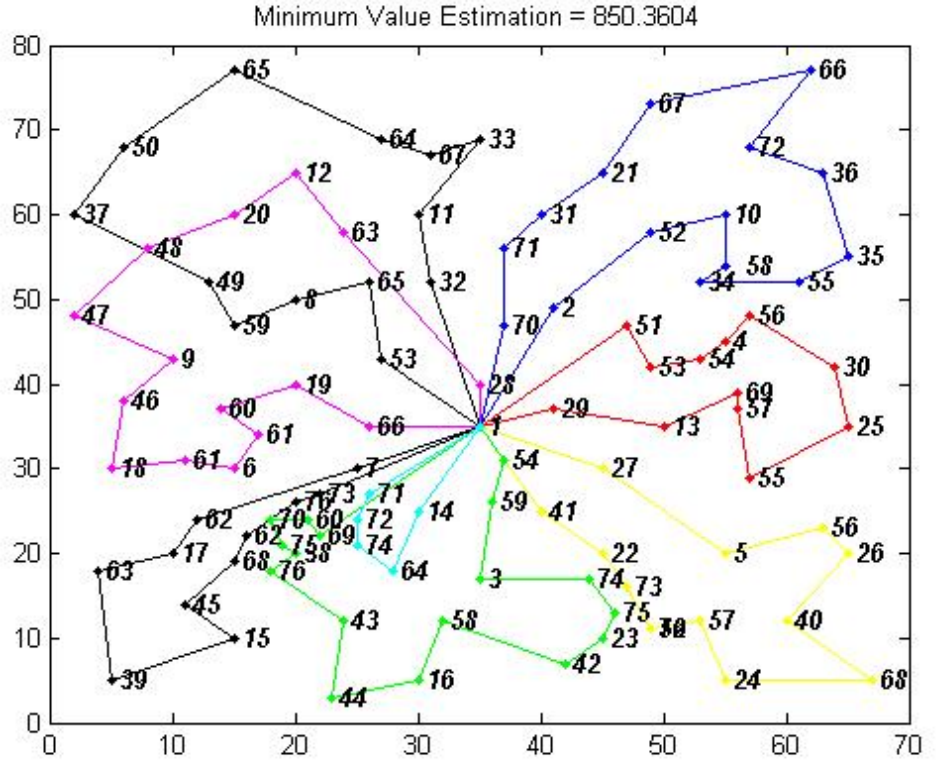
Διάγραμμα 4.2: par_6: n=51, Q=160, max_tour_length=99999, service_time=10



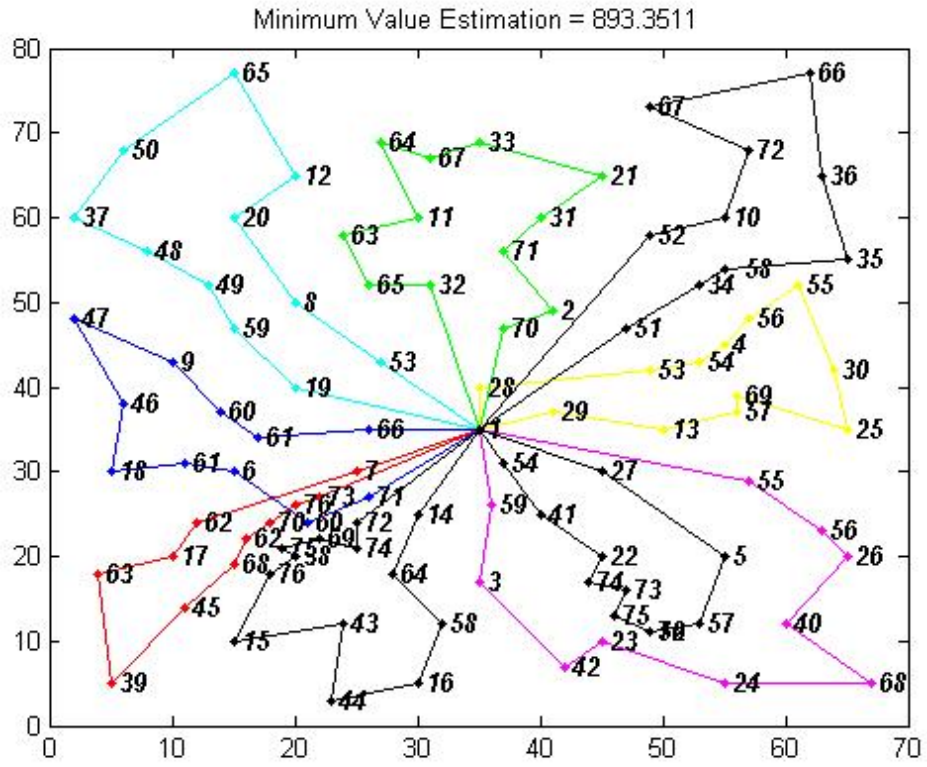
Διάγραμμα 4.3: par_2: n=76, Q=140, max_tour_length=99999, service_time=0,



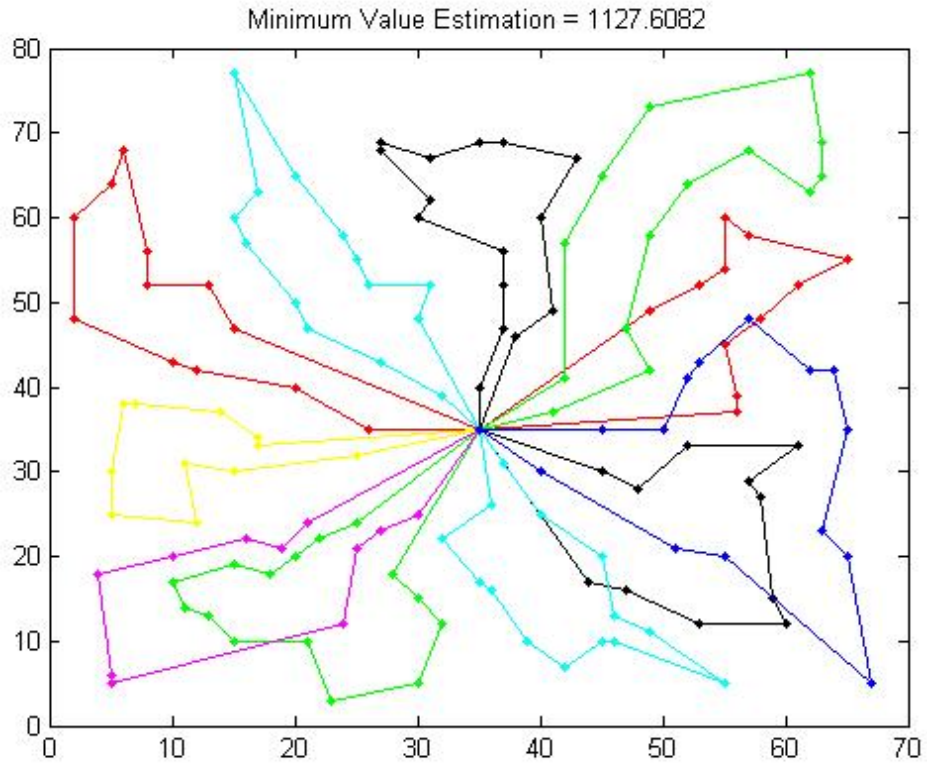
Διάγραμμα 4.4: par_7: n=76, Q=140, max_tour_length=160, service_time=10



Διάγραμμα 4.5: par_3: n=101, Q=200, max_tour_length=99999, service_time=0



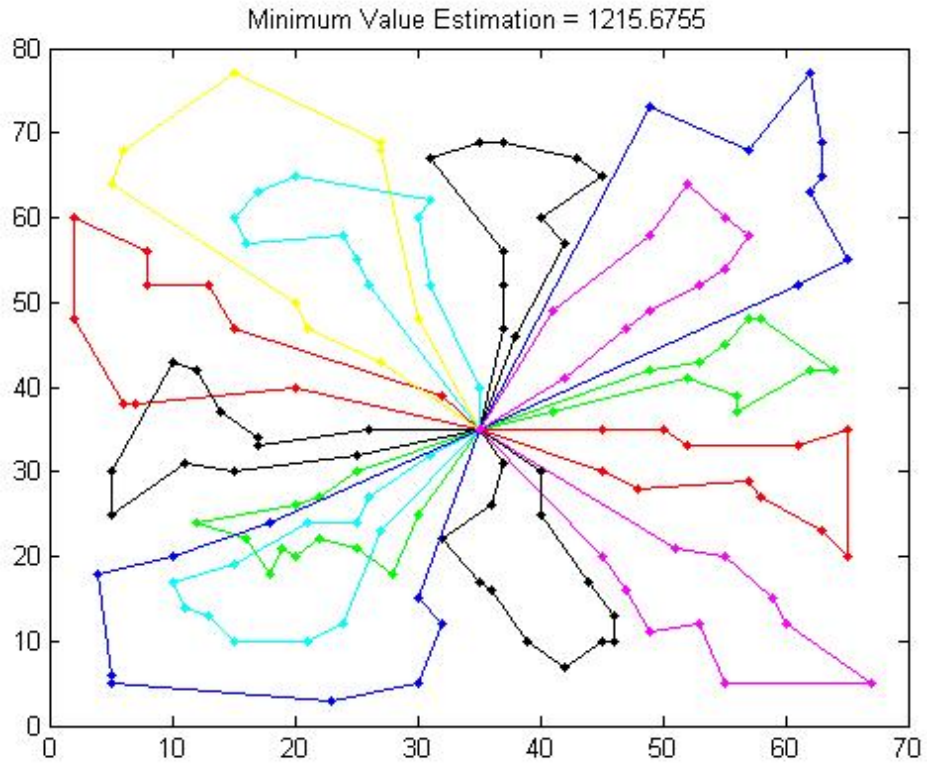
Διάγραμμα 4.6: par_8: n=101, Q=200, max_tour_length=230, service_time=10



Διάγραμμα 4.7: par_4: n=151, Q=200, max_tour_length=99999, service_time=0

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

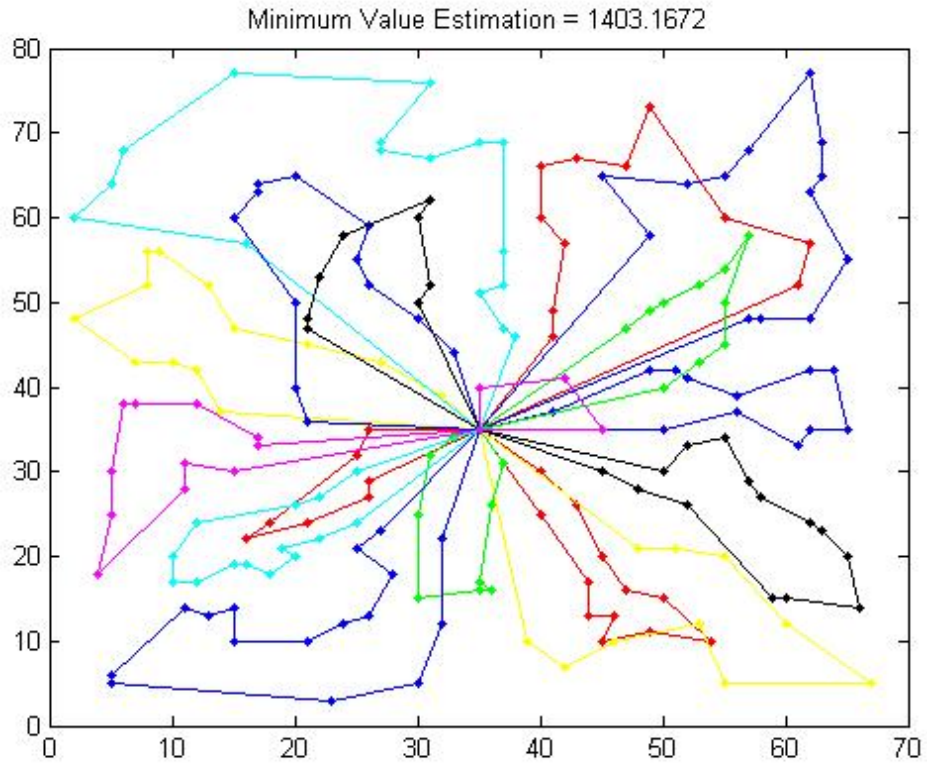
Πίνακας 4.1: Διαδρομή - **par_4**: n=151



Διάγραμμα 4.8: par_9: n=151, Q=200, max_tour_length=200, service_time=10

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

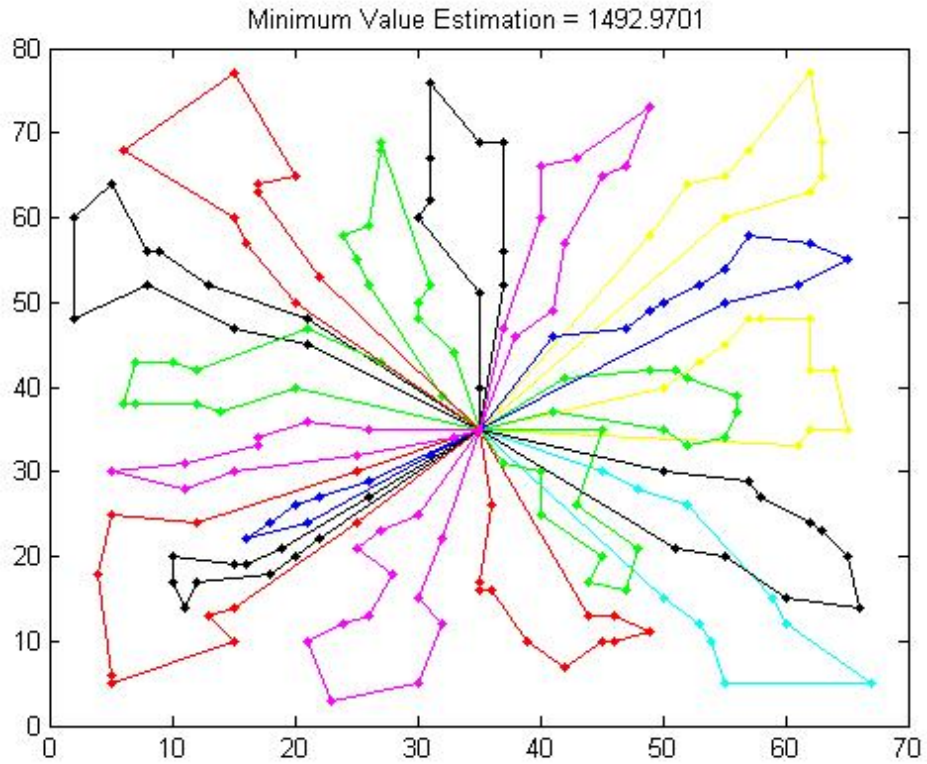
Πίνακας 4.2: Διαδρομή - par_9: n=151



Διάγραμμα 4.9: par_5: n=201, Q=200, max_tour_length=99999, service_time=0

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

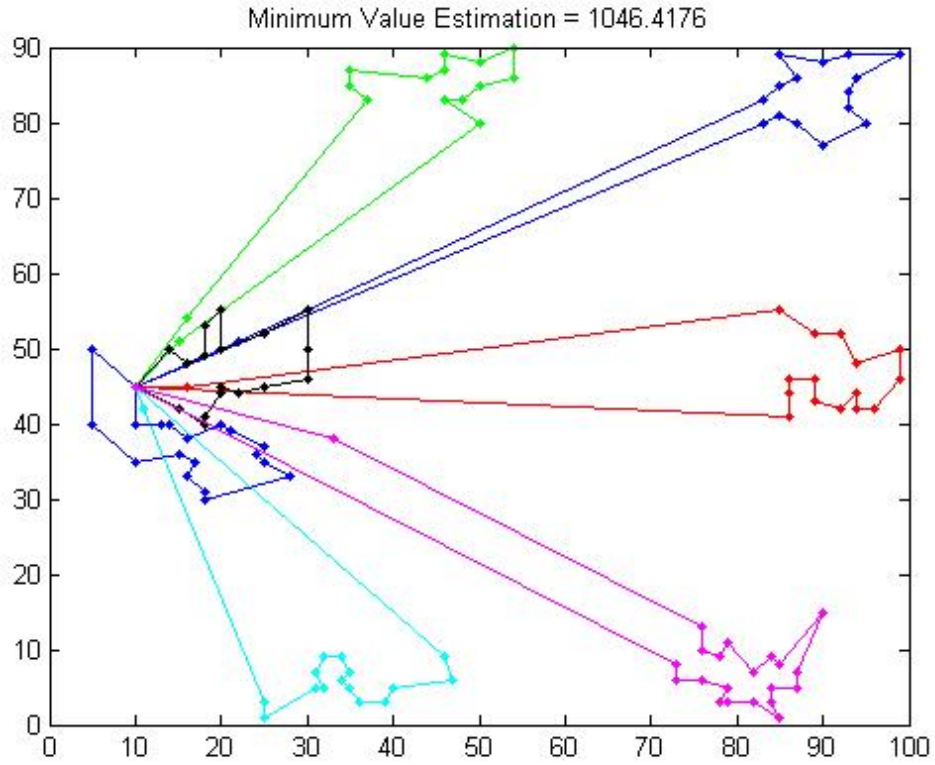
Πίνακας 4.3: Διαδρομή - par_5: n=201



Διάγραμμα 4.10: par_10: n=201, Q=200, max_tour_length=200, service_time=10

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

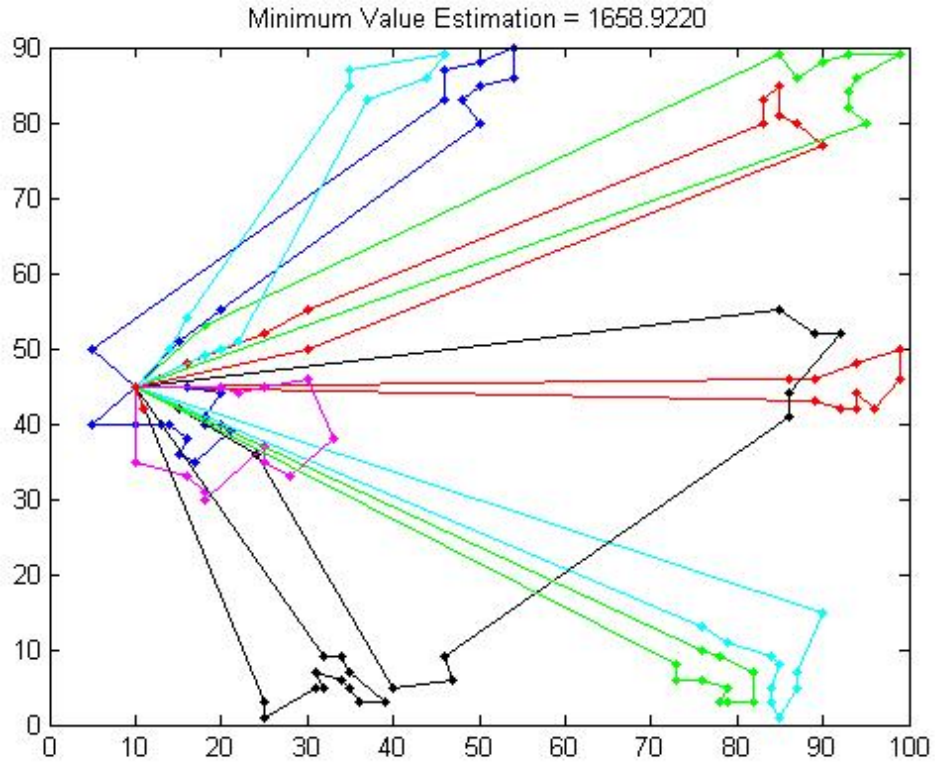
Πίνακας 4.4: Διαδρομή - par_10: n=201



Διάγραμμα 4.11: par_11: n=121, Q=200, max_tour_length=99999, service_time=0

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

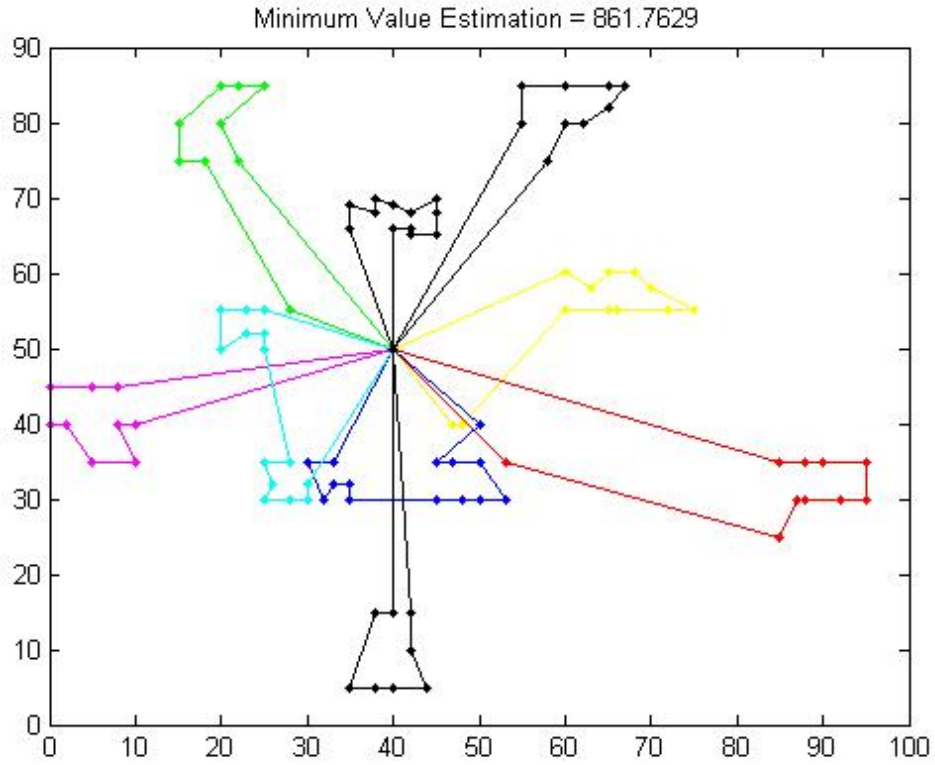
Πίνακας 4.5: Διαδρομή - par_11: n=121



Διάγραμμα 4.12: par_13: n=121, Q=200, max_tour_length=720, service_time=50

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

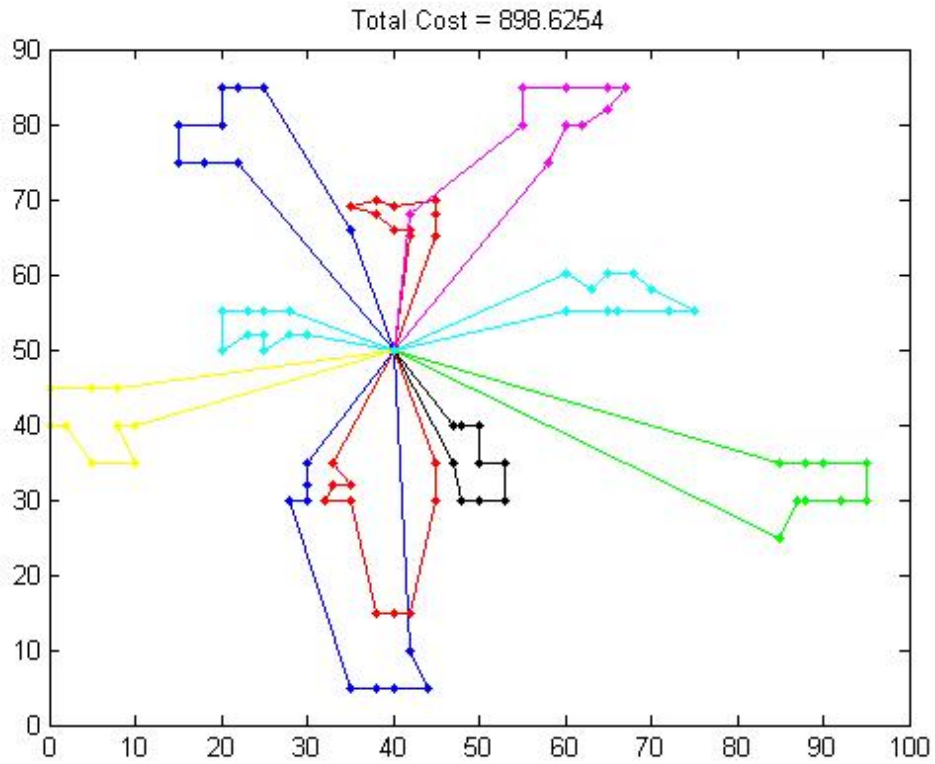
Πίνακας 4.6: Διαδρομή - par_13: n=121



Διάγραμμα 4.13: par_12: n=101, Q=200, max_tour_length=99999, service_time=0

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

Πίνακας 4.7: Διαδρομή - par_12: n=101



Διάγραμμα 4.14: par_14: n=101, Q=200, max_tour_length=1040, service_time=90

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

Πίνακας 4.8: Διαδρομή - par_14: n=101

File-n	*Best Cost*	*DE*	*iter*	*απόκλιση*
par1-50	**524,61**	**524,61**	469	**0,00%**
par2-75	**835,26**	853,47	1519	**2,18%**
par3-100	**826,14**	850,36	5857	**2,93%**
par4-150	**1028,42**	1127,60	9497	**9,64%**
par5-199	**1291,29**	1403,20	29425	**8,67%**
par6-50	**555,43**	556,67	987	**0,22%**
par7-75	**909,68**	947,96	3340	**4,21%**
par8-100	**865,94**	893,35	2778	**3,17%**
par9-150	**1162,55**	1215,70	20316	**4,57%**
par10-199	**1395,55**	1492,97	35417	**6,98%**
par11-120	**1042,11**	1046,40	2797	**0,41%**
par12-100	**819,56**	861,76	1861	**5,15%**
par13-120	**1541,19**	1658,90	3706	**7,64%**
par14-100	**866,37**	898,62	3785	**3,72%**

Πίνακας 4.9: Αποτελέσματα με Differential Evolution

Ο αλγόριθμος εντοπίζει καλύτερες λύσεις στα προβλήματα όπου δεν ισχύει ο περιορισμός της μέγιστης διαδρομής (par_1, par_2, par_3, par_11) σε αντίθεση με τα προβλήματα που ισχύει (par_6, par_7, par_8, par_13). Στα μεγάλα προβλήματα όμως (par_4-par_9, par_5-par_10) συμβαίνει το αντίθετο.

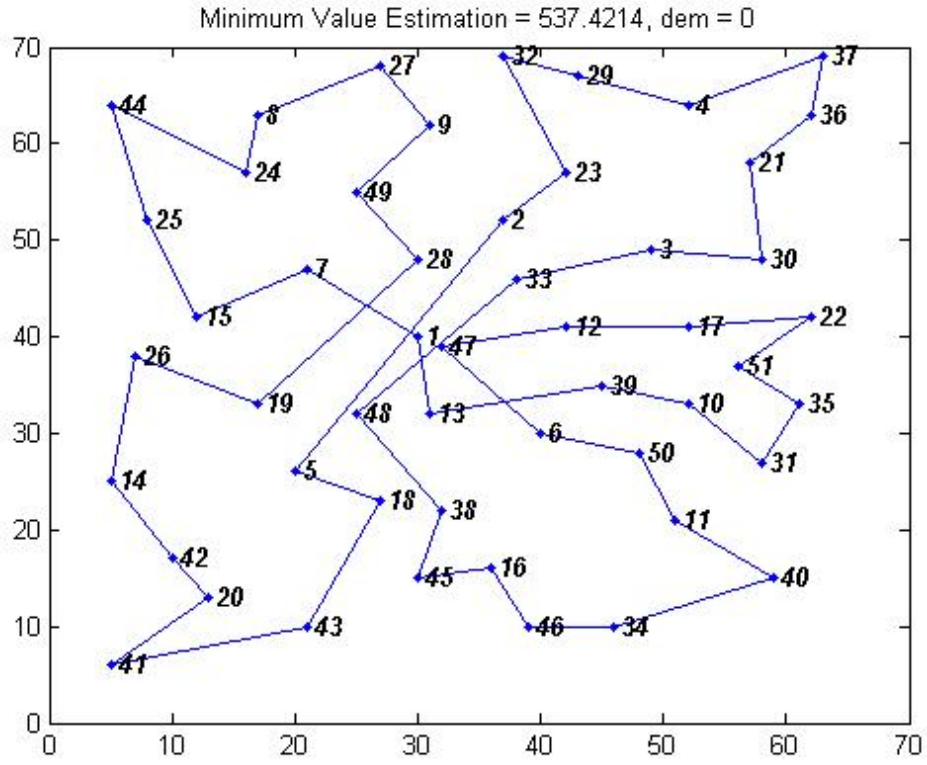
Εξετάζοντας τα γραφήματα πιο προσεχτικά, παρατηρείται ότι στα προβλήματα όπου η τιμή της λύσης που βρίσκει ο αλγόριθμος είναι πολύ κοντά στο βέλτιστη, οι διαδρομές δεν διασταυρώνονται. Ειδικότερα στα προβλήματα όπου ο αριθμός των πελατών είναι μεγάλος το όχημα επιλέγει να εξυπηρετήσει σε διαφορετικές διαδρομές τους πελάτες που βρίσκονται πιο μακριά από την αποθήκη από τους πελάτες που βρίσκονται σε πιο κοντινή απόσταση (par_11). Επίσης εκτός από την απόσταση επιλέγεται η κατεύθυνση όπου βρίσκεται ο κάθε πελάτης, δηλαδή οι διαδρομές είναι ακτινικές ως προς την αποθήκη και επιλέγονται οι πιο απομακρυσμένοι πελάτες να εξυπηρετηθούν στην ίδια διαδρομή εφόσον βρίσκονται στην ίδια κατεύθυνση.

4.5 Επίλυση του SVRP με χρήση Differential Evolution

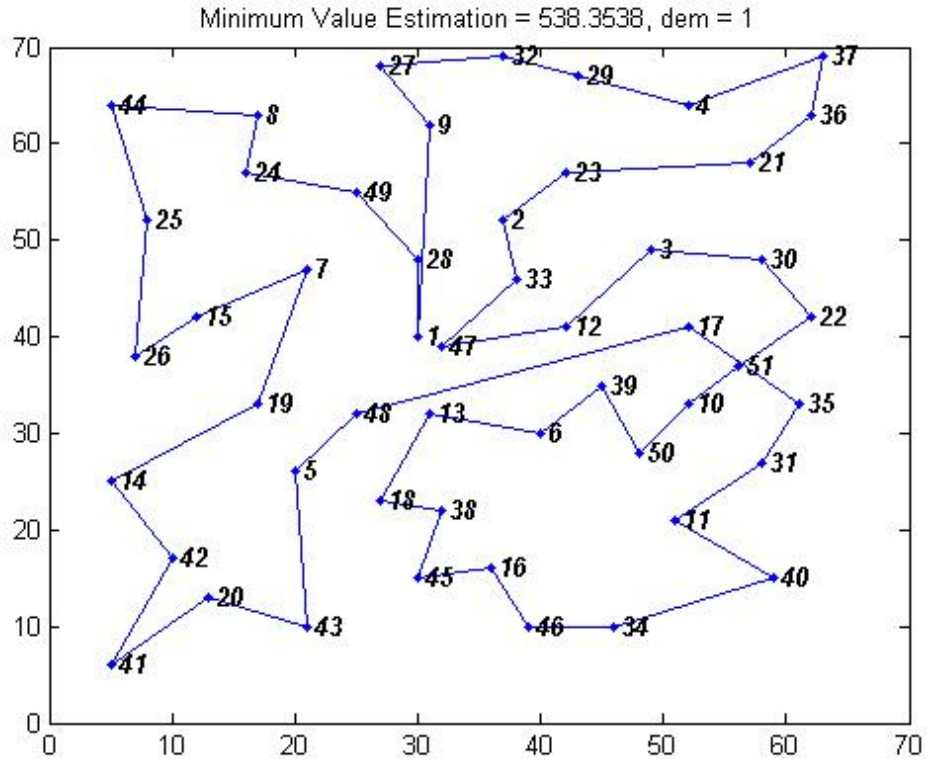
Σε αυτήν την ενότητα επιλύεται το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση με χρήση του *αλγόριθμου 4.1* και τοπική αναζήτηση 3-opt. Η αντικειμενική συνάρτηση προκύπτει από τις Εξ. (2.10), (2.11), (2.12) τροποποιημένη ώστε να υπολογίζει το κόστος για συγκεκριμένη χωρητικότητα του οχήματος. Επίσης η αντικειμενική συνάρτηση δεν υπολογίζει το κόστος για όλες τις πιθανές τιμές της ζήτησης (0<d<ζήτηση πελάτη), αλλά για συγκεκριμένες τιμές που ορίζουμε.

Τα πακέτα δεδομένων καθώς και η αντικειμενική συνάρτηση είναι ίδια με της Παραγράφου 3.6 που χρησιμοποιήθηκαν και στο γενετικό αλγόριθμο.

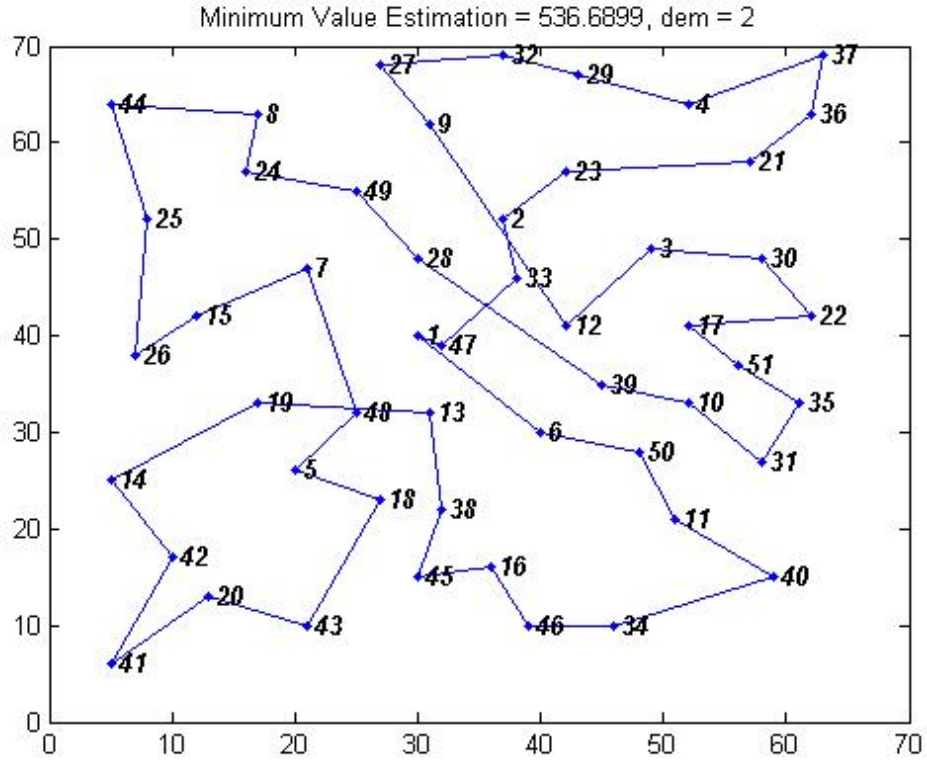
4.5.1. Αποτελέσματα SVRP με χρήση DE



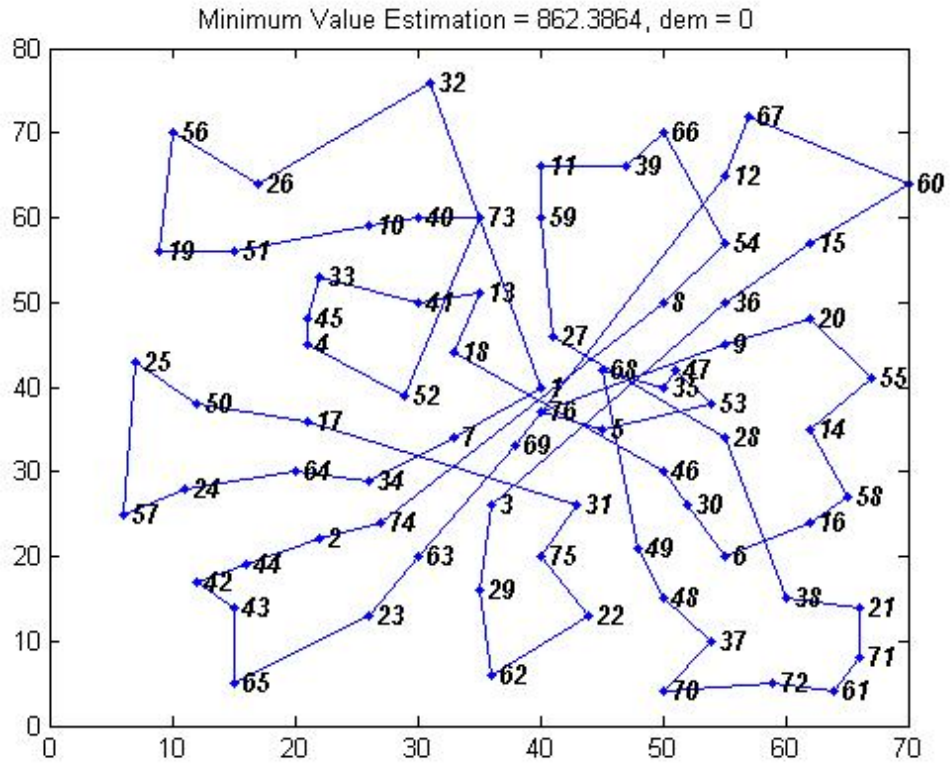
Διάγραμμα 4.15: par1**:** n=51, dem=0, Q=160



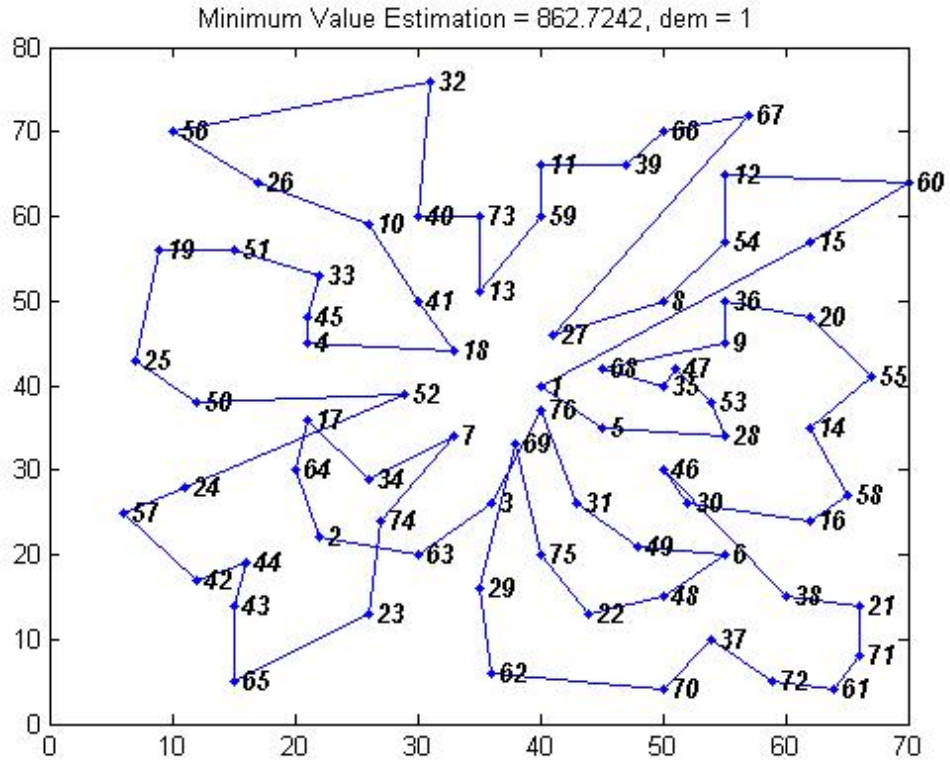
Διάγραμμα 4.16: par1: n=51, dem=1, Q=160



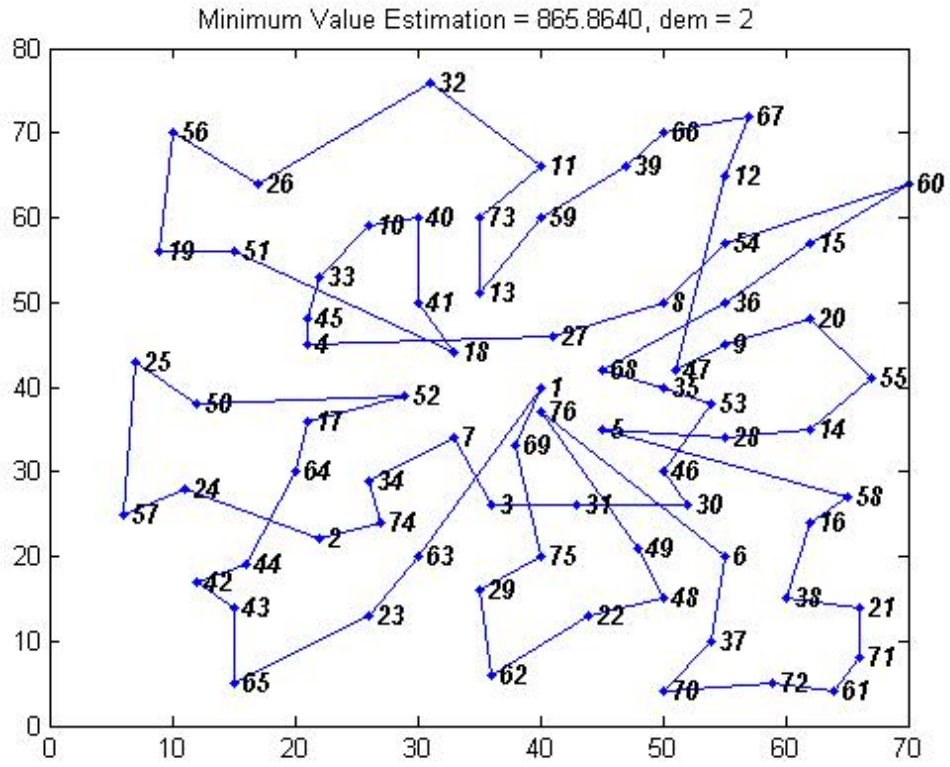
Διάγραμμα 4.17: par1: n=51, dem=2, Q=160



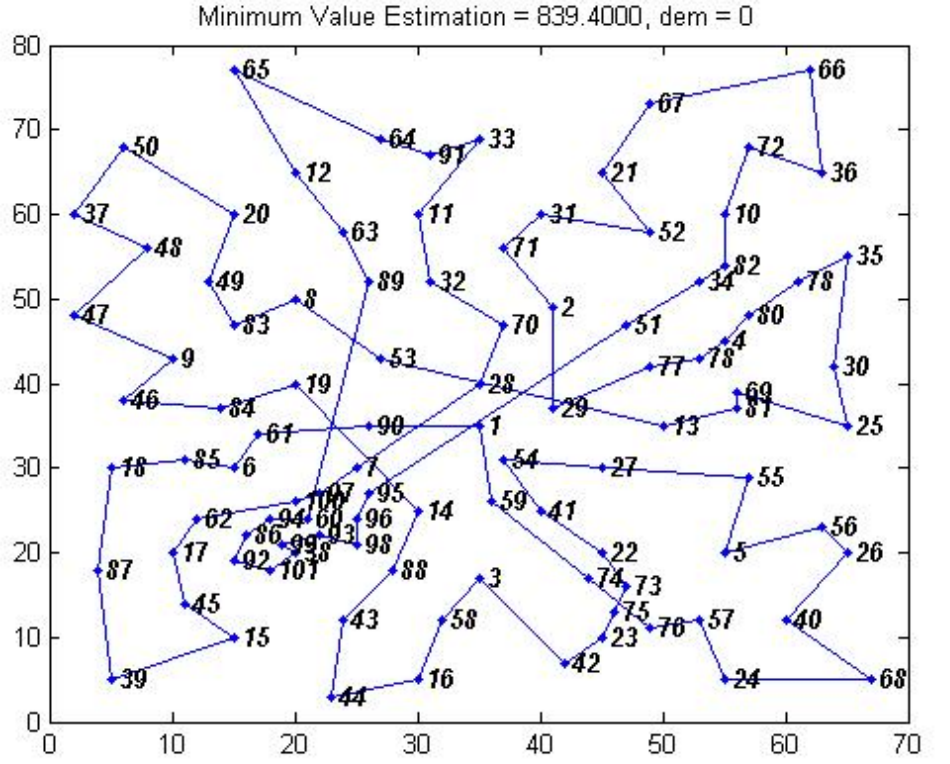
Διάγραμμα 4.18: par2: n=76, dem=0, Q=140



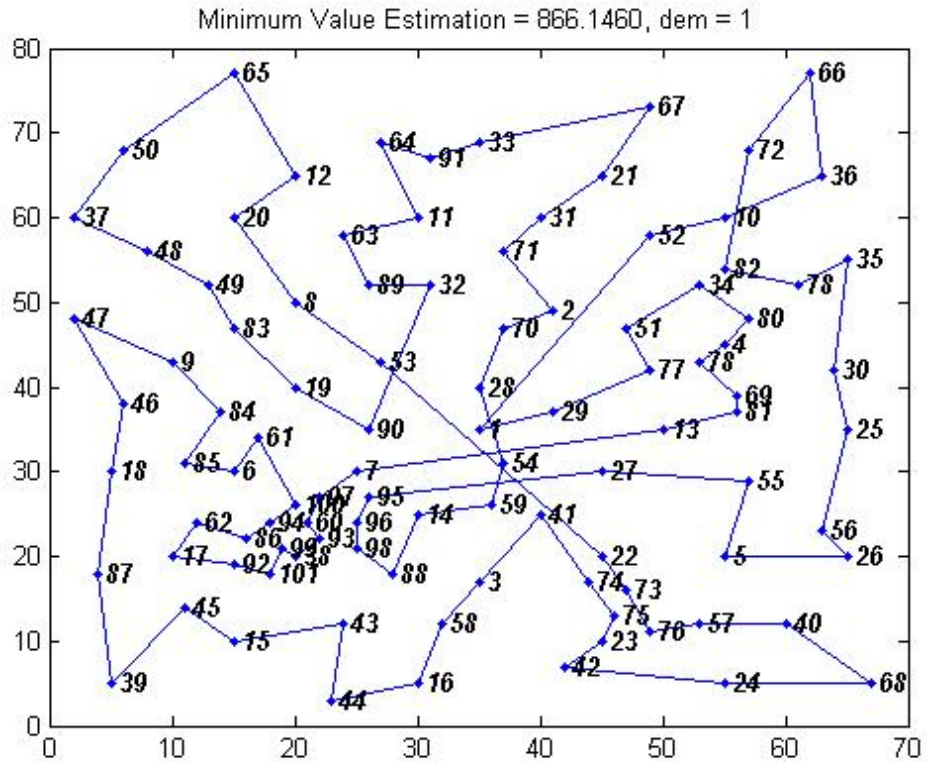
Διάγραμμα 4.19: par2: n=76, dem=1, Q=140



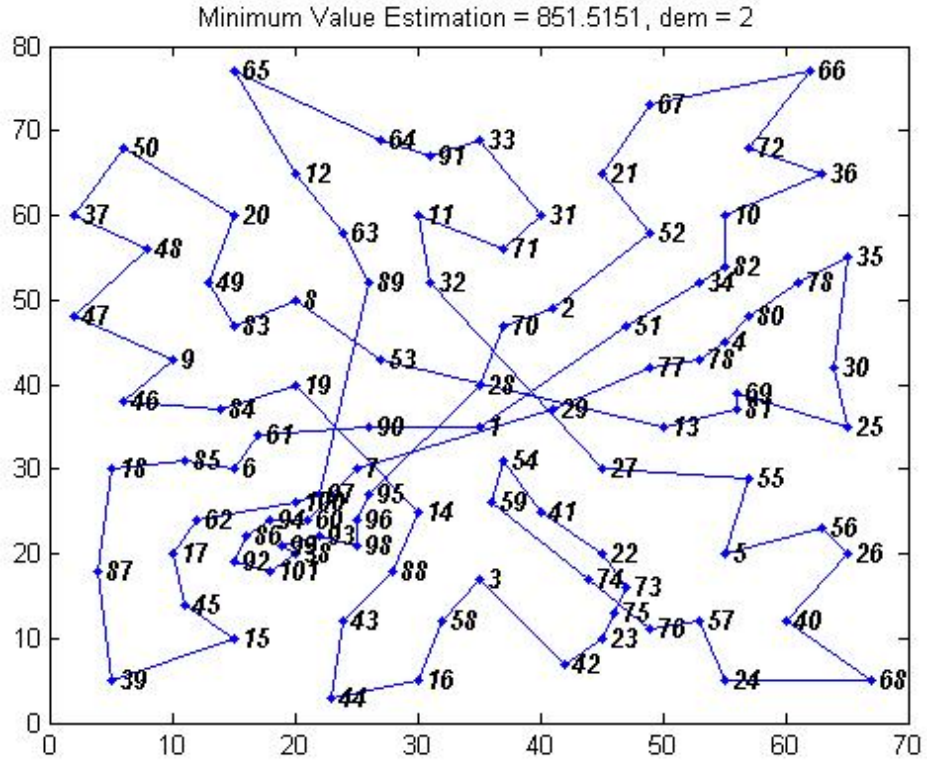
Διάγραμμα 4.20: par2: n=76, dem=2, Q=140



Διάγραμμα 4.21: par3: n=101, dem=0, Q=200



Διάγραμμα 4.22: par3: n=101, dem=1, Q=200



Διάγραμμα 4.23: par3: n=101, dem=2, Q=200

File-n	*dem*	*Best Cost*	*DE*	*iter*	*απόκλιση*
par_1	0	**524,61**	537,4214	967	**2,44%**
	1		538,3538	944	**2,62%**
	2		536,6899	649	**2,30%**
par_2	0	**835,26**	862,3864	978	**3,25%**
	1		862,7242	1007	**3,29%**
	2		865,864	719	**3,66%**
par_3	0	**826,14**	839,4	2057	**1,61%**
	1		866,146	1542	**4,84%**
	2		851,5151	2083	**3,07%**

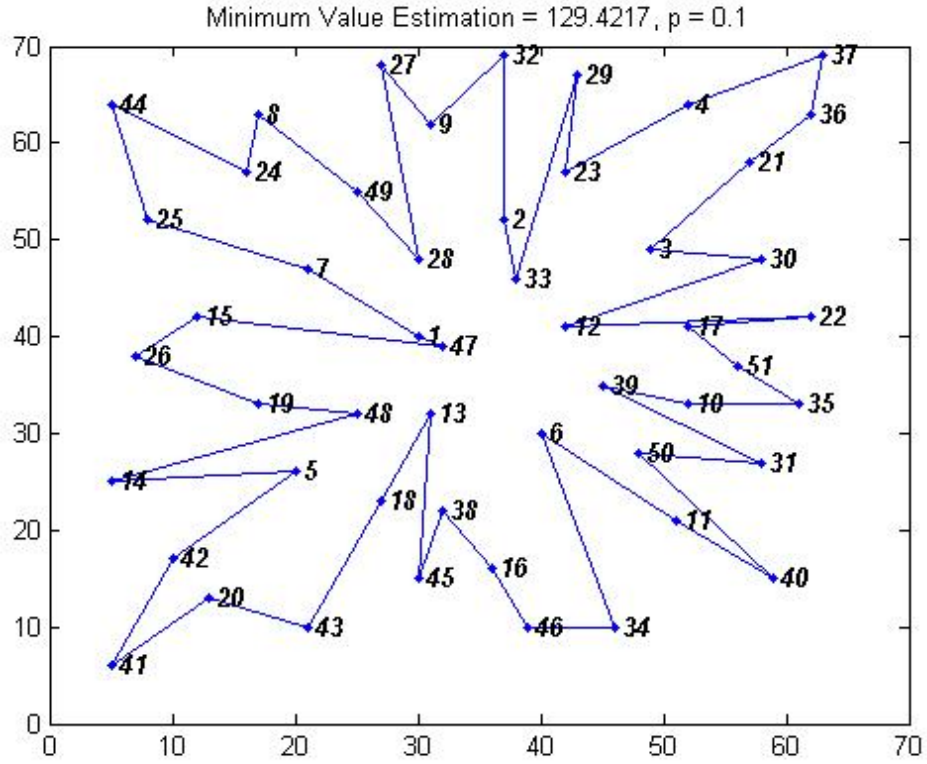
Πίνακας 4.10: Αποτελέσματα SVRP με DE

Να υπενθυμίσουμε ότι όταν η μεταβλητή dem είναι 0 ουσιαστικά επιλύουμε το απλό πρόβλημα δρομολόγησης οχημάτων. Στα προβλήματα par_1 και par_2 παρατηρούμε ότι σε όλες τις περιπτώσεις η απόκλιση της λύσης κυμαίνεται στο 2,5% και 3,3% αντίστοιχα. Ενώ στο πρόβλημα par_3 έχουμε μεγαλύτερες διακυμάνσεις. Αυτό που αξίζει να σημειωθεί είναι ότι στο par_3 η λύση που βρίσκει ο αλγόριθμος για dem=0 , κόστος=839,4 και απόκλιση 1,61%, είναι καλύτερη από τη λύση που βρήκε στο απλό πρόβλημα δρομολόγησης οχημάτων, κόστος=850,36 με απόκλιση 2,93% (Διάγραμμα 4.5). Επίσης η λύση εντοπίζεται και γρηγορότερα σε αυτήν την περίπτωση (λιγότερες γενιές). Αυτό πολύ πιθανό να οφείλεται στην αρχικοποίηση του πληθυσμού καθώς όλες οι παράμετροι των αλγορίθμων που χρησιμοποιούνται οι ίδιες με μοναδική διαφορά τον τρόπος εύρεσης του κόστους της διαδρομής, δηλαδή της αντικειμενικής συνάρτησης.

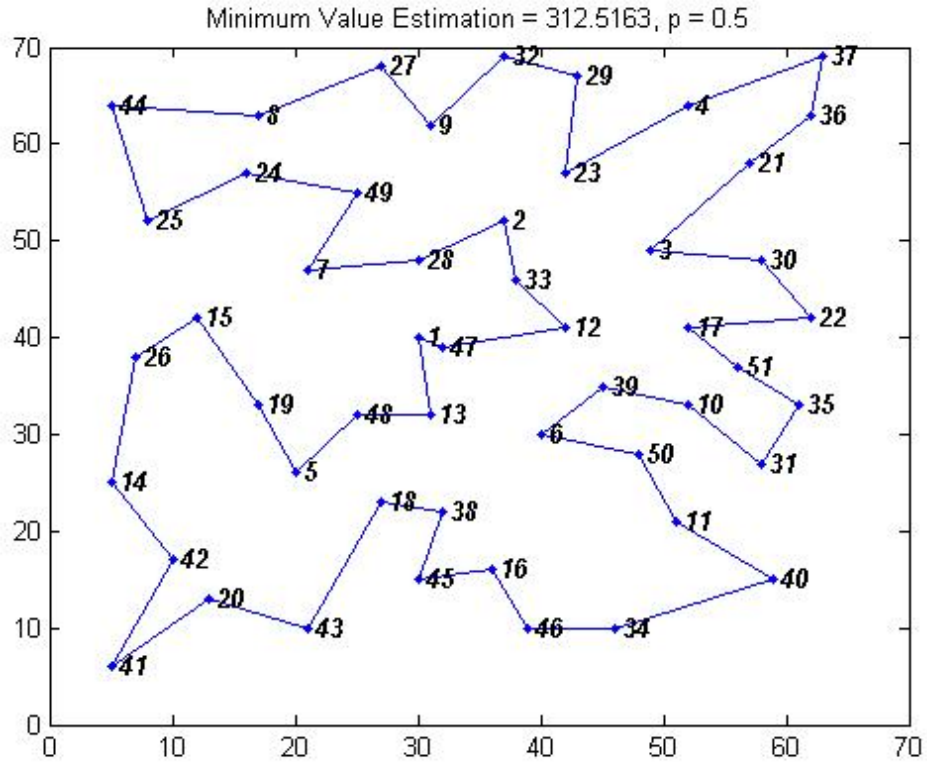
4.6 Επίλυση του TSP & PTSP με χρήση Differential Evolution

Σε αυτήν την ενότητα επιλύεται το στοχαστικό πρόβλημα του πλανόδιου με χρήση του *αλγόριθμου 4.1* και τοπική αναζήτηση 3-opt Η αντικειμενική συνάρτηση προκύπτει από την Εξ. (2.7), (2.8) και (2.9). Τα πακέτα δεδομένων είναι ίδια με την Παράγραφο 3.7.

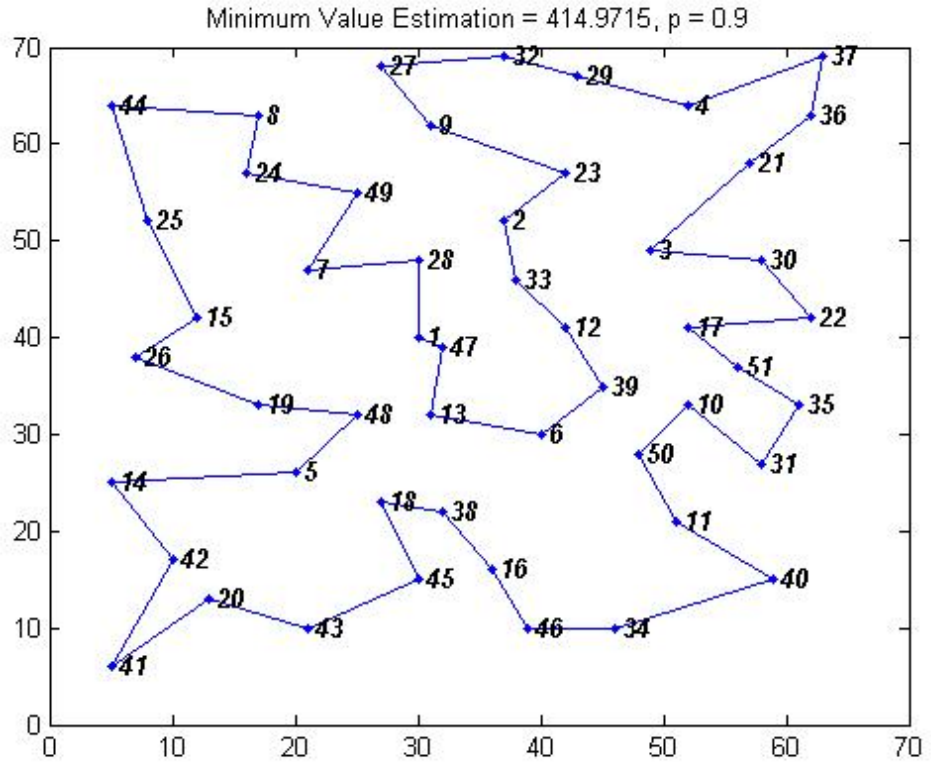
4.6.1. Αποτελέσματα TSP & STSP με χρήση DE



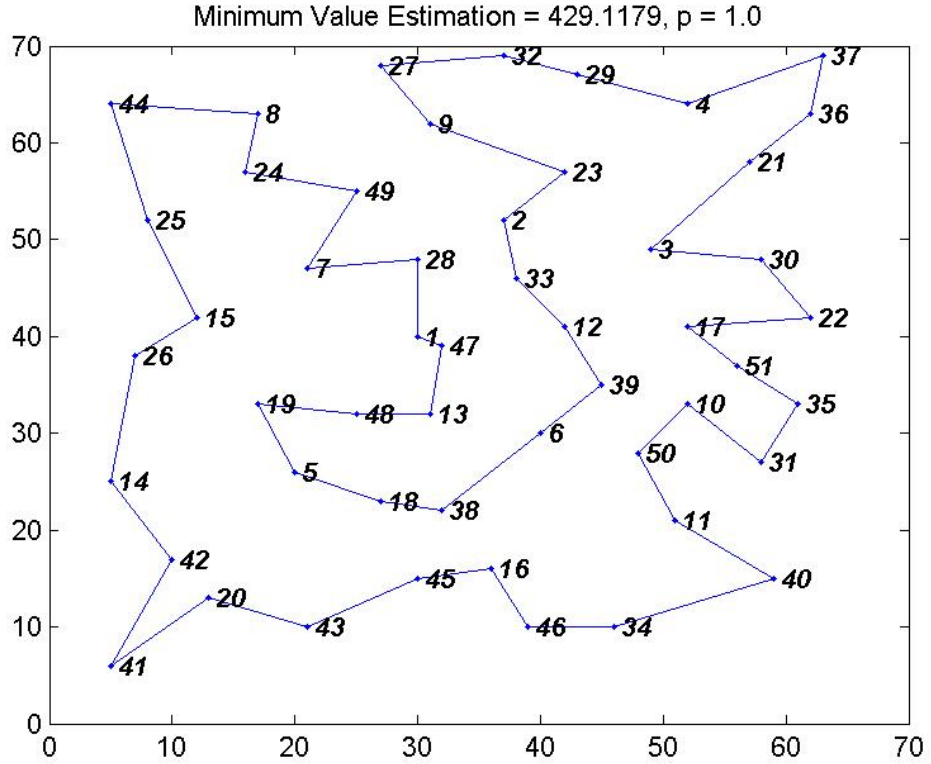
Διάγραμμα 4.24: eil51 – n=51, p=0.1



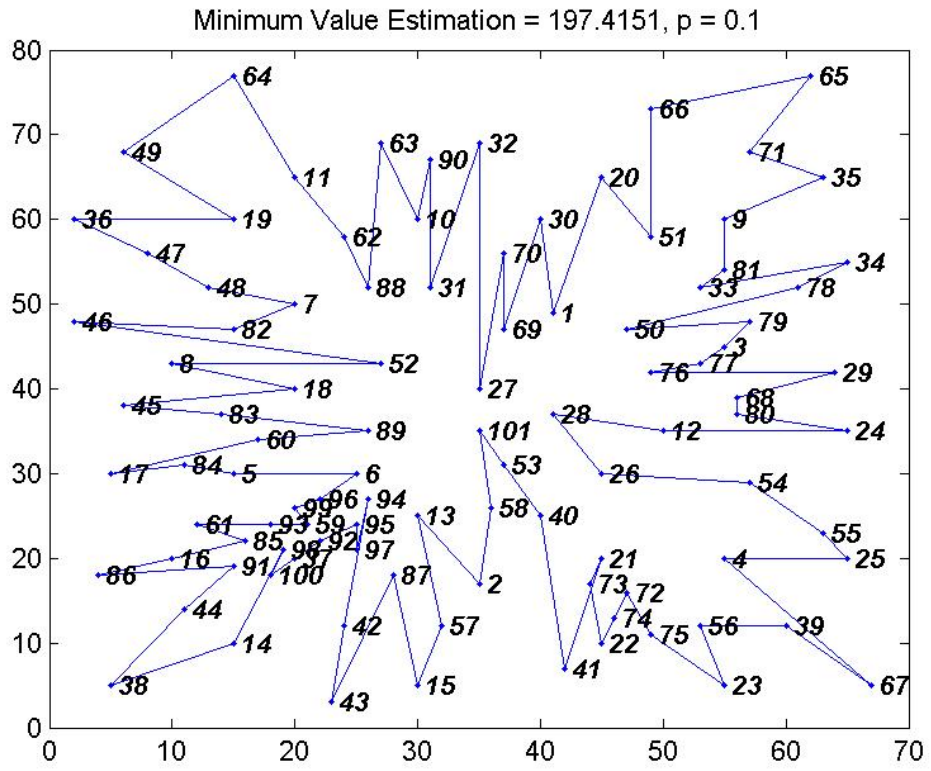
Διάγραμμα 4.25: eil5**1** – n=51, p=0.5



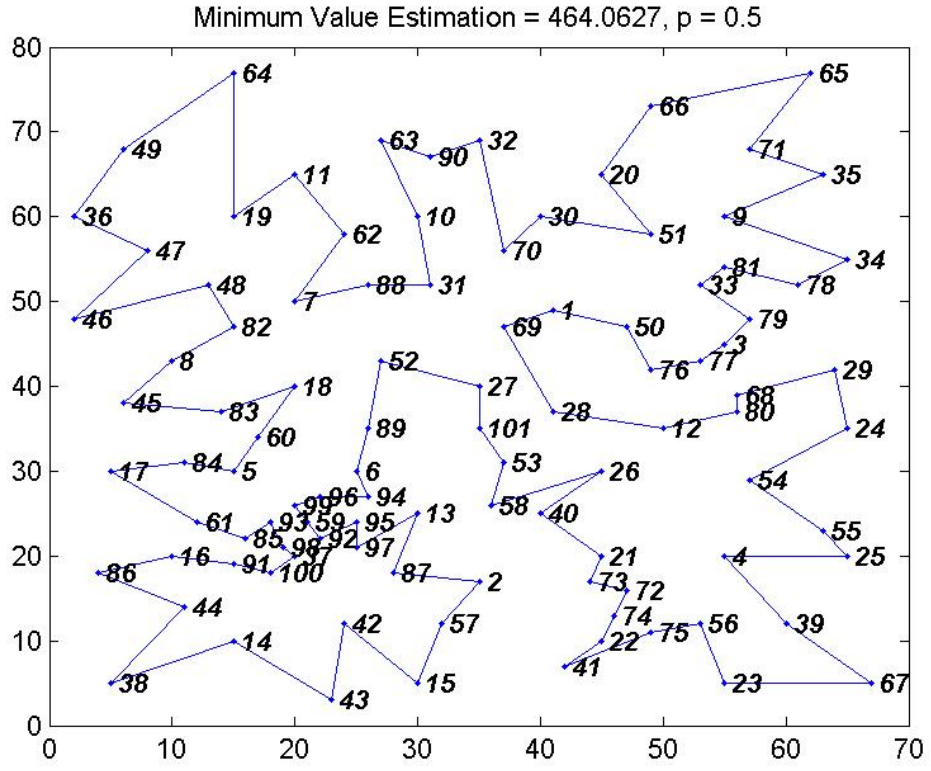
Διάγραμμα 4.26: eil51 – n=51, p=0.9



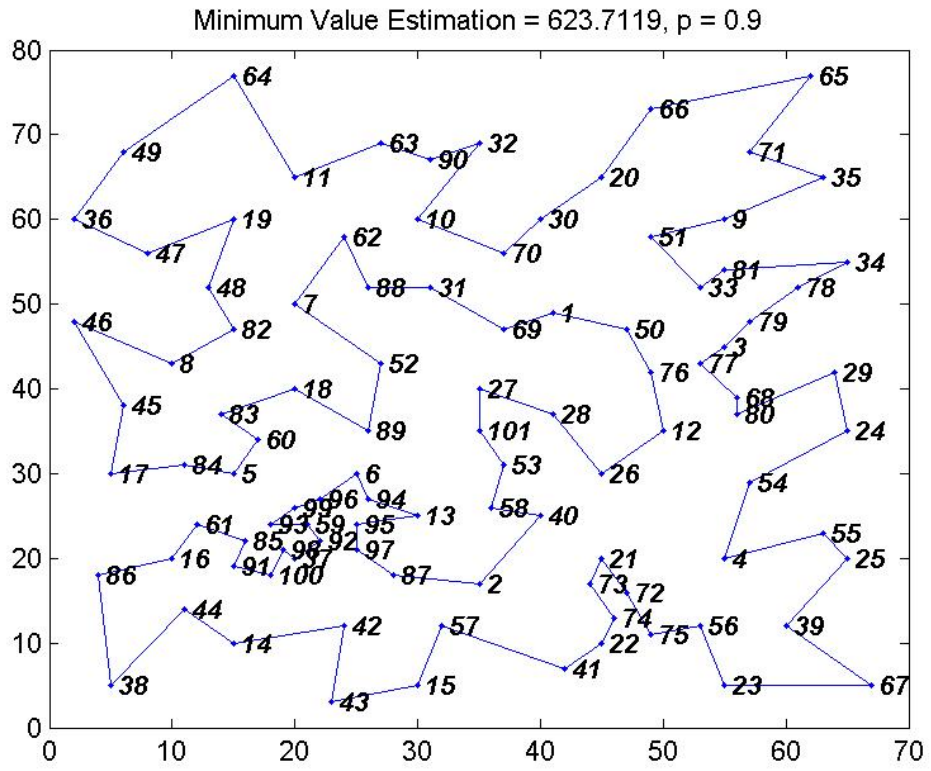
Διάγραμμα 4.27: eil51 – n=51, p=1.0



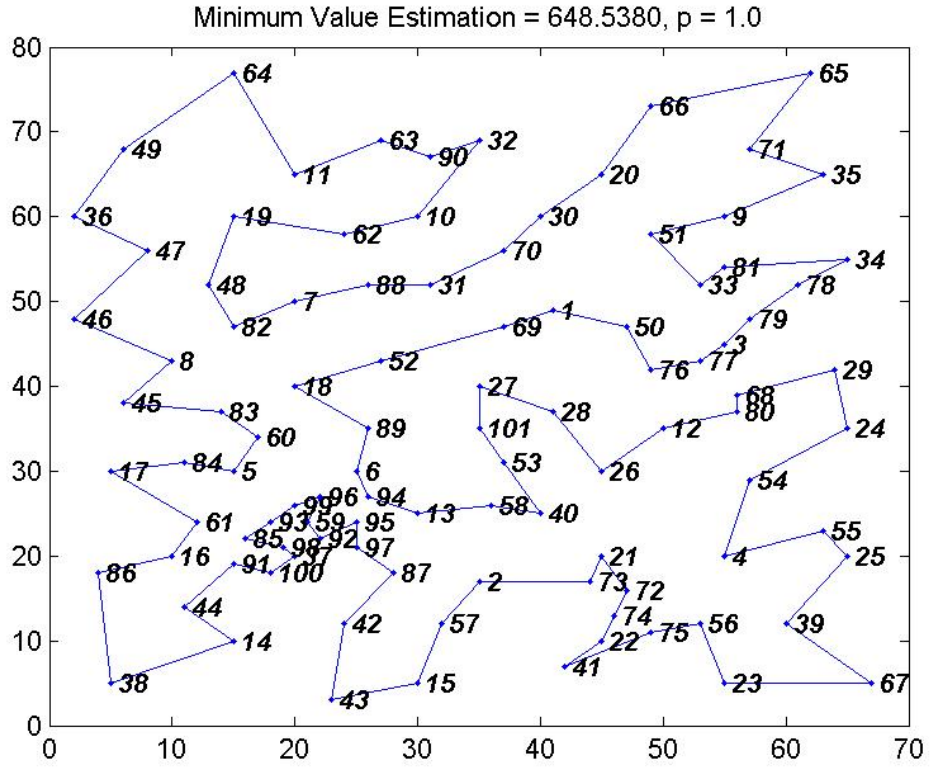
Διάγραμμα 4.28: eil101 – n=51, p=0.1



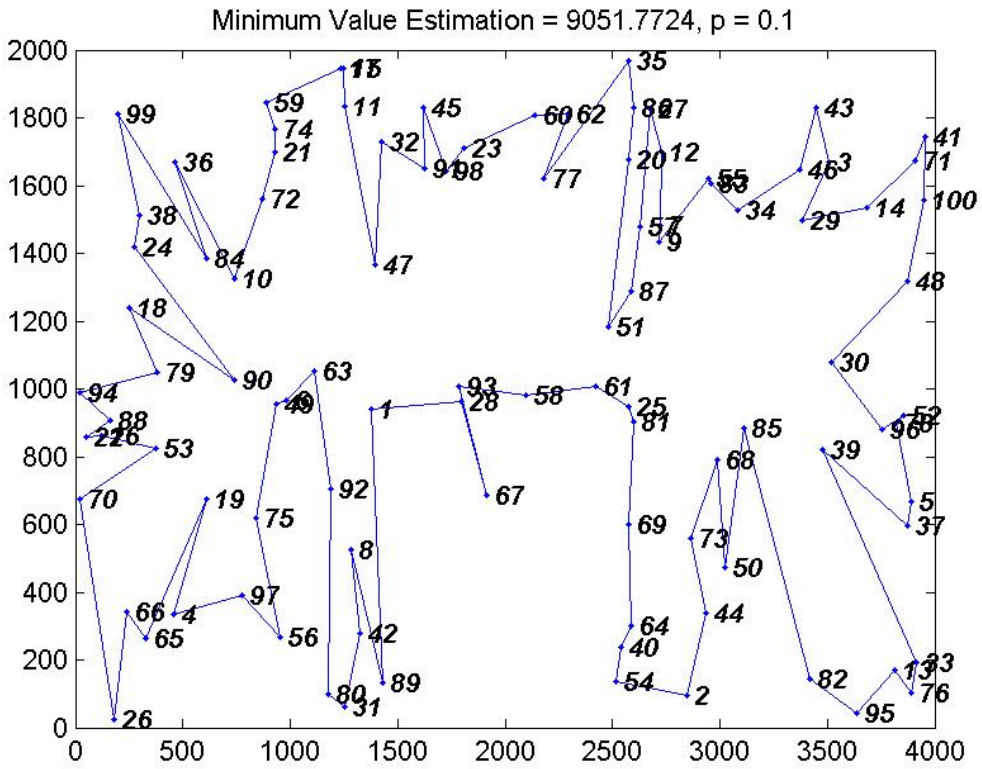
Διάγραμμα 4.29: eil101 – n=51, p=0.5



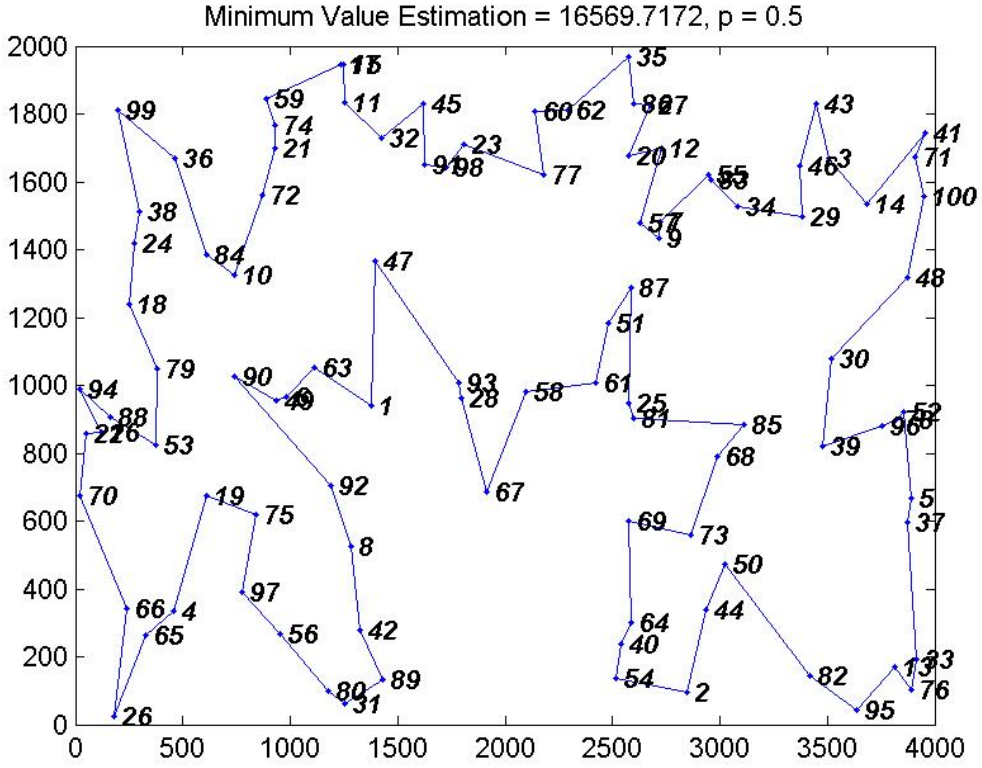
Διάγραμμα 4.30: eil101 – n=51, p=0.9



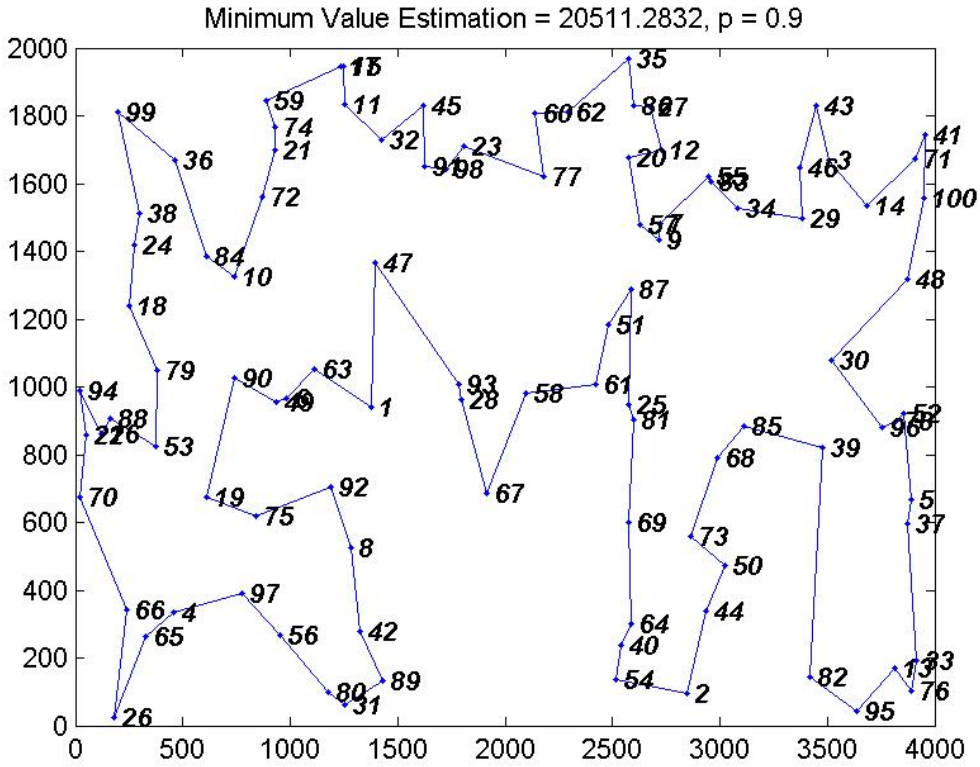
Διάγραμμα 4.31: eil101 – n=51, p=1.0



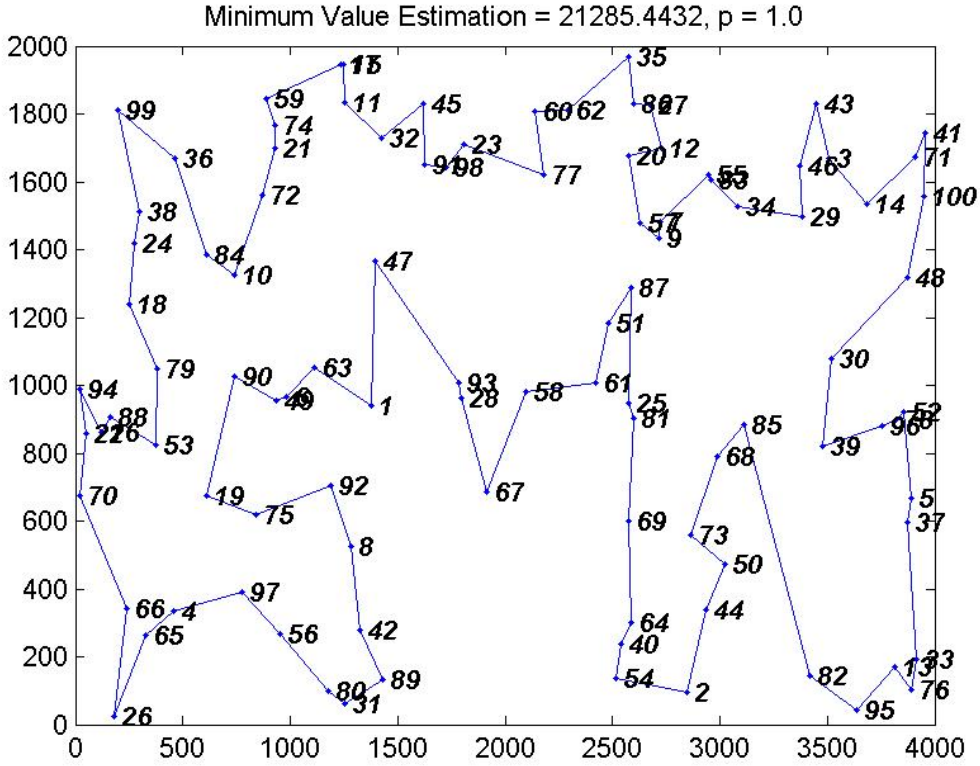
Διάγραμμα 4.32: kroA100 – n=51, p=0.1



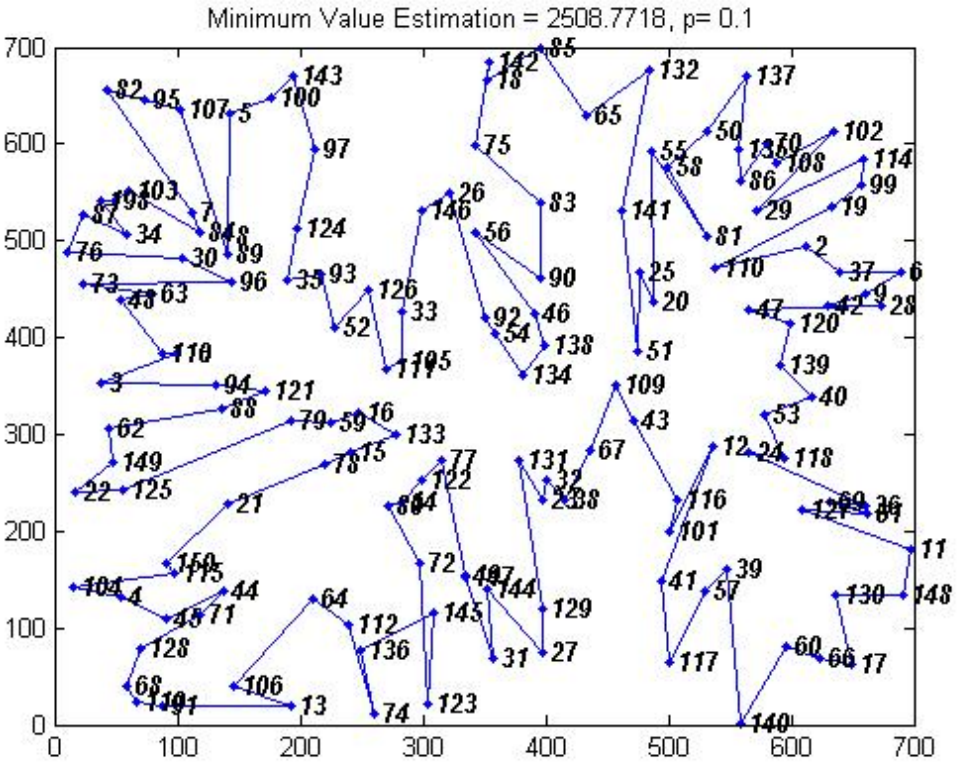
Διάγραμμα 4.33: kroA100 – n=51, p=0.5



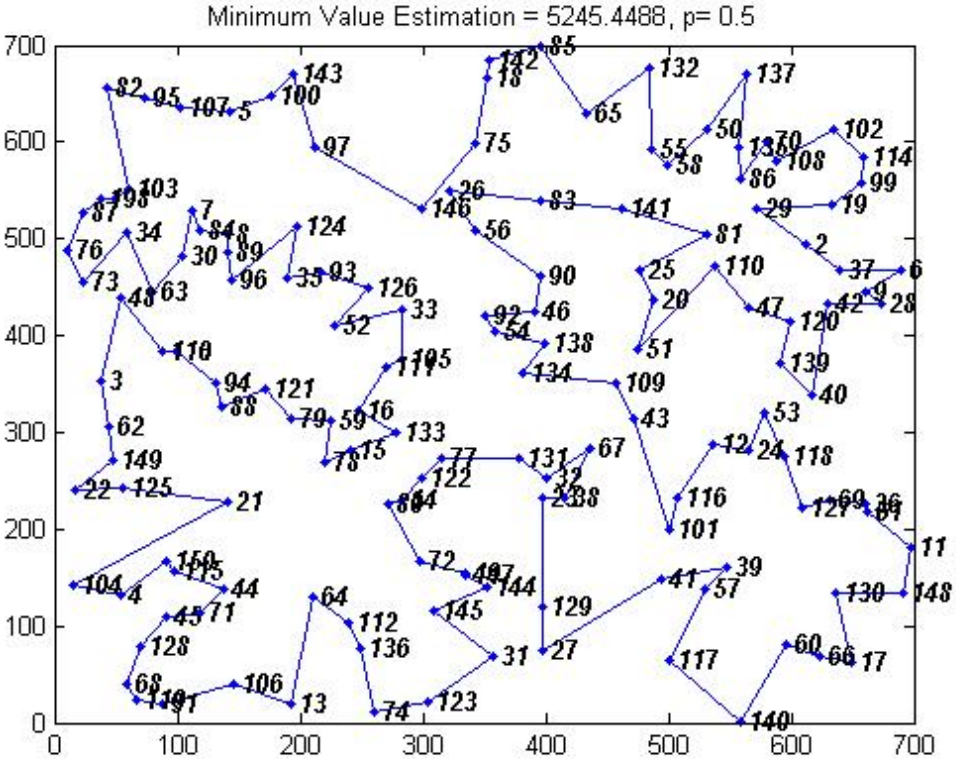
Διάγραμμα 4.34: kroA100 – n=51, p=0.9



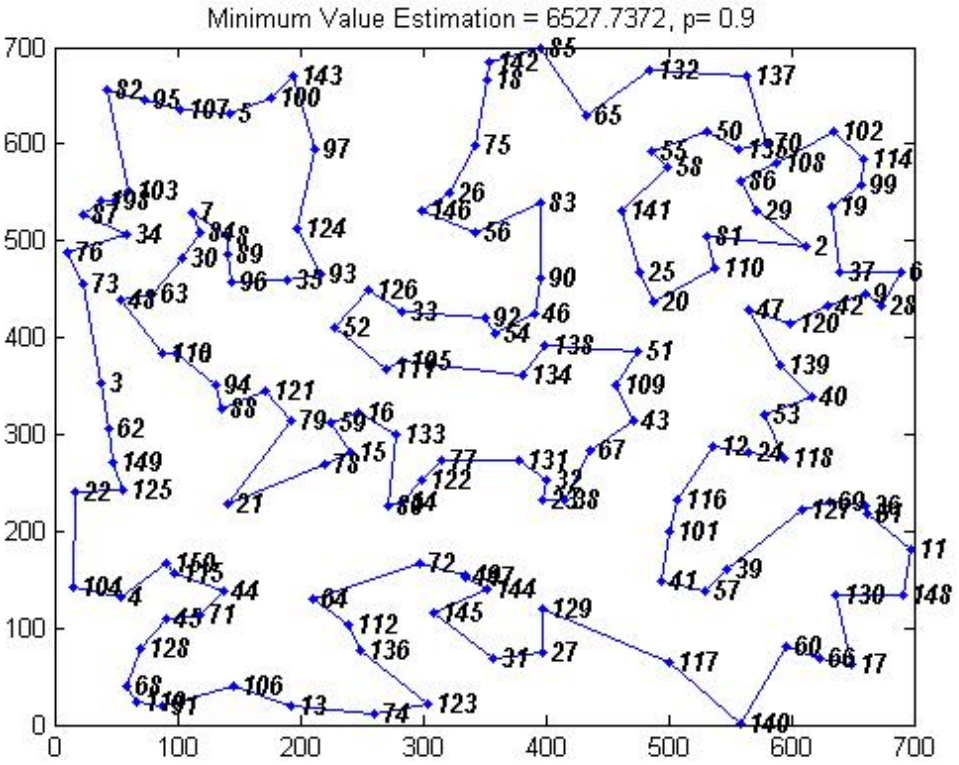
Διάγραμμα 4.35: kroA100 – n=51, p=1.0



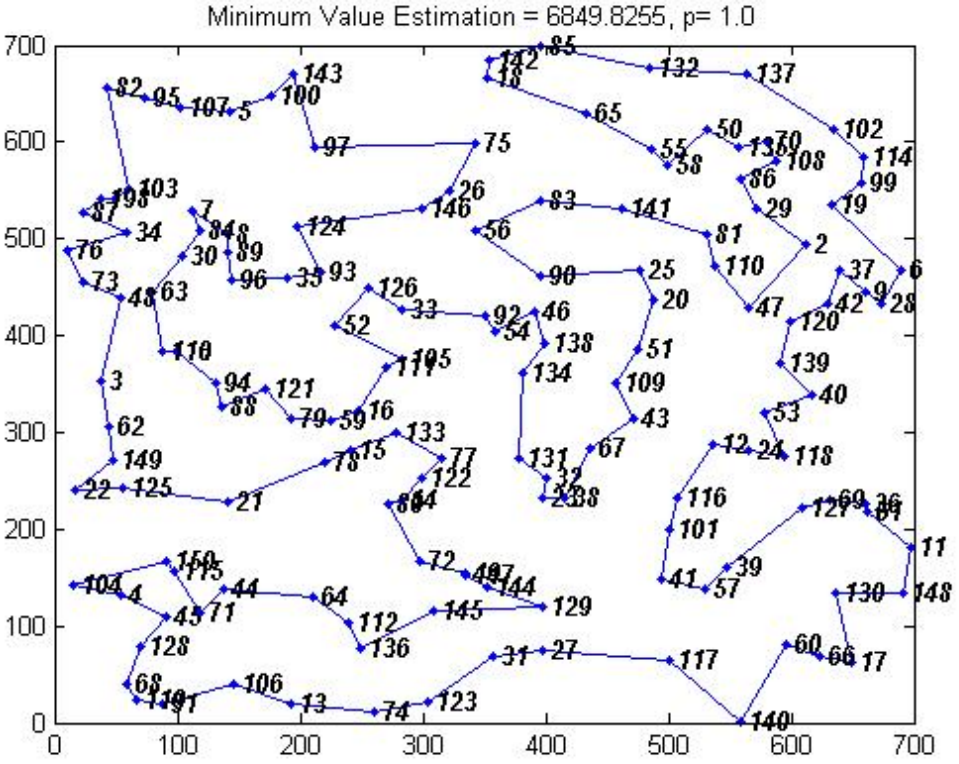
Διάγραμμα 4.36: ch150 – n=51, p=0.1



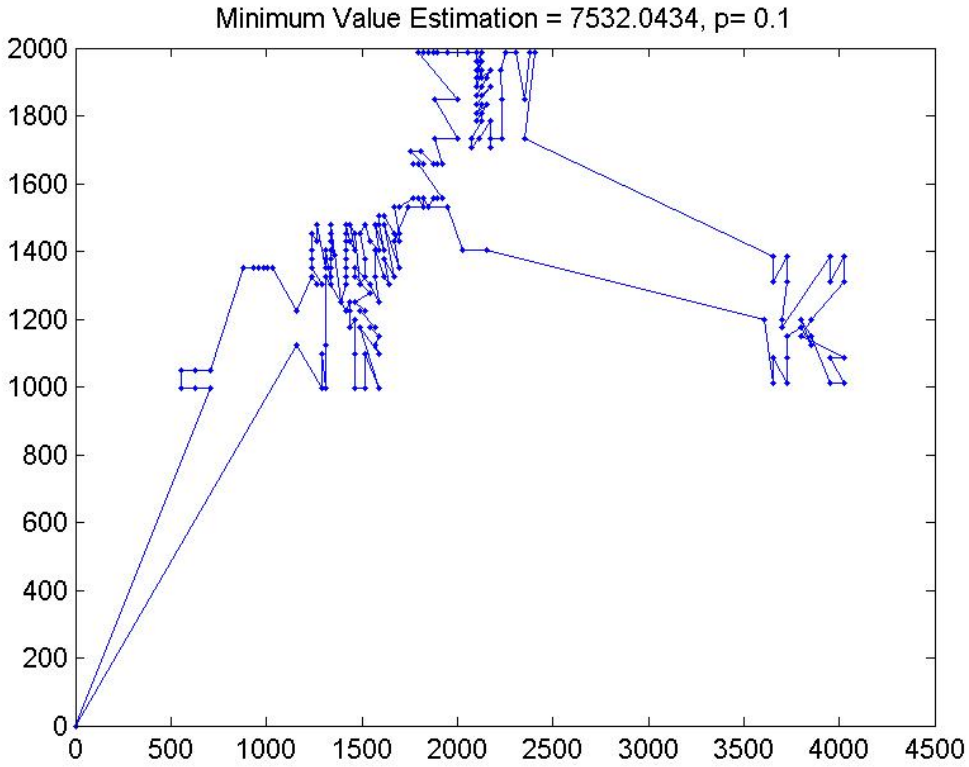
Διάγραμμα 4.37: ch150 – n=51, p=0.5



Διάγραμμα 4.38: ch150 – n=51, p=0.9



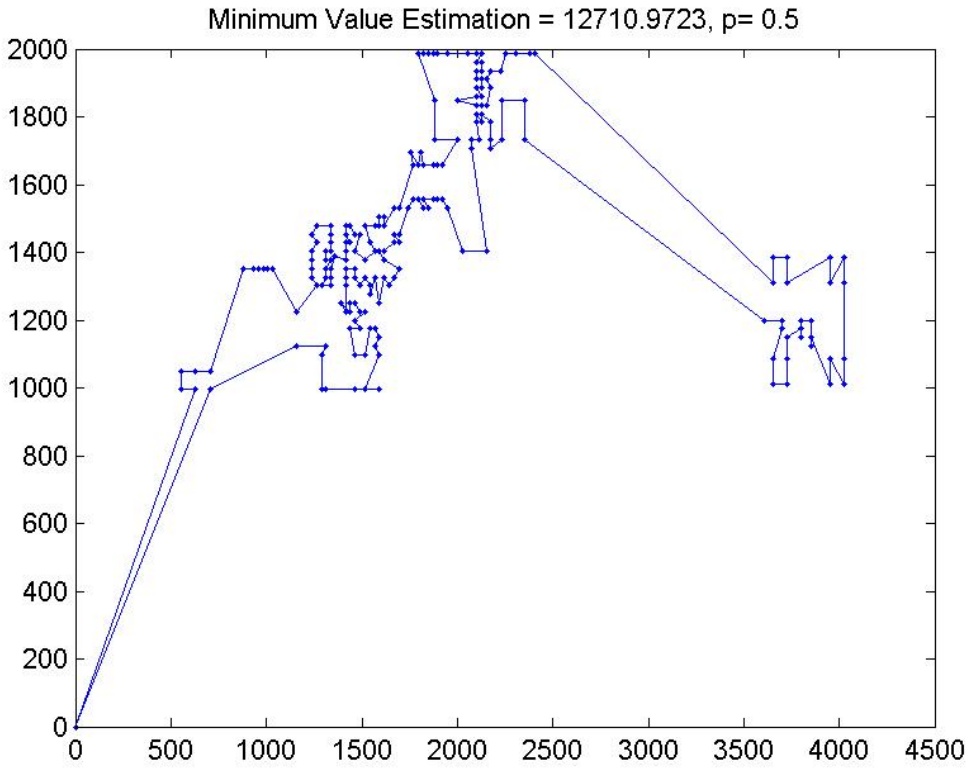
Διάγραμμα 4.39: ch150 – n=51, p=1.0



Διάγραμμα 4.40: d198 – n=51, p=0.1

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

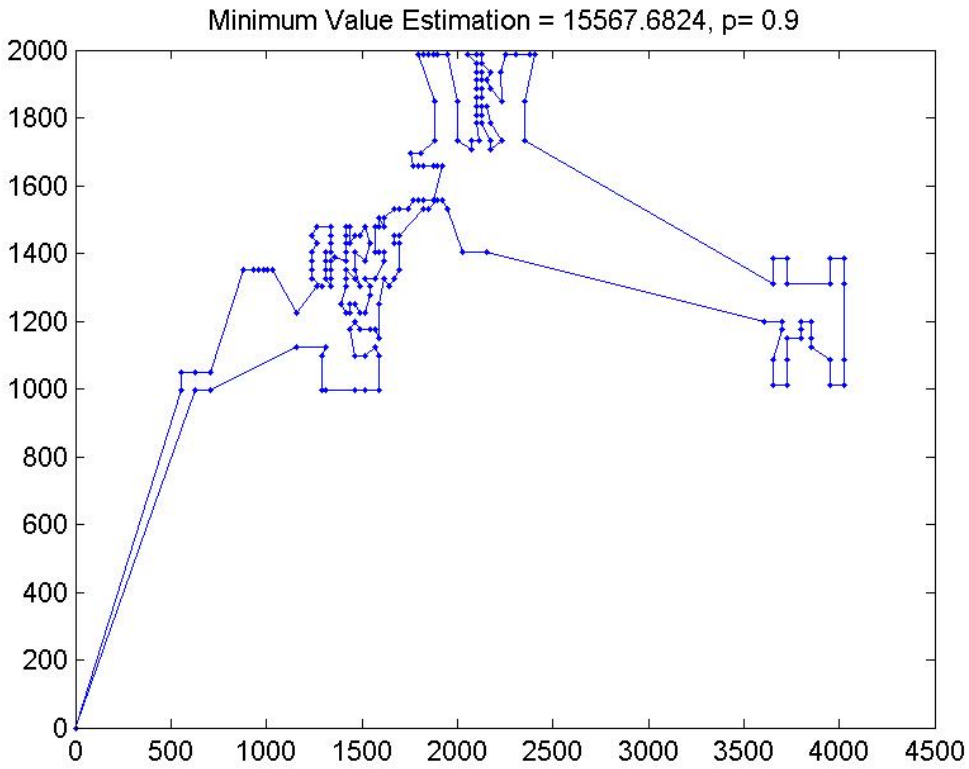
Πίνακας 4.11: Διαδρομή - d198, n=198, p=0.1



Διάγραμμα 4.41: d198 – n=51, p=0.5

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

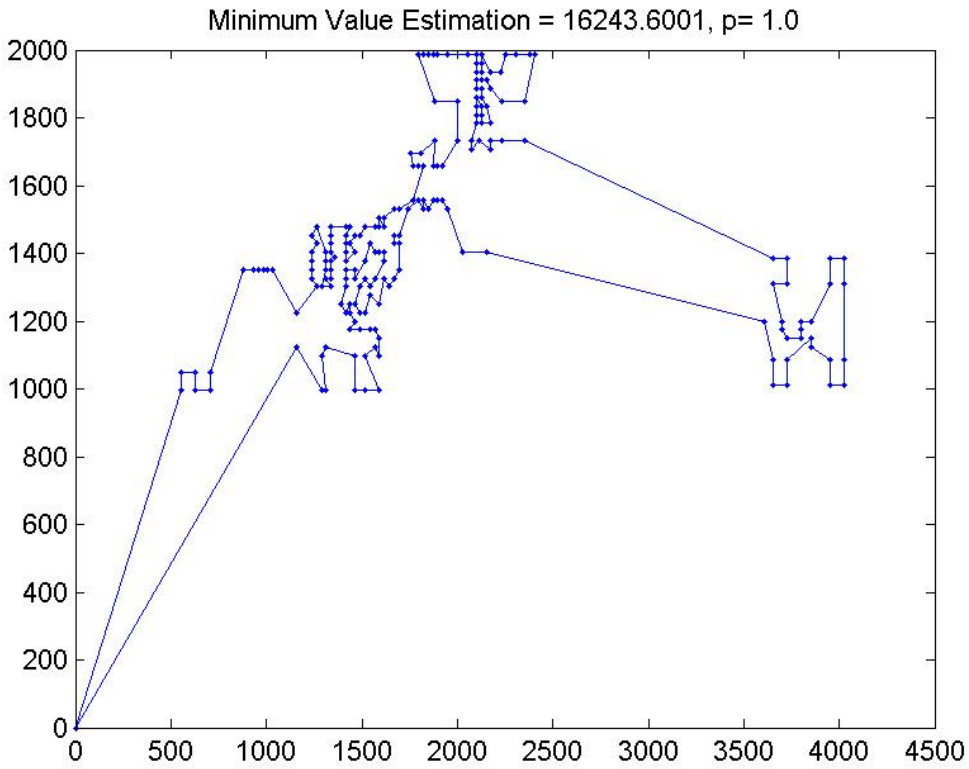
Πίνακας 4.12: Διαδρομή - d198, n=198, p=0.5



Διάγραμμα 4.42: d198 – n=51, p=0.9

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

Πίνακας 4.13: Διαδρομή - d198, n=198, p=0.9



Διάγραμμα 4.43: d198 – n=51, p=1.0

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

Πίνακας 4.14: Διαδρομή - d198, n=198, p=0.1

File-n	*p*	*Best Cost*	*min value estimation*	*iter*	*απόκλιση*
eil 51	0,1	**130,12**	**129,42**	334	**-0,54%**
	0,5	**310,75**	312,51	565	0,57%
	0,9	**407,92**	414,97	312	1,73%
	1	**426**	429,11	262	0,73%
eil101	0,1	**200,03**	**197,41**	786	**-1,31%**
	0,5	**455,65**	464,06	624	1,85%
	0,9	**601,5**	623,71	928	3,69%
	1	**629**	649,54	1207	3,27%
kroA100	0,1	**9074,94**	**9051,77**	972	**-0,26%**
	0,5	**16581,64**	**16569,71**	646	**-0,07%**
	0,9	**20508,77**	20511,28	1157	0,01%
	1	**21282**	21285,44	1153	0,02%
ch150	0,1	**2510,11**	**2508,77**	1401	**-0,05%**
	0,5	**5016,85**	5245,44	1422	4,56%
	0,9	**6292,01**	6527,73	3781	3,75%
	1	**6528**	6849,83	3763	4,93%
d198	0,1	**7504,94**	7532,00	1999	0,36%
	0,5	**12527,56**	12710,97	1671	1,46%
	0,9	**15216,61**	15568,00	1518	2,31%
	1	**15780**	16244,00	1547	2,94%

Πίνακας 4.15: Αποτελέσματα με Differential Evolution

Τα αποτελέσματα του Differential Evolution στο πρόβλημα του πλανόδιου πωλητή είναι πολύ κοντά στη βέλτιστη λύση σε όλα τα προβλήματα ενώ σε ορισμένες περιπτώσεις καταφέρνει να εντοπίσει και καλύτερη λύση από τη μέχρι στιγμής βέλτιστη. Γενικά η απόκλιση από τη βέλτιστη λύση δεν ξεπερνά το 4,93%. Στο eil51 με πιθανότητα 0,1 η λύση βελτιώνεται κατά 0,54% από τη βέλτιστη, στο eil01 με πιθανότητα 0,1 η λύση βελτιώνεται κατά 1,31%, στο kroA100 όταν η πιθανότητα είναι 0,1 και 0,5 έχουμε λύση καλύτερη κατά 0,26% και 0,07% αντίστοιχα. Επίσης παρουσιάζεται εύρεση καλύτερης λύση και σε μεγαλύτερο πρόβλημα, στο ch150 με πιθανότητα 0,1 έχουμε βελτίωση της λύση κατά 0,05%, αν και στις περιπτώσεις με μεγαλύτερη πιθανότητα, η απόκλιση από τη βέλτιση λύση είναι μεγαλύτερη σε σχέση με τα υπόλοιπα προβλήματα.

Κεφάλαιο 5

Σύγκριση των δύο αλγορίθμων

5.1 Σύγκριση των δυο μεθόδων στο VRP

Στον ακόλουθο πίνακα παρουσιάζονται τα αποτελέσματα των δύο αλγορίθμων για το πρόβλημα δρομολόγησης οχημάτων.

		Genetic Algorithm			*Differential Evolution*		
File-n	**Best Cost**	**GA**	**iter**	**απόκλιση**	**DE**	**iter**	**απόκλιση**
par1-51	**524,61**	539,68	1388	**2,87%**	**524,61**	955	**0,00%**
par2-76	**835,26**	884,48	2151	**5,89%**	853,47	1519	**2,18%**
par3-101	**826,14**	856,86	3850	**3,72%**	850,36	5857	**2,93%**
par4-151	**1028,42**	1149,80	9927	**11,80%**	1127,60	9497	**9,64%**
par5-200	**1291,29**	1439,90	29758	**11,51%**	1403,20	29425	**8,67%**
par6-51	**555,43**	566,02	1429	**1,91%**	556,67	987	**0,22%**
par7-76	**909,68**	954,34	2464	**4,91%**	947,96	3340	**4,21%**
par8-101	**865,94**	900,69	4977	**4,01%**	893,35	2778	**3,17%**
par9-151	**1162,55**	1243,90	13600	**7,00%**	1215,70	20316	**4,57%**
par10-200	**1395,55**	1592,90	37616	**14,14%**	1492,97	35417	**6,98%**
par11-121	**1042,11**	1203,00	3226	**15,44%**	1046,40	2797	**0,41%**
par12-101	**819,56**	890,46	2747	**8,65%**	861,76	1861	**5,15%**
par13-121	**1541,19**	1674,30	5793	**8,64%**	1658,90	3706	**7,64%**
par14-101	**866,37**	903,31	2177	**4,26%**	898,62	3785	**3,72%**

Πίνακας 5.1: Συγκριτικός πίνακας των δύο αλγορίθμων

Σε κάθε πρόβλημα που εξετάστηκε ο Differential Evolution καταφέρνει να εντοπίσει καλύτερη λύση από τον Γενετικό Αλγόριθμο. Ειδικότερα στο πρόβλημα par_1 εντοπίζει τη βέλτιστη λύση 524,61. Η απόκλιση στον DE δεν ξεπερνά το 8,8% σε όλα τα προβλήματα ενώ GA σε ορισμένα αγγίζει και το 15,5%. Ιδιαίτερα στο πρόβλημα par_11 η λύση που εντοπίζει ο Γενετικός Αλγόριθμος έχει απόκλιση από τη βέλτιστη λύση 15,44% ενώ η αντίστοιχη λύση του Differential Evolution σχεδόν εντοπίζει τη βέλτιστη με απόκλιση 0,41% και σε αριθμό γενεών 2797 δηλαδή 429 επαναλήψεις λιγότερες από τον γενετικό.

5.2 Σύγκριση των δυο μεθόδων στο SVRP

Στον ακόλουθο πίνακα παρουσιάζονται τα αποτελέσματα των δύο αλγορίθμων για το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση.

			Genetic Algorithm			*Differential Evolution*		
File-n	***dem***	***Best Cost***	***min value estimation***	***iter***	***απόκλιση***	***min value estimation***	***iter***	***απόκλιση***
par - 51	0	**524,61**	542,62	489	**3,43%**	537,42	967	**2,44%**
	1		543,80	643	**3,66%**	538,35	944	**2,62%**
	2		540,59	486	**3,05%**	536,69	649	**2,30%**
par2-76	0	**835,26**	867,86	1158	**3,90%**	862,39	978	**3,25%**
	1		878,45	2364	**5,17%**	862,72	1007	**3,29%**
	2		886,87	1194	**6,18%**	865,86	719	**3,66%**
par3-101	0	**826,14**	850,66	2269	**2,97%**	839,4	2057	**1,61%**
	1		854,51	2401	**3,43%**	866,15	1542	**4,84%**
	2		854,18	1658	**3,39%**	851,52	2083	**3,07%**

Πίνακας 5.2: Συγκριτικός πίνακας των δύο αλγορίθμων

Σε κάθε πρόβλημα που εξετάστηκε ο Differential Evolution καταφέρνει να εντοπίσει καλύτερη λύση από τον Γενετικό Αλγόριθμο, με εξαίρεση το par_3 και dem=1. Ειδικότερα, όπως έχει ήδη σχολιαστεί (Κεφάλαιο 4, παράγραφος 4.5) στο par_3, με dem=0, ο DE δίνει καλύτερη λύση μέσα από το στοχαστικό πρόβλημα από ότι δίνει ο ίδιο ο αλγόριθμός στο απλό πρόβλημα δρομολόγησης οχημάτων (Πίνακας 5.1).

5.3 Σύγκριση των δυο μεθόδων στο TSP & STSP

Στον ακόλουθο πίνακα παρουσιάζονται τα αποτελέσματα των δύο αλγορίθμων για το πρόβλημα του πλανόδιου πωλητή.

			Genetic Algorithm			*Differential Evolution*		
File-n	**p**	**Best Cost**	**min value estimation**	**iter**	**απόκλιση**	**min value estimation**	**iter**	**απόκλιση**
eil 51	0,1	**130,12**	**129,42**	424	**-0,54%**	**129,42**	334	**-0,54%**
	0,5	**310,75**	316,52	293	1,86%	312,52	565	0,57%
	0,9	**407,92**	416,12	610	2,01%	414,97	312	1,73%
	1	**426**	438,13	370	2,85%	429,12	262	0,73%
eil101	0,1	**200,03**	**197,34**	1355	**-1,35%**	197,42	786	**-1,31%**

	0,5	**455,65**	464,48	1479	1,94%	464,06	624	1,85%
	0,9	**601,5**	629,71	1199	4,69%	623,71	928	3,69%
	1	**629**	667,61	1359	6,14%	649,54	1207	3,27%
kroA100	0,1	**9074,94**	9175,32	992	1,11%	**9051,77**	972	**-0,26%**
	0,5	**16581,64**	17135,07	991	3,34%	**16569,72**	646	**-0,07%**
	0,9	**20508,77**	22590,58	926	10,15%	20511,28	1157	0,01%
	1	**21282**	22768,04	1486	6,98%	21285,44	1153	0,02%
ch150	0,1	**2510,11**	2530,71	1324	0,82%	**2508,77**	1401	**-0,05%**
	0,5	**5016,85**	5264,08	2337	4,93%	5245,44	1422	4,56%
	0,9	**6292,01**	6617,50	3664	5,17%	6527,73	3781	3,75%
	1	**6528**	6946,53	3190	6,41%	6849,83	3763	4,93%
d198	0,1	**7504,94**	7770,59	1987	3,54%	7532,00	1999	0,36%
	0,5	**12527,56**	12965,90	2198	3,50%	12710,97	1671	1,46%
	0,9	**15216,61**	15857,86	1585	4,21%	15568,00	1518	2,31%
	1	**15780**	16557,83	1583	4,93%	16244,00	1547	2,94%

Πίνακας 5.3: Συγκριτικός πίνακας των δύο αλγορίθμων

Σε κάθε πρόβλημα που εξετάστηκε ο Differential Evolution καταφέρνει να εντοπίσει καλύτερη λύση από τον Γενετικό Αλγόριθμο. Με χρήση του DE η απόκλιση δεν είναι μεγαλύτερη του 4,93% ενώ με τον GA η απόκλιση κυμαίνεται στο 6% ενώ σε μία περίπτωση φτάνει και το 10,15%. Ιδιαίτερα στο πρόβλημα kroA100 με πιθανότητα 0,9 ο GA έχει απόκλιση από τη βέλτιστη κατά 10,15% ενώ ο DE εντοπίζει τη βέλτιστη με απόκλιση 0.01%. Επίσης οι επαναλήψεις που χρειάζεται ο DE είναι λιγότερες ενώ εντοπίζει καλύτερη λύση από τη βέλτιστη σε περισσότερες περιπτώσεις. Στο eil51 με πιθανότητα 0.1 οι δύο μέθοδοι βρίσκουν την ίδια λύση 129,42 η οποία είναι καλύτερη από τη βέλτιστη κατά 0,54%, στο eil101 με πιθανότητα 0,1 πάλι εντοπίζεται καλύτερη λύση και από τις δύο μεθόδους αλλά η λύση του DE (197,42) είναι κατά 0,04% καλύτερη από του GA. Ο DE βρίσκει καλύτερη λύση από τη μέχρι στιγμής βέλτιστη και στο kroA100 με πιθανότητα 0,1 και 0,5 και η απόκλιση είναι 0,26% και 0,07% αντίστοιχα με τιμή 9051,77 και 16569,72. Τέλος στο πρόβλημα ch150 ο DE βελτιώνει τη βέλτιστη λύση κατά 0,05% με τιμή 2508,77.

Κεφάλαιο 6
Συμπεράσματα

Στα προβλήματα που αναλύθηκαν παραπάνω ο Differential Evolution δουλεύει καλύτερα από τον Genetic Algorithm. Τα αποτελέσματα του πρώτου αλγορίθμου παρουσιάζουν μικρότερο σφάλμα και οι τιμές βρίσκονται πιο κοντά στη βέλτιστη λύση στο Πρόβλημα Δρομολόγησης Οχημάτων (VRP) ενώ και στο SVRP τα αποτελέσματα είναι ανάλογα. Στο Πρόβλημα του Πλανόδιου Πωλητή (TSP) ο DE εμφανίζει και πάλι καλύτερα αποτελέσματα. Στο PTSP ενώ και οι δύο αλγόριθμοι δίνουν ικανοποιητικά αποτελέσματα ο DE καταφέρνει να εντοπίσει καλύτερη λύση από τις μέχρι τώρα, σε περισσότερες περιπτώσεις.

Για μελλοντική έρευνα προτείνεται η επίλυση των προβλημάτων και με άλλες μεθόδους καθώς και η δημιουργία πιο αποδοτικών αλγορίθμων σε μεγαλύτερα προβλήματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Altinkemer, K., & Gavish B. (1991). Parallel savings based heuristics for the delivery problem. Operations Research, 39(3), 456-469.

Baker B.M. & Ayechew M.A. (2003). A genetic algorithm for the vehicle routing problem. Computers and Operations Research, 30(5), 787-800.

Barbarosoglou G. & Ozgur D. (1999). A tabu search algorithm for the vehicle routing problem. Computers and Operations Research, 26, 255-270.

Berger J. & Barkaoui M. (2003). A hybrid genetic algorithm for the capacitated vehicle routing problem. In Proceedings of the genetic and evolutionary computation conference (pp. 646-656). Chicago.

Bertsimas DJ (1988). Probabilistic combinatorial optimization problem. PhD thesis, MIT, Cambridge, MA, USA.

Bianchi L. (2006). Ant colony optimization and local search for the probabilistic travelling salesman problem: a case study in stochastic combinatorial optimization. PhD thesis, Universite Libre de Bruxelles, Belgium.

Bianchi Leonora, Marco Dorigo, Luca Maria Gambardella (2003). New approaches for solving the Probabilistic Traveling Salesman Problem.

Bianchi Leonora, Monaldo Mastrolilli, Mauro Birattari, Max Manfrin, Marco Chiarabdini, Luis Paquete, Olivia Rossi-Doria. Report for Task,Research on the Vehicle Routing Problem with Stochastic Demand.

Bodin L. & Golden B. (1981). Classification in vehicle routing and scheduling. Networks, 11, 97-108.

Bodin L., Golden B., Assad A., Ball M. (1983). The state of the art in the routing and scheduling of vehicles and crews. Computers and Operations Research, 10, 63-212.

Bullnheimer B., Hartl P.F. & Strauss C. (1999). An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 89, 319-328.

Christofides N., Mingozzi A.& Toth P. (1979),. The vehicle routing problem. In N.Christofides, A.Mingozzi, P.Toth & C.Sandi (Eds.), Combinatorial optimization. Chichester: Wiley

Clarke G. & Wright J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12, 568-581.

Cordeau J.F., Gendreau M., Laporte G., Potvin J.Y. & Semete F.(2002). A guide to vehicle routing heyristics. Journal of the Operational Research Society, 53, 512-522.

Cordeau J.F., Gendreau M., Hertz A., Laport G. & Sormany J.S. (2005). New heurisrics for the vehicle routing problem. In A.Langevine & D.Riopel (Eds.), Logistics systems: Design and optimization (pp. 279-298). Wiley and Sons.

Destrochers M. & Verhoog T.W. (1989). A matching based savings algorithm for the vehicle routing problem. Les Cahiers du GERAD G-89-04. Ecole des Hautes Etudes Commerciales de Montreal.

Fisher M.L. & Jaikumar R. (1981). A generalized assignment heuristic for vehicle routing. Networks, 11, 109-124.

Foster B.A. & Ryan D.M. (1976). An integer programming approach to the vehicle scheduling problem. Operations Research, 27, 367-384.

Gendreau M., Hertz A. & Laporte G. (1994). A tabu search heuristic for the vehicle routing problem. Management Science, 40, 1276-1290.

Gillett B.E. & Miller L.R. (1974). A heuristic algorithm for the vehicle dispatch problem. Operations Research, 22, 240-349.

Jaillet P. (1985). Probabilistic travelling salesman problems. PhD thesis, MIT, Cambridge, MA, USA.

Jaillet P. (1988). A priori solution of a travelling salesman problem in which a random subset of customers are visited. Operations Research; 36(6):929-36.

Kenneth V.Price, Rainer M.Storn, Jouni A.Lampinen (2005). Differential Evolution – A Practical Approach to Global Optimization.

Lin S. (1965). Computer solutions of the traveling salesman problem. Bell Systems Technical Journal, 44, 2245-2269.

Liu YH. (2007). A hybrid scatter search for the probabilistic travelling salesman problem. Computers and Operations Research, 34(10):2949-63.

S.Lin and B.W.Kernigham (1973). 'An effective heuristic algorithm for the traveling salesman problem', Operational Research, vol.21,pp.498-516.

Marinakis Y., Migdalas A. & Pardalos P.M. (2007a). Multiple phase neighborhood, search GRASP based on Lagrangean relaxation and random backtraching Lin Kernigham for the traveling salesman problem. Journal of Combinatorial Optimization. (Available on line doi: 10.1007/s10878-007-9104-2).

Marinakis Y., Marinaki M. & Dounias G. (2008). Honey bees mating optimization

algorithm for the vehicle routing problem. In N.Krasnogor, G.Nicosia, M.Pavone & D.Pelta (Eds), Nature inspired cooperative strategies for optimization – NISCO 2007. Studies in computational intelligence (Vol. 129, pp.139-148). Berlin: Springer-Verlag.

Mester B. & Braysy O. (2005). Active guided evolution strategies for the large scale vehicle routing problems with time windows. Computers and Operations Research, 32, 1593-1614.

Mester B. & Braysy O. (2007). Active guided evolution strategies for the large scale capacitated vehicle routing problems. Computers and Operations Research, 34, 2964-2975.

Mole R.H. & Jameson S.R. (1976). A sequential route-building algorithm employing a generalized savings criterion. Operations Research Quarterly, 27, 503-511.

Osman I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems. Annals of Operations Research, 41, 421-451.

Powell WB., Jaillet P., Odoni A., (1995). Stochastic and dynamic networks and routing. In: Ball MO, Magnanti TL, Momma CL, Nemhauser GL, editors. Network routing, handbooks in operations research and management science, vol 8. Amsterdam: Elsevier Science B.V. p. 141-295.

Prins C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research, 31, 1985-2002.

Reimann M., Stummer M. & Doerner K. (2002). A savings based ant system for the vehicle routing problem. In Proccedings of the genetic and evolutionary computation conference (pp. 1317-1326). New York.

Rego C. (1998). A subpath ejection method for the vehicle routing problem. Management Science, 44, 1447-1459.

Rego C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. Parallel Computing, 27(3), 201-222.

Rochat Y. & Taillard E.D. (1995). Probabilistic diversification and intensification in local search for vehicle routing problem. Journal of Heuristics, 1. 147-167.

Taillard E.D. (1993). Parallel iterative search methods for vehicle routing problems. Networks, 23, 661-672.

Tarantillis C.D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. Computers and Operations Research, 32(9), 2309-

2327.

Tarantillis C.D. & Kiranoudis C.T. (2002). BoneRoute: An adaptive memory-based method for effective fleet management. Annals of Operations Research, 115(1), 227-241.

Toth P. & Vigo D. (2003). The granular tabu search (and its application to the vehicle routing problem). INFORMS Journal on Computing, 15(4), 333-348.

Wark P. & Holt J. (1994). A repeated matching heuristic for the vehicle routing problem. Journal of the Operations Research Society, 45, 1156-1167.

Xu J. & Kelly J.P. (1996).A new network flow-based tabu search heuristic for the vehicle routing problem. Transportation Science, 30, 379-393.