ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Επίλυση "πράσινων" προβλημάτων δρομολόγησης (ΠΔΟ) με περιορισμό χωρητικότητας και χρόνους εξυπηρέτησης με χρήση του αλγορίθμου βελτιστοποίησης Αποικίας Μυρμηγκιών.

ΑΛΕΞΑΝΔΡΟΣ ΛΟΡΕΝΤΣ

ΑΜ: 2017010141

Επιβλέπων καθηγητής: Ιωάννης Μαρινάκης

Χανιά, Φεβρουάριος 2025

Περιεχόμενα

1.ΕΙΣΑΓΩΓΗ

1.1 ΠΕΡΙΛΗΨΗ

Το Πρόβλημα Δρομολόγησης Οχημάτων (ΠΔΟ) αποτελεί αναπόσπαστο κομμάτι της Εφοδιαστικής Αλυσίδας λόγω της συνεχώς αυξανόμενης κίνησης και ζήτησης στον τομέα του εμπορίου και όχι μόνο παγκοσμίως. Ανά τα χρόνια έχει μελετηθεί αρκετά σταθμίζοντας διαφορετικό παράγοντα κάθε φορά ανάλογα με τις ανάγκες και τους περιορισμούς που τίθενται.

Η συγκεκριμένη εργασία επιχειρεί να προσαρμόσει το πρόβλημα στις οικολογικές συνθήκες της εποχής μας δημιουργώντας ένα «πράσινο» ΠΔΟ-“Green VRP” όπου ελαχιστοποιείται η κατανάλωση ενέργειας του συστήματος μη παραμελώντας την ανάγκη για ελαχιστοποίηση του κόστους – κρίσιμου σημείου για τη βιωσιμότητα των επιχειρήσεων . Μιμούμενοι τη φύση και πιο συγκεκριμένα τη συμπεριφορά των μυρμηγκιών, χρησιμοποιώντας τον αλγόριθμο βελτιστοποίησης Αποικίας μυρμηγκιών (ACO), αναπτύσσεται κώδικας σε προγραμματιστικό περιβάλλον Matlab με σκοπό τη βελτιστοποίηση των διαδρομών ενός οχήματος λαμβάνοντας υπόψη τον χρόνο εξυπηρέτησης και τη χωρητικότητα του με γνώμονα το ενεργειακό αποτύπωμα.

1.2 ΕΦΟΔΙΑΣΤΙΚΗ ΑΛΥΣΙΔΑ

Η ανάγκη για συνεχή εφοδιασμό και διατήρηση αποθεμάτων υπάρχει από την αρχαιότητα. Ένα χαρακτηριστικό παράδειγμα αποτελεί η εκστρατεία του Μεγάλου Αλεξάνδρου προς την Ασία, η οποία θα ήταν αδύνατη χωρίς ένα αποτελεσματικό σύστημα εφοδιασμού. Ωστόσο, η Διαχείριση Εφοδιαστικής Αλυσίδας (ΔΕΑ), όπως τη γνωρίζουμε σήμερα, άρχισε να αναπτύσσεται μετά το 1900. Η συμβολή της επιστήμης του Μάρκετινγκ ενίσχυσε τη σημασία της, ιδιαίτερα κατά τη διάρκεια του Β’ Παγκοσμίου Πολέμου. Οι αυξημένες ανάγκες για συνεχή τροφοδοσία σε τρόφιμα, πυρομαχικά και άλλες προμήθειες, καθώς και η μαζική μετακίνηση στρατευμάτων, κατέστησαν την εφοδιαστική αλυσίδα κρίσιμης σημασίας. Η εξέλιξη της τεχνολογίας και η ανάπτυξη των υπολογιστικών συστημάτων επέτρεψαν τη δημιουργία νέων μοντέλων διαχείρισης, τα οποία βοήθησαν στην επιβεβαίωση ή την απόρριψη των ήδη υπαρχόντων προσεγγίσεων.

Η Διαχείριση Εφοδιαστικής Αλυσίδας (Supply Chain Management - SCM) αφορά τον σχεδιασμό, την οργάνωση και τον συντονισμό όλων των δραστηριοτήτων που σχετίζονται με τη ροή υλικών, πληροφοριών και υπηρεσιών από τους προμηθευτές πρώτων υλών, μέσω των εργοστασίων και των αποθηκών, έως τους τελικούς πελάτες.

Το SCM αποτελεί έναν σύγχρονο και δυναμικά εξελισσόμενο τομέα με σημαντική επίδραση στην αποδοτικότητα των επιχειρήσεων και τη διασφάλιση ποιοτικών διαδικασιών, σε ένα ανταγωνιστικό επιχειρηματικό περιβάλλον. Η ευρεία εφαρμογή του οφείλεται στα οφέλη που προσφέρει, όπως η μείωση του κόστους μέσω της αποτελεσματικότερης διαχείρισης των αποθεμάτων και ο βελτιωμένος συντονισμός των

διαδικασιών μεταξύ προμηθευτών και διανομέων. Με την ολοκληρωμένη εφαρμογή του, εξασφαλίζεται ότι το προϊόν φτάνει στον πελάτη τη σωστή στιγμή, στη σωστή ποσότητα, ποιότητα και τιμή, ενώ παράλληλα περιορίζονται παράγοντες που αυξάνουν το κόστος.

Στην Ελλάδα, η ενημέρωση, ο σχεδιασμός και η εφαρμογή του SCM βρίσκονται ακόμα σε αρχικό στάδιο. Σύμφωνα με πρόσφατα δεδομένα, για πολλές επιχειρήσεις παραμένει ένας «άγνωστος» τομέας. Παρόλα αυτά, η ευελιξία, η συνοχή και η πολυσυλλεκτικότητα των συστημάτων SCM συνιστούν βασικά χαρακτηριστικά που προσδίδουν σημαντική προστιθέμενη αξία στις επιχειρήσεις που τα υιοθετούν.

2. «ΠΡΑΣΙΝΟ» ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ (ΠΠΔΟ-GVRP)

Στο ευρύτερο πλαίσιο των συστημάτων μεταφορών, το πρόβλημα δρομολόγησης οχημάτων (ΠΔΟ) αναγνωρίζεται ως ένα από τα πιο σημαντικά ζητήματα στο επίπεδο επιχειρησιακών αποφάσεων. Αποτελεί μια εξειδικευμένη κατηγορία συνδυαστικού προβλήματος βελτιστοποίησης στο πλαίσιο της επιχειρησιακής έρευνας. Παραδοσιακά, το ΠΔΟ αφορά την εξυπηρέτηση πελατών μέσω ενός ή περισσότερων στόλων οχημάτων, με στόχο την ελαχιστοποίηση του λειτουργικού κόστους, λαμβάνοντας υπόψη συγκεκριμένους περιορισμούς. Σε αυτό το πλαίσιο, ένας στόλος ξεκινά από ένα κέντρο διανομής (depot), παραδίδει προϊόντα στους πελάτες εντός ενός προκαθορισμένου αστικού δικτύου και επιστρέφει στο αρχικό σημείο. Από την πρώτη διατύπωση του προβλήματος από τους Dantzig και Ramser (1959), η επιστημονική βιβλιογραφία έχει προτείνει πολλές παραλλαγές του.

Η αυξανόμενη ερευνητική δραστηριότητα που επικεντρώνεται στη μείωση των εκπομπών CO□ από τις οδικές μεταφορές έχει γνωρίσει σημαντική άνοδο από το 2010, γεγονός που υπογραμμίζει τη σημασία της μείωσης των εκπομπών αερίων του θερμοκηπίου (greenhouse gas-GHG) στον τομέα των μεταφορών.

Η «Πράσινη Εφοδιαστική» (Green Logistics) έχει κερδίσει σημαντική προσοχή τα τελευταία χρόνια, τόσο από κυβερνητικούς φορείς όσο και από επιχειρήσεις. Η αυξανόμενη σημασία της προκύπτει από το γεγονός ότι οι τρέχουσες στρατηγικές παραγωγής και διανομής δεν εξασφαλίζουν μακροπρόθεσμη βιωσιμότητα. Για τον λόγο αυτό, ο σχεδιασμός πολιτικών εφοδιαστικής δεν επικεντρώνεται πλέον αποκλειστικά σε οικονομικά κριτήρια, αλλά λαμβάνει υπόψη και τις περιβαλλοντικές, οικολογικές και κοινωνικές επιπτώσεις.

Η υιοθέτηση μιας περιβαλλοντικά υπεύθυνης προσέγγισης στην εφοδιαστική απαιτεί την αναδιάρθρωση των μεταφορικών συστημάτων και τη μετάβαση σε ένα βιώσιμο δίκτυο διανομής, με μειωμένες αρνητικές συνέπειες στο περιβάλλον και την οικολογία. Αυτό είναι ιδιαίτερα κρίσιμο, καθώς ο τομέας των μεταφορών αποτελεί βασικό στοιχείο της εφοδιαστικής αλυσίδας. Η «Πράσινη Μεταφορά» (Green Transportation) περιλαμβάνει ένα ευρύ φάσμα ζητημάτων, όπως η προώθηση εναλλακτικών

καυσίμων, η ανάπτυξη ηλεκτρικών οχημάτων επόμενης γενιάς, τα έξυπνα και φιλικά προς το περιβάλλον συστήματα μεταφορών, καθώς και άλλες βιώσιμες υποδομές. Η αποτελεσματική χρήση των οχημάτων και η βέλτιστη δρομολόγησή τους αποτελούν καθοριστικούς παράγοντες για την επίτευξη βιώσιμων μεταφορικών πρακτικών.

Σε αυτό το πλαίσιο, η ανάπτυξη ενός «πράσινου» δικτύου διανομής μέσω προηγμένων μοντέλων δρομολόγησης οχημάτων αποτελεί μία από τις σημαντικότερες προκλήσεις στον τομέα.

2.1 ΠΔΟ με περιορισμό χωρητικότητας και χρόνους εξυπηρέτησης - Capacitated VRP

Στο πρόβλημα δρομολόγησης οχημάτων (ΠΔΟ) με περιορισμό στη χωρητικότητα και τους χρόνους εξυπηρέτησης δεν μπορεί ένα όχημα να επισκεφθεί όλους τους πελάτες οπότε αναζητούνται πολλές κυκλικές διαδρομές με αφετηρία και τερματισμό την αποθήκη ώστε σε κάθε μία από αυτές να εξυπηρετείται ένα υποσύνολο των πελατών και συνολικά κάθε πελάτης να εξυπηρετείται ακριβώς μία φορά. Στόχος του προβλήματος είναι η ελαχιστοποίηση του κόστους. Ως κόστος λαμβάνεται ο συνολικός χρόνος που απαιτείται για την εξυπηρέτηση όλων των πελατών.

Οι περιορισμοί του προβλήματος είναι:

1. Κάθε πελάτης εξυπηρετείται ακριβώς μία φορά από ακριβώς ένα όχημα.
2. Όλες οι διαδρομές των οχημάτων ξεκινούν και καταλήγουν στην αποθήκη.
3. Για κάθε διαδρομή η συνολική ζήτηση που εξυπηρετείται δεν πρέπει να ξεπερνά την χωρητικότητα του οχήματος.
4. Ο συνολικός χρόνος παραμονής κάθε οχήματος σε κάθε διαδρομή, συμπεριλαμβανομένου και του χρόνου εξυπηρέτησης, δεν πρέπει να ξεπερνά ένα ανώτατο όριο.

Οι παράμετροι του προβλήματος είναι:

1. Χωρητικότητα οχήματος: η ποσότητα που μπορεί να μεταφέρει κάθε όχημα.
2. Ζήτηση: αντιπροσωπεύει την ποσότητα αγαθών που πρέπει να αποδοθεί σε κάθε πελάτη.
3. Πίνακας αποστάσεων: ένας πίνακας διαστάσεων $n*n$, όπου n ο αριθμός των πελατών και η αποθήκη, που περιλαμβάνει τις αποστάσεις μεταξύ όλων των κόμβων.
4. Χρόνος εξυπηρέτησης: ο χρόνος παραμονής σε κάθε πελάτη.
5. Μέγιστος χρόνος παραμονής του οχήματος στη διαδρομή: το ανώτατο χρονικό όριο για κάθε διαδρομή, δηλαδή το άθροισμα των χρόνων μετάβασης από την αποθήκη στους πελάτες και πάλι πίσω συν τον χρόνο εξυπηρέτησης δεν πρέπει να υπερβαίνει αυτό το όριο.

3. ΑΛΓΟΡΙΘΜΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

3.1 ΑΛΓΟΡΙΘΜΟΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΑΠΟΙΚΙΑΣ ΜΥΡΜΗΓΚΙΩΝ

Ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών είναι ένας μεθευρετικός αλγόριθμος εμπνευσμένος από τη συμπεριφορά των μυρμηγκιών στη διαδικασία εύρεσης τροφής. Χρησιμοποιείται για την εύρεση λύσεων σε προβλήματα με τη χρήση γράφου, όπως για παράδειγμα το πρόβλημα του πλανόδιου πωλητή. Τα μυρμήγκια, προκειμένου να εξασφαλίσουν την εύρεση της συντομότερης διαδρομής από την φωλιά στην τροφή και πάλι πίσω έχουν αναπτύξει μια τεχνική σύμφωνα με την οποία ξεκινούν να αναζητούν τροφή με τυχαίο τρόπο και καθώς κινούνται αφήνουν πίσω τους μια ποσότητα φερομόνης ώστε να σημαδέψουν τη διαδρομή που ακολούθησαν. Η ποσότητας της φερομόνης που αφήνουν σε κάθε διαδρομή εξαρτάται από την απόσταση, την ποιότητα και την ποσότητα της τροφής που βρέθηκε. Κάθε επόμενο μυρμήγκι που θα φύγει από τη φωλιά είναι πιθανό να ακολουθήσει αυτή τη διαδρομή και να αφήσει και αυτό ποσότητα φερομόνης. Καθώς η ποσότητα φερομόνης στη διαδρομή αυξάνεται όλο και περισσότερα μυρμήγκια επιλέγουν να την ακολουθήσουν. Συγχρόνως, όμως, παρατηρείται και εξάτμιση της φερομόνης με την πάροδο του χρόνου κυρίως σε διαδρομές που δεν επιλέγονται από πολλά μυρμήγκια.

Εν τέλει, όλα τα μυρμήγκια ακολουθούν την ίδια διαδρομή που είναι η βέλτιστη ή σχεδόν βέλτιστη και στις υπόλοιπες διαδρομές η φερομόνη εξατμίζεται πλήρως.

3.1.1 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

Η μοντελοποίηση του προβλήματος γίνεται με στόχο την εύρεση της διαδρομής που ελαχιστοποιεί το κόστος. Κάθε μυρμήγκι αντιπροσωπεύει μια λύση. Ο αλγόριθμος επαναλαμβάνεται για συγκεκριμένο αριθμό επαναλήψεων, που ορίζεται κάθε φορά από το χρήστη ανάλογα με τις ανάγκες του, όπου σε κάθε επανάληψη κάθε μυρμήγκι ξεκινά την κατασκευή μιας λύσης βάσει της εμπειρίας που έχει αποκτηθεί από τις λύσεις που δημιούργησαν τα προηγούμενα μυρμήγκια. Στο τέλος των επαναλήψεων, αν όχι όλα, η πλειοψηφία των μυρμηγκιών ακολουθούν αυτή τη λύση που θεωρείται η βέλτιστη. Στα ΠΔΟ η φερομόνη τ_{ij} σχετίζεται με τα τόξα, συνεπώς συμβολίζει την επιθυμία μετάβασης αμέσως μετά από τον πελάτη i στον πελάτη j. Η ευρετική πληροφορία (heuristic value) είναι:

$$n_{ij} = \frac{1}{c_{ij}}$$

όπου c_{ij} είναι η απόσταση από το τόξο i στο τόξο j, δηλαδή η επιθυμία να μεταβεί από τον ένα πελάτη στον άλλο είναι αντιστρόφως ανάλογη της μεταξύ τους απόστασης.

Ένας τρόπος υπολογισμού της αρχικής φερομόνης είναι:

$$\tau_{ij} = \frac{m}{Cost}$$

όπου m είναι το πλήθος των μυρμηγκιών και Cost το κόστος μιας τυχαίας αρχικής λύσης που κατασκευάστηκε με έναν απλούστερο αλγόριθμο, για παράδειγμα τον αλγόριθμο του πλησιέστερου γείτονα. Σε κάθε βήμα της διαδικασίας κατασκευής της λύσης κάθε μυρμήγκι επιλέγει σε ποιον πελάτη θα πάει βάσει της πιθανότητας:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[n_{ij}]^{\beta}}{\sum_{l=1}^{M}[\tau_{il}]^{\alpha}[n_{il}]^{\beta}}$$

όπου M το σύνολο των πελατών και α, β παράμετροι που καθορίζουν αν επηρεάζει την επιλογή περισσότερο η φερομόνη ή η ευρετική πληροφορία κάθε τόξου.

Όταν όλα τα μυρμήγκια έχουν κατασκευάσει τις αρχικές διαδρομές ενημερώνεται η φερομόνη σε κάθε τόξο, δηλαδή προστίθεται μια ποσότητα φερομόνης στα τόξα που χρησιμοποιήθηκαν από τα μυρμήγκια ενώ παράλληλα αφαιρείται από όλα τα τόξα η ίδια ποσότητα φερομόνης λόγω εξάτμισης. Αρχικά αφαιρείται ποσότητα φερμόνης από όλα τα τόξα:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$$

όπου 0 < ρ < 1 είναι ο συντελεστής εξάτμισης και χρησιμοποιείται για να εξαλείψει κακές αποφάσεις των προηγούμενων επαναλήψεων. Συνεπώς, αν ένα τόξο δεν επιλέγεται από τα μυρμήγκια η ποσότητα φερομόνης μειώνεται με την πάροδο των επαναλήψεων και τείνει στο μηδέν. Στη συνέχεια, αυξάνεται η φερομόνη των τόξων που επιλέγονται από τα μυρμήγκια:

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

όπου $\Delta\tau_{ii}^{k}$ είναι η ποσότητα φερομόνης που αφήνει κάθε μυρμήγκι k στα τόξα από τα οποία πέρασε και δίνεται απο τον τύπο:

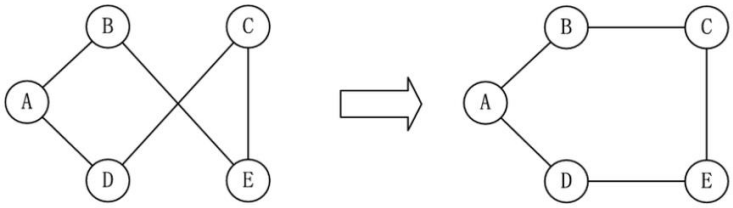
$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C^{k}} & \text{αν επιλεγεί το τόξο από το k} \\ 0 & \text{αλλιώς} \end{cases}$$

όπου C^k το κόστος της διαδρομής που έχει δημιουργήσει το k μυρμήγκι. Όσο καλύτερη είναι η διαδρομή τόσο μεγαλύτερη η ποσότητα φερομόνης που αφήνει το μυρμήγκι στα τόξα άρα μεγαλώνει και η πιθανότητα επιλογής σε επόμενη επανάληψη.

3.2 ΤΕΛΕΣΤΕΣ ΤΟΠΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ

3.2.1 Μέθοδος 2-opt

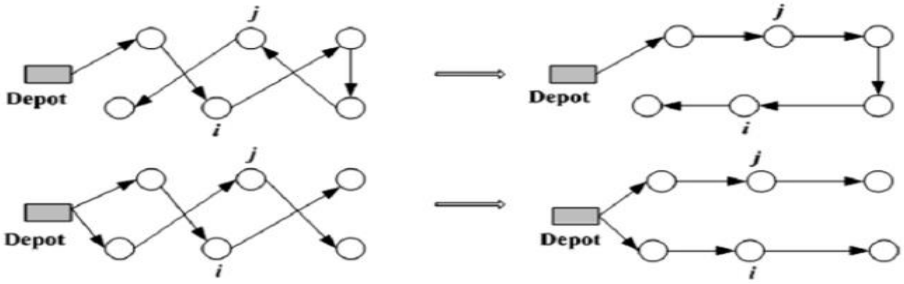
Η μέθοδος 2-opt αποτελεί διαδικασία τοπικής αναζήτησης, δηλαδή κάνει μικρές αλλαγές σε μια υπάρχουσα λύση ώστε να τη βελτιώσει. Παίρνει ως είσοδο μια διαδρομή και διαγράφει 2 ακμές και συνδέει τους κόμβους με διαφορετικό τρόπο με στόχο τη δημιουργία νέας διαδρομής με στόχο τη βελτίωση της αρχικής λύσης.



Ένα παράδειγμα του αλγορίθμου 2-opt

3.2.2 Αλγόριθμος 1-1 Exchange

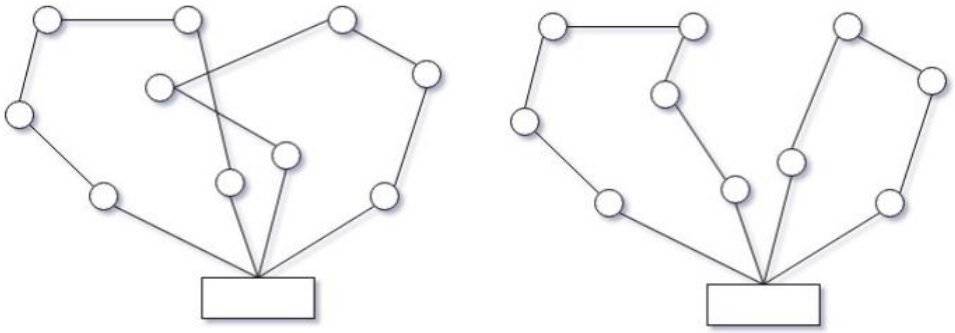
Ο αλγόριθμος 1-1 Exchange αποτελεί και αυτός αλγόριθμο τοπικής αναζήτησης και ανήκει στην κατηγορία αλγορίθμων που προσπαθούν να βελτιώσουν την ανάθεση πελατών σε οχήματα και εφαρμόζονται μεταξύ 2 ή και περισσότερων διαδρομών. Η μέθοδος 1-1 Exchange είναι η ταυτόχρονη ανταλλαγή 2 πελατών που βρίσκονται σε διαφορετικές διαδρομές με στόχο τη βελτίωση του συνολικού κόστους.



Η 1-1 exchange για μία ή πολλαπλές διαδρομές.

3.2.3 Αλγόριθμος 1-0 relocate

Ο 1-0 relocate είναι ένας αλγόριθμος τοπικής αναζήτησης του οποίου κύριος στόχος είναι η δημιουργία ομοιόμορφων διαδρομών καθώς και η διαγραφή των περιττών διαδρομών. Η λογική του είναι απλή, διαγράφεται ένας πελάτης από μια διαδρομή και επανατοποθετείται σε μια άλλη με μικρότερο κόστος.



παράδειγμα 1-0 relocate

4. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΟΣ ΚΑΙ ΚΩΔΙΚΑΣ

4.1 Ανάπτυξη αντικειμενικής συνάρτησης

Το μοντέλο στοχεύει **στην ελαχιστοποίηση του συνολικού καυσίμου** για όλες τις διαδρομές

$$\min \sum_{r \in R} \sum_{(i,j) \in r} c_{ij} \cdot FCR(veh\_load)$$

Όπου:

- R είναι το σύνολο των διαδρομών
- (i,j) είναι οι διαδοχικοί κόμβοι της κάθε διαδρομής
- c_{ij} είναι η απόσταση μεταξύ των κόμβων i και j
- FCR(veh_load) – Fuel Consumption Rate είναι η συνάρτηση του ρυθμού κατανάλωσης καυσίμου, η οποία εξαρτάται από το φορτίο του οχήματος τη τρέχουσα χρονική στιγμή
- veh_load: Το αρχικό συνολικό φορτίο του οχήματος κάθε διαδρομής το οποίο μειώνεται με τη πάροδο του χρόνου δηλαδή την εξυπηρέτηση πελατών – κόμβων

Περιορισμοί:

1. Χωρητικότητα
   Κάθε όχημα δεν πρέπει να ξεπερνά το **μέγιστο επιτρεπτό φορτίο**:
   $$\sum_{i \in r} d_i \leq veh\_capacity, \quad \forall r \in R$$
   Όπου d_i είναι η ζήτηση του κόμβου i.
2. Περιορισμός χρόνου
   Ο συνολικός χρόνος που απαιτείται για μια διαδρομή δεν πρέπει να ξεπερνά το **ανώτατο όριο**:
   $$\sum_{(i,j) \in r} t_{ij} + \sum_{i \in r} service\_time_i \leq time\_limit, \quad \forall r \in R$$
   Όπου t_{ij} είναι ο χρόνος ταξιδιού μεταξύ δύο κόμβων , service_time είναι ο χρόνος εξυπηρέτησης και time_limit το ανώτατο χρονικό όριο.
3. Κάθε πελάτης πρέπει να εξυπηρετείται **ακριβώς μία φορά**:
   $$\sum_{r \in R} x_{i,r} = 1, \quad \forall i \in N, i \neq 0$$
   δηλαδή κάθε κόμβος (εκτός της αποθήκης) πρέπει να έχει **μία είσοδο** και **μία έξοδο** στη διαδρομή του οχήματος.
4. Όλες οι διαδρομές ξεκινούν και τελειώνουν στην αποθήκη:
   $$\sum_{j \in N} x_{0j} = 1, \quad \sum_{i \in N} x_{i0} = 1$$

όπου:

- χ_{0j} είναι 1 αν το όχημα φεύγει από την αποθήκη προς τον κόμβο j, αλλιώς είναι 0.
- χ_{i0} είναι 1 αν το όχημα επιστρέφει στην αποθήκη από τον κόμβο i, αλλιώς είναι 0.

4.2 Ανάπτυξη αλγορίθμου

Για ευκολία στην κατανόηση και στην επίλυση τυχών προβλημάτων στον κώδικα, ο αλγόριθμος διαμοιράστηκε στη main συνάρτηση "**CVRP ACO.m**" και σε επιμέρους συναρτήσεις - functions - που εκτελούν τον αλγόριθμο ACO, "**ACO_FC.m**", καθώς και τους αλγορίθμους τοπικής αναζήτησης 2-opt, "**twoOpt.m**", 1-1 exchange, "**exchange.m**" και 1-0 relocate,"**oneZero.m**" . Επιπλέον, οι συναρτήσεις "**cost_calculator.m**" , "**graphPlotter.m**" και "**distance_calculator.m**" χρησιμοποιούνται για τον υπολογισμό του κόστους κάθε διαδρομής, την δημιουργία του γραφήματος και υπολογισμό συνολικής διαδρομής αντίστοιχα.

4.2.1 CVRP ACO.m

Φορτώνει δεδομένα για συγκεκριμένα προβλήματα από ένα αρχείο Excel-παραδείγματα που ονομάζουμε «parN.xlsx». Η πρώτη τιμή στο αρχείο δεδομένων αντιπροσωπεύει τον αριθμό των κόμβων, ενώ οι επόμενες τιμές καθορίζουν βασικούς περιορισμούς όπως η χωρητικότητα του οχήματος, η μέγιστη επιτρεπόμενη διάρκεια διαδρομής και ο χρόνος εξυπηρέτησης σε κάθε κόμβο. Εξάγει τις συντεταγμένες κάθε κόμβου, καθώς και τη ζήτηση που σχετίζεται με κάθε πελάτη.

```
1 data = xlsread('data\par7.xlsx');
2
3 N = data(1,1);                                    % Number of nodes
4 veh_capacity = data(3,1);                         % Vehicle capacity
5 time_limit = data(4,1);                           % Maximum vehicle on-route time
6 service_time = data(5,1);                         % Service time
7 coords = data(6:N+5,[2 3]);                       % Node coordinates
8 demand = data(N+6:end, 2);                        % Node demand
```

Χρησιμοποιώντας αυτές τις συντεταγμένες, υπολογίζεται ο Ευκλείδειος πίνακας απόστασης για τον προσδιορισμό των αποστάσεων κατά ζεύγη μεταξύ όλων των κόμβων. Για να ληφθούν υπόψη οι χρόνοι εξυπηρέτησης, αυτές οι τιμές

προστίθενται στον πίνακα απόστασης, με ειδική διόρθωση που εφαρμόζεται για να διασφαλιστεί ότι η ίδια η αποθήκη δεν έχει χρόνο εξυπηρέτησης.

```
dist_matrix = squareform(pdist(coords));                % N by N matrix - Euclidean distance between each Node
dist_matrix = dist_matrix + service_time;               % Service Time Addition
dist_matrix(:,1) = dist_matrix(:,1) - service_time;     % The depot does not have a service time
dist_matrix = dist_matrix - diag(diag(dist_matrix));
```

Στη συνέχεια, ορίζονται οι παράμετροι βάρους του οχήματος. Κάνουμε μια υπόθεση σχετικά με τον συντελεστή βάρους με βάση ένα όχημα αναφοράς, το Mercedes-Benz Atego 1523. Το απόβαρο και το ωφέλιμο φορτίο του οχήματος υπολογίζονται με βάση τη μέγιστη χωρητικότητά του, με το απόβαρο να θεωρείται ότι είναι το μισό του μέγιστου φορτίου πολλαπλασιαζόμενο με έναν συντελεστή βάρους.

```
% % Example vehicle: Mercedes-Benz Atego 1523
% GVW = 15000                                % Gross Vehicle Weight
% true_veh_curb_weight = 5000; % Approx
% true_veh_payload    = 10000; % Approx

weight_coef = 10000 / 160       % Assumption

veh_curb_weight = (veh_capacity * 0.5) * weight_coef
veh_payload     = veh_capacity * weight_coef
```

Η κατανάλωση καυσίμου μοντελοποιείται χρησιμοποιώντας μια εμπειρική συνάρτηση που προέρχεται από έρευνα των Yiyong Xiao et al. (2012) . Ο ρυθμός κατανάλωσης καυσίμου (FCR) ορίζεται ως συνάρτηση του βάρους του οχήματος, όπου μια γραμμική σχέση καθορίζει την κατανάλωση καυσίμου ανά χιλιόμετρο. Χρησιμοποιώντας αυτή τη λειτουργία, υπολογίζονται οι ρυθμοί κατανάλωσης καυσίμου για ένα άδειο και πλήρως φορτωμένο όχημα. Στη συνέχεια, ορίζεται η συνάρτηση FCR_fun για τον καθορισμό του ρυθμού κατανάλωσης καυσίμου δυναμικά με βάση το φορτίο του οχήματος σε κάθε δεδομένη στιγμή.

```
% From Yiyong Xiao et al.
L_per_km = @(veh_weight) 0.0000793 * veh_weight -
    0.026;

FCR_no_load  = L_per_km(veh_curb_weight);% no-load
    Fuel Consumption Rate
FCR_max_load = L_per_km(veh_curb_weight + veh_payload)
    ;% max-load Fuel Consumption Rate

 % Function for Fuel Consumption Rate given a load
FCR_fun = @(load_) FCR_no_load + ((FCR_max_load -
    FCR_no_load)/veh_payload)*load_;
```

Στη συνέχεια ορίζουμε βασικές παραμέτρους για τον αλγόριθμο Ant Colony Optimization (ACO). Αυτά περιλαμβάνουν το άλφα, το οποίο καθορίζει την επίδραση της έντασης της φερομόνης, το βήτα, την επίδραση της ευρετικής πληροφορίας και το rho, τον ρυθμό εξάτμισης της φερομόνης. Καθορίζεται επίσης ο αριθμός των μυρμηγκιών ανά διαδρομή και ο συνολικός αριθμός επαναλήψεων.

```
alpha = 1;
beta = 1;
rho = 0.4;
batch_size = 20;
iters = 10;
```

Με αυτές τις παραμέτρους καθορισμένες, καλεί τη συνάρτηση ACO_FC για να λύσει το πρόβλημα δρομολόγησης χρησιμοποιώντας τον δεδομένο πίνακα απόστασης, τους περιορισμούς ζήτησης και τη συνάρτηση κατανάλωσης καυσίμου. Στη συνέχεια, ο αλγόριθμος δημιουργεί ένα βελτιστοποιημένο σύνολο διαδρομών. Τέλος, απεικονίζει την καλύτερη λύση χρησιμοποιώντας μια συνάρτηση σχεδίασης που ονομάζεται graphPlotter. Η έξοδος εμφανίζεται σε ένα σχήμα που ονομάζεται "Ant Colony Optimization Algorithm", που δείχνει τις βέλτιστες διαδρομές που προέκυψαν μέσω της διαδικασίας ACO.

```
[routes] = ACO_FC(dist_matrix, demand, veh_capacity,
    time_limit, alpha, beta, rho, batch_size, iters,
    service_time, FCR_fun);

figure('Name', 'Ant Colony Optimization Algorithm');
graphPlotter(routes, coords)
```

4.2.2 ACO_FC.m

Η συνάρτηση ACO_FC εφαρμόζει τον αλγόριθμο Ant Colony Optimization (ACO) για την επίλυση ενός προβλήματος δρομολόγησης οχήματος (CVRP) με κριτήριο την κατανάλωση καυσίμου. Ο αλγόριθμος επιδιώκει να ελαχιστοποιήσει το συνολικό κόστος των διαδρομών λαμβάνοντας υπόψη την κατανάλωση καυσίμου του οχήματος, την απόσταση ταξιδιού και τους περιορισμούς φορτίου.

Ξεκινά με την ανάγνωση των δεδομένων εισόδου, συμπεριλαμβανομένου του πίνακα αποστάσεων, τη ζήτηση των πελατών, της χωρητικότητας του οχήματος και των χρονικών ορίων για τις διαδρομές. Ο αριθμός των κόμβων (N) εξάγεται από τον πίνακα αποστάσεων και υπολογίζεται μια αρχική εκτίμηση του απαιτούμενου αριθμού διαδρομών του οχήματος. Αυτή η εκτίμηση βασίζεται στη συνολική ζήτηση διαιρούμενη με τη χωρητικότητα του οχήματος, με ένα πρόσθετο buffer για να ληφθούν υπόψη οι ανεπάρκειες στην κατανομή διαδρομής.

```
function [routes] = ACO_FC(dist_matrix, demand,
    veh_capacity, time_limit, alpha, beta, rho,
    batch_size, iters, service_time, FCR_fun)

N = size(dist_matrix,1);

estimatedRouteNo = round((sum(demand)/veh_capacity)) +
     round(0.2*N);
```

Για να καθοδηγήσει τα μυρμήγκια στην κατασκευή αποτελεσματικών διαδρομών, δημιουργείται ένας ευρετικός πίνακας, όπου κάθε καταχώρηση αντιπροσωπεύει το αντίστροφο της απόστασης μεταξύ δύο κόμβων. Αυτές οι ευρετικές πληροφορίες βοηθούν στην ιεράρχηση συντομότερων διαδρομών. Τυχόν άπειρες τιμές αντικαθίστανται με μηδέν για την αποφυγή υπολογιστικών σφαλμάτων. Ο πίνακας φερομόνης, ο οποίος διαδραματίζει κρίσιμο ρόλο στην ενίσχυση των καλών λύσεων σε επαναλήψεις, αρχικοποιείται με μια μικρή τιμή (0,001), διασφαλίζοντας ότι όλες οι διαδρομές αρχικά εξετάζονται εξίσου πριν ξεκινήσει η διαδικασία των επαναλήψεων.

```
heuristic_table = power(dist_matrix, -1);
heuristic_table(heuristic_table == inf) = 0;
heuristic_table_const = heuristic_table;

global_best_cost = inf;

tau = ones(N, N)*0.001;      % initial pheromone amount
tau = tau - diag(diag(tau));
```

Ο πυρήνας της συνάρτησης αποτελείται από έναν επαναληπτικό βρόχο που εκτελείται για έναν προκαθορισμένο αριθμό επαναλήψεων (iters). Σε κάθε επανάληψη, πολλαπλά μυρμήγκια (batch_size) εξερευνούν πιθανές λύσεις κατασκευάζοντας

διαδρομές βασισμένες σε επίπεδα φερομόνης και ευρετικές πληροφορίες. Κάθε μυρμήγκι ξεκινά από την αποθήκη και προοδευτικά επιλέγει κόμβους πελατών, σχηματίζοντας μια εφικτή διαδρομή. Η διαδικασία επιλογής ακολουθεί έναν κανόνα πιθανολογικής απόφασης, όπου ένας κόμβος επιλέγεται με βάση την ένταση της φερομόνης (τ) και την ευρετική τιμή (η), σταθμισμένη με τις παραμέτρους άλφα και βήτα.

```
for i=1:iters
    batch_costs = zeros(batch_size, 1);
    batch_adjacency_matrices = cell(batch_size, 1);
    batch_routes = cell(batch_size, 1);

    for ant = 1:batch_size
        start_node = 1;
        route_idx = 1;

        batch_adjacency_matrices{ant} = zeros(N, N);
        batch_routes{ant} = cell(1, estimatedRouteNo);
        batch_routes{ant}{1} = 1;

        heuristic_table = heuristic_table_const;
        heuristic_table(:, 1) = 0;

        ant_tour_demand = zeros(estimatedRouteNo, 1);
        ant_tour_distance = zeros(estimatedRouteNo, 1)
            ; % <<< doesnt have to be array

        node_counter = 1;
        while node_counter < N
            % calculate probability of choosing a
                destination
            prob = power(tau(start_node, :), alpha).*
                power(heuristic_table(start_node, :),
                beta);
            temp = randsample(prob, 1, true, prob);
            [~, dest_node] = find(prob == temp, 1);
```

Καθώς κάθε μυρμήγκι χτίζει τη διαδρομή του, οι περιορισμοί που σχετίζονται με τη χωρητικότητα του οχήματος και τους χρονικούς περιορισμούς ελέγχονται συνεχώς. Εάν η προσθήκη ενός νέου κόμβου παραβιάζει αυτούς τους περιορισμούς, το μυρμήγκι επιστρέφει στην αποθήκη για να ξεκινήσει μια νέα διαδρομή. Διαφορετικά, ο επιλεγμένος κόμβος προσαρτάται στην τρέχουσα διαδρομή, λαμβάνεται υπόψη η ζήτηση του και ενημερώνεται η ευρετική του τιμή για να αποτραπεί η επανεπιλογή.

```
% check for capacity or time violations
            capacity_flag  = (ant_tour_demand(
                route_idx) + demand(dest_node)) >
                veh_capacity;
            timelimit_flag = (ant_tour_distance(
                route_idx) + dist_matrix(start_node,
                dest_node) + dist_matrix(dest_node, 1))
                 > time_limit;

            if (capacity_flag || timelimit_flag)
                batch_routes{ant}{route_idx} = [
                    batch_routes{ant}{route_idx}  1];
                ant_tour_distance(route_idx) =
                    ant_tour_distance(route_idx) +
                    dist_matrix(start_node, 1);
                start_node = 1;
                route_idx = route_idx + 1;
                batch_routes{ant}{route_idx} = 1;
            else
                ant_tour_demand(route_idx) =
                    ant_tour_demand(route_idx)  +
                    demand(dest_node);
                ant_tour_distance(route_idx) =
                    ant_tour_distance(route_idx) +
                    dist_matrix(start_node, dest_node);
                batch_adjacency_matrices{ant}(
                    start_node, dest_node) = 1;
                batch_routes{ant}{route_idx} = [
                    batch_routes{ant}{route_idx}
                    dest_node];
                heuristic_table(:, dest_node) = 0;
                start_node = dest_node;
                node_counter = node_counter + 1;
```

Μόλις όλα τα μυρμήγκια κατασκευάσουν τις διαδρομές τους, εφαρμόζονται τεχνικές βελτιστοποίησης τοπικής αναζήτησης, όπως η 1-0 relocate, οι 1-1 exchange και 2-opt για να τελειοποιηθούν οι λύσεις. Αυτά τα βήματα βελτιστοποίησης βοηθούν στην εξάλειψη των περιττών παρακάμψεων και βελτιώνουν την αποτελεσματικότητα κάθε διαδρομής.

```
batch_routes{ant}{route_idx} = [batch_routes{ant}{
    route_idx}  1];

        [batch_routes{ant}] = oneZero(batch_routes{ant
            }, ant_tour_demand, dist_matrix, demand,
            veh_capacity, time_limit, FCR_fun);
        [batch_routes{ant}] = exchange(batch_routes{
            ant}, ant_tour_demand, dist_matrix, demand,
             veh_capacity, time_limit, FCR_fun);
        [batch_routes{ant}] = twoOpt(batch_routes{ant
            }, dist_matrix, demand, FCR_fun);

        batch_costs(ant) = cost_calculator(
            batch_routes{ant}, dist_matrix, demand,
            FCR_fun);
    end
```

Στη συνέχεια, το κόστος της λύσης κάθε μυρμηγκιού υπολογίζεται χρησιμοποιώντας τη λειτουργία υπολογισμού κόστους, η οποία ενσωματώνει τις εκτιμήσεις κατανάλωσης καυσίμου. Η καλύτερη λύση στην τρέχουσα επανάληψη συγκρίνεται με την καλύτερη λύση που έχει βρεθεί μέχρι στιγμής. Εάν ανακαλυφθεί μια καλύτερη λύση, η global_best ενημερώνεται ανάλογα.

```
[batch_best_cost, best_ant_idx] = min(batch_costs);

    if batch_best_cost < global_best_cost
        routes = batch_routes{best_ant_idx};
        global_best_cost = batch_best_cost;
    end
```

Τέλος, ο πίνακας φερομόνης ενημερώνεται για να αντικατοπτρίζει την ποιότητα της καλύτερης λύσης που βρέθηκε στην τρέχουσα επανάληψη. Οι τιμές της φερομόνης μειώνονται με την πάροδο του χρόνου για να αποτρέψουν την πρόωρη σύγκλιση σε μη βέλτιστες λύσεις, ενώ ενθαρρύνουν την εξερεύνηση εναλλακτικών οδών.

```
% Update pheromone
    pheromone = 1 / batch_best_cost;
    batch_adjacency_matrices{best_ant_idx} =
        batch_adjacency_matrices{best_ant_idx} *
        pheromone;
    tau = tau*(1-rho) + batch_adjacency_matrices{
        best_ant_idx};

    fprintf('Iteration: %d -- batch score: %.2f\n' ,i,
        batch_best_cost);
```

4.2.3 twoOpt.m

Αυτό το function εφαρμόζει τον αλγόριθμο 2-Opt για τη βελτιστοποίηση ενός συνόλου διαδρομών οχημάτων ελαχιστοποιώντας το κόστος που βασίζεται στο καύσιμο. Η συνάρτηση ξεκινά ορίζοντας τις εισόδους της: το τρέχον σύνολο διαδρομών, έναν πίνακα απόστασης, ένα διάνυσμα ζήτησης και μια συνάρτηση που υπολογίζει τον ρυθμό κατανάλωσης καυσίμου με βάση το φορτίο του οχήματος. Ο στόχος είναι να βελτιωθούν οι δεδομένες διαδρομές με συστηματική εναλλαγή τμημάτων για να βρεθεί μια πιο αποτελεσματική διαδρομή.

Η συνάρτηση μετράει πρώτα τον αριθμό των έγκυρων (μη κενών) διαδρομών. Αυτό διασφαλίζει ότι στη φάση βελτιστοποίησης γίνεται επεξεργασία μόνο σημαντικών διαδρομών.

```
function [routes]=twoOpt(routes, dist_matrix, demand,
    FCR_fun)

n_routes=sum(cellfun(@(x)~isempty(x),routes));
```

Η κύρια βελτιστοποίηση πραγματοποιείται εντός ενός βρόχου while που εκτελείται εφόσον βρεθεί τουλάχιστον μία βελτίωση. Στην αρχή κάθε επανάληψης, το βελτιωμένο ορίζεται σε false, που σημαίνει ότι δεν έχουν γίνει ακόμη αλλαγές. Στη συνέχεια, η συνάρτηση καθορίζει το μήκος της τρέχουσας διαδρομής.

Στη συνέχεια, η συνάρτηση εισέρχεται σε μια δομή ένθετου βρόχου όπου επαναλαμβάνεται σε όλα τα πιθανά ζεύγη δεικτών (k, j) εντός της διαδρομής. Αυτοί οι δείκτες ορίζουν τμήματα της διαδρομής που μπορούν να αντιστραφούν. Για κάθε ζεύγος (k, j), η συνάρτηση δημιουργεί μια νέα διαδρομή όπου το τμήμα μεταξύ k+1 και j αναστρέφεται. Αυτή η εναλλακτική διαδρομή συγκρίνεται στη συνέχεια με την αρχική.

```
while improved
        improved = false;
        route_i_len = size(routes{i},2);

        for k = 1:route_i_len-2
            for j = (k+1):route_i_len-1
                new_route = routes{i};
                new_route(k+1:j) = flip(routes{i}(k+1:
                    j));
```

Για να προσδιορίσει εάν η ανταλλαγή είναι επωφελής, η συνάρτηση υπολογίζει το κόστος με βάση το καύσιμο τόσο για την αρχική όσο και για τις τροποποιημένες διαδρομές. Εάν το κόστος της τροποποιημένης διαδρομής είναι χαμηλότερο, η συνάρτηση ενημερώνει τη διαδρομή με τη βελτιωμένη έκδοση και ορίζει τη βελτιωμένη σε true.

```
curr_cost = cost_calculator(routes{i}, dist_matrix,
    demand, FCR_fun);
                new_cost  = cost_calculator(new_route,
                    dist_matrix, demand, FCR_fun);

                if new_cost < curr_cost
                    routes{i} = new_route;
                    improved = true;
```

Η συνάρτηση επαναλαμβάνει αυτή τη διαδικασία μέχρι να μην βρεθούν περαιτέρω βελτιώσεις. Αφού υποβληθούν σε επεξεργασία και βελτιστοποιηθούν όλες οι διαδρομές, η συνάρτηση επιστρέφει το τελικό σύνολο βελτιστοποιημένων διαδρομών.

4.2.4 Exchange.m

Αυτή η συνάρτηση υλοποιεί μια λειτουργία ανταλλαγής 1-1 για τη βελτιστοποίηση δρομολόγησης οχημάτων. Ο στόχος είναι να βελτιωθούν οι διαδρομές με την εναλλαγή ενός κόμβου από μια διαδρομή με έναν κόμβο από άλλη διαδρομή, υπό την προϋπόθεση ότι μειώνει το συνολικό κόστος και τηρεί περιορισμούς όπως η χωρητικότητα του οχήματος και τα χρονικά όρια.

```
function [routes] = exchange(routes, tour_demand,
    dist_matrix, demand, veh_capacity, time_limit,
    FCR_fun)
```

Η συνάρτηση ξεκινά με τον προσδιορισμό του αριθμού των διαδρομών (n_routes) που πρόκειται να υποβληθούν σε επεξεργασία. Στη συνέχεια, εισέρχεται στον κύριο βρόχο που επαναλαμβάνεται σε κάθε διαδρομή, παρακάμπτοντας τις κενές διαδρομές και εκείνες με λιγότερους από 3 κόμβους (καθώς οι ανταλλαγές απαιτούν τουλάχιστον 3 κόμβους).

```
n_routes = numel(routes);

    % 1-1 exchange
    for route_i_idx = 1:(n_routes-1)
        if isempty(routes{route_i_idx})
            continue; % Skip empty routes
        end

        route_i_len = numel(routes{route_i_idx});
        if route_i_len < 3
            continue; % Skip routes with less than 3
                elements
        end
```

Μέσα σε αυτόν τον βρόχο, η συνάρτηση εξετάζει κάθε κόμβο στην τρέχουσα διαδρομή (εξαιρουμένης της αποθήκης στην αρχή και στο τέλος) για πιθανή ανταλλαγή. Στη συνέχεια ελέγχει όλες τις άλλες διαδρομές για πιθανή ανταλλαγή κόμβων μεταξύ των δύο διαδρομών.

```
for node_i_idx = 2:(route_i_len - 1)
            exchange_done = 0;
            node_i = routes{route_i_idx}(node_i_idx);

            for route_j_idx = (route_i_idx+1):n_routes
                if exchange_done == 1
                    break % if there was an exchange,
                        change the i node
                end
                if isempty(routes{route_j_idx})
                    continue % Skip empty routes
                end

                route_j_len = numel(routes{route_j_idx
                    });
                if route_j_len < 3
                    continue % Skip routes with less
                        than 3 elements
                end

                for node_j_idx = 2:(route_j_len - 1)
                    node_j = routes{route_j_idx}(
                        node_j_idx);
```

Για κάθε πιθανό ζεύγος κόμβων, η συνάρτηση ελέγχει εάν η εναλλαγή τους υπερβαίνει τη χωρητικότητα του οχήματος. Εάν ο έλεγχος χωρητικότητας περάσει, η συνάρτηση προχωρά στην εκτέλεση της ανταλλαγής δημιουργώντας δύο νέες διαδρομές: μία για την τροποποιημένη διαδρομή i και άλλη για τη διαδρομή j.

```
% Capacity check
                    if ((tour_demand(route_i_idx) -
                        demand(node_i) + demand(node_j)
                        ) > veh_capacity) || ((
                        tour_demand(route_j_idx) -
                        demand(node_j) + demand(node_i)
                        ) > veh_capacity)
                         continue
                    end

                    % Node 1-1 exchange
                    new_route_i = routes{route_i_idx};
                    new_route_i(node_i_idx) = node_j;

                    new_route_j = routes{route_j_idx};
                    new_route_j(node_j_idx) = node_i;

                    % Calculate new distances
                    new_route_i_dist =
                        distance_calculator(new_route_i
                        , dist_matrix);
                    new_route_j_dist =
                        distance_calculator(new_route_j
                        , dist_matrix);
```

Στη συνέχεια, η συνάρτηση υπολογίζει τη συνολική απόσταση και για τις δύο νέες διαδρομές χρησιμοποιώντας το distance_calculator. Εάν κάποια από τις νέες διαδρομές υπερβεί το καθορισμένο χρονικό όριο, η ανταλλαγή απορρίπτεται και ο αλγόριθμος μετακινείται στο επόμενο ζεύγος κόμβων.

```
if (new_route_i_dist > time_limit) || (
    new_route_j_dist > time_limit)
                        continue;
                    end
```

Στη συνέχεια, η συνάρτηση υπολογίζει το συνολικό κόστος και για τις δύο διαδρομές πριν και μετά την ανταλλαγή, χρησιμοποιώντας το cost_calculator. Εάν τα νέα δρομολόγια έχουν χαμηλότερο ή ίσο κόστος από τα αρχικά, η ανταλλαγή γίνεται αποδεκτή και τα δρομολόγια ενημερώνονται ανάλογα. Η διαδικασία συνεχίζεται μέχρι να διερευνηθούν όλες οι έγκυρες ανταλλαγές.

```
new_route_i_cost = cost_calculator(new_route_i,
    dist_matrix, demand, FCR_fun);
                    new_route_j_cost = cost_calculator
                        (new_route_j, dist_matrix,
                        demand, FCR_fun);

                    route_i_cost = cost_calculator(
                        routes{route_i_idx},
                        dist_matrix, demand, FCR_fun);
                    route_j_cost = cost_calculator(
                        routes{route_j_idx},
                        dist_matrix, demand, FCR_fun);

                    if (new_route_i_cost +
                        new_route_j_cost) <= (
                        route_i_cost + route_j_cost)
                         routes{route_i_idx} =
                            new_route_i;
                         routes{route_j_idx} =
                            new_route_j;
                         exchange_done = 1;
                         break
```

Μόλις ολοκληρωθεί η επεξεργασία όλων των κόμβων, η συνάρτηση επιστρέφει το ενημερωμένο σύνολο διαδρομών.

4.2.5 oneZero.m

Αυτή η συνάρτηση υλοποιεί τον αλγόριθμο τοπικής αναζήτησης 1-0 Relocate. Ο στόχος του αλγορίθμου είναι να βελτιώσει την τρέχουσα λύση μεταφέροντας έναν μεμονωμένο κόμβο από την τρέχουσα διαδρομή του σε άλλη διαδρομή, διασφαλίζοντας ότι η λύση σέβεται τη χωρητικότητα του οχήματος και τα χρονικά όρια, ελαχιστοποιώντας παράλληλα το κόστος.

```
function [routes] =  oneZero(routes, tour_demand,
    dist_matrix, demand, veh_capacity, time_limit,
    FCR_fun)
```

Η συνάρτηση ξεκινά με τον προσδιορισμό του αριθμού των διαδρομών (n_routes) και του συνολικού αριθμού των κόμβων (N).

```
n_routes = sum(cellfun(@(x)~isempty(x),routes));
N = size(demand, 1);
```

Στη συνέχεια, σε ένα βρόχο while ο αλγόριθμος εξετάζει κάθε κόμβο ξεκινώντας από τον κόμβο 2 (καθώς ο κόμβος 1 είναι η αποθήκη). Για κάθε κόμβο i, ο αλγόριθμος προσδιορίζει τη διαδρομή (route_i_idx) όπου βρίσκεται ο κόμβος i αυτήν τη στιγμή.

```
improved = 1;
while improved == 1
    improved = 0;
    node_i = 2;
    while node_i <= N && improved == 0;

        % find Route and position of node_i
        for m = 1:n_routes
            node_i_idx = find(routes{m} == node_i);
            if node_i_idx > 0
                route_i_idx = m;
                break
            end
        end
```

Στη συνέχεια, ελέγχει εάν αυτός ο κόμβος μπορεί να μεταφερθεί σε άλλες διαδρομές χωρίς να παραβιαστεί η χωρητικότητα του οχήματος. Η λίστα canReloc περιέχει τους δείκτες των διαδρομών στις οποίες θα μπορούσε ενδεχομένως να μετακινηθεί ο κόμβος. Οι διαδρομές όπου η μετακίνηση θα παραβίαζε τη χωρητικότητα εξαιρούνται και ο αλγόριθμος παρακάμπτει τον τρέχοντα κόμβο εάν δεν υπάρχουν έγκυρες επιλογές επανατοποθέτησης.

```
canReloc = find((tour_demand + demand(node_i)) <=
    veh_capacity);
canReloc(canReloc == route_i_idx) = 0;

    if isempty(nonzeros(canReloc)) == 1
        node_i = node_i + 1;
        continue
    end
```

Στη συνέχεια, η συνάρτηση υπολογίζει το κόστος της τρέχουσας διαδρομής (route_i_cost) και το κόστος της διαδρομής εάν αφαιρέθηκε ο κόμβος i (new_route_i_cost).

```
route_i_len = numel(routes{route_i_idx});
route_i_cost = cost_calculator(routes{route_i_idx},
    dist_matrix, demand, FCR_fun);
new_route_i = [routes{route_i_idx}(1:node_i_idx-1)
    routes{route_i_idx}(node_i_idx + 1:end)];
new_route_i_cost = cost_calculator(new_route_i,
    dist_matrix, demand, FCR_fun);
```

Μετά από αυτό, για κάθε διαδρομή όπου μπορεί να μεταφερθεί ο κόμβος i (route_j_idx), ο αλγόριθμος εξετάζει όλες τις πιθανές θέσεις (pos_j_idx) εντός της διαδρομής όπου θα μπορούσε να εισαχθεί ο κόμβος i.

```
for route_j_idx = nonzeros(canReloc)'
    route_j_len = numel(routes{route_j_idx});
    route_j_cost = cost_calculator(routes{route_j_idx
        }, dist_matrix, demand, FCR_fun);

    for pos_j_idx = 1:route_j_len-1
        new_route_j = [routes{route_j_idx}(1:pos_j_idx
            ) node_i routes{route_j_idx}(pos_j_idx + 1:
            end)];
```

Για κάθε πιθανή εισαγωγή, ο αλγόριθμος εκτελεί έναν έλεγχο χρονικού ορίου για να διασφαλίσει ότι η νέα διαδρομή δεν υπερβαίνει τον μέγιστο επιτρεπόμενο χρόνο. Εάν η εισαγωγή δεν παραβιάζει το χρονικό όριο, υπολογίζεται το νέο κόστος της διαδρομής με τον κόμβο i που έχει εισαχθεί (new_route_j_cost).

```
new_route_j_dist = distance_calculator(new_route_j,
    dist_matrix);
if new_route_j_dist > time_limit
    continue
end
```

Στη συνέχεια, ο αλγόριθμος συγκρίνει το συνολικό κόστος των τροποποιημένων διαδρομών (η αρχική διαδρομή χωρίς τον κόμβο i και η τροποποιημένη διαδρομή με τον κόμβο i εισαγόμενο) με το τρέχον κόστος. Εάν το νέο συνολικό κόστος είναι χαμηλότερο, η επανατοποθέτηση γίνεται αποδεκτή και τα δρομολόγια ενημερώνονται. Αντίστοιχα ενημερώνονται και οι απαιτήσεις για τα τροποποιημένα δρομολόγια. Η διαδικασία επαναλαμβάνεται μέχρι να μην γίνουν περαιτέρω βελτιώσεις, οπότε η συνάρτηση επιστρέφει τις ενημερωμένες διαδρομές.

```
            new_route_j_cost = cost_calculator(
                new_route_j, dist_matrix, demand,
                FCR_fun);
            if (new_route_i_cost + new_route_j_cost) <
                (route_i_cost + route_j_cost)
            routes{route_i_idx} = new_route_i;
            routes{route_j_idx} = new_route_j;
            tour_demand(route_i_idx) = tour_demand(
                route_i_idx) - demand(node_i);
            tour_demand(route_j_idx) = tour_demand(
                route_j_idx) + demand(node_i);
            improved = 1;
            break
        end
    end
    if improved == 1
        break
    end
end
if improved == 1
    break
end
node_i = node_i + 1;
```

4.2.6 cost_calculator.m

Η συνάρτηση cost_calculator είναι υπεύθυνη για τον υπολογισμό του συνολικού κόστους καυσίμου μιας ή περισσότερων διαδρομών με βάση την απόσταση μεταξύ των κόμβων, τη ζήτηση σε κάθε κόμβο και τον ρυθμό κατανάλωσης καυσίμου. Αυτό το κόστος υπολογίζεται λαμβάνοντας υπόψη τόσο τη διανυθείσα απόσταση όσο και τη μεταβαλλόμενη κατανάλωση καυσίμου ανάλογα με το φορτίο του οχήματος.

```
function [total_cost, route_costs] = cost_calculator(
    routes, dist_matrix, demand, FCR_fun)
```

Αρχικά, η συνάρτηση ελέγχει εάν οι διαδρομές εισόδου είναι μια συστοιχία κελιών, που σημαίνει ότι έχει πολλαπλές διαδρομές ή απλώς μια διαδρομή. Εάν η είσοδος είναι ένας πίνακας κελιών, η συνάρτηση θα υπολογίσει το κόστος για κάθε διαδρομή ξεχωριστά και, στη συνέχεια, θα αθροίσει τα επιμέρους κόστη για να λάβει το συνολικό κόστος. Εάν η είσοδος είναι μία διαδρομή, η συνάρτηση εκτελεί τον υπολογισμό του κόστους μόνο για αυτήν τη διαδρομή.

```
if iscell(routes) == 1
    n_routes = sum(cellfun(@(x)~isempty(x),routes));
    route_costs = zeros(n_routes, 1);
```

Κατά το χειρισμό πολλαπλών διαδρομών, η συνάρτηση υπολογίζει το συνολικό φορτίο του οχήματος αθροίζοντας τη ζήτηση όλων των κόμβων (εξαιρουμένων των κόμβων έναρξης και λήξης).

```
for i = 1:n_routes
    route_len = size(routes{i}, 2);

    % Calculate initial load
    veh_load = 0;
    for node = 2:route_len-1
        veh_load = veh_load + demand(routes{i}(node));
    end
```

Στη συνέχεια, για κάθε τμήμα της διαδρομής (δηλαδή, μεταξύ διαδοχικών κόμβων), υπολογίζει την απόσταση μεταξύ των κόμβων και του καυσίμου που καταναλώνεται με βάση το τρέχον φορτίο. Η κατανάλωση καυσίμου εξαρτάται από την ποσότητα της ζήτησης που μεταφέρει το όχημα εκείνη τη στιγμή. Μετά τον υπολογισμό του κόστους καυσίμου για όλα τα τμήματα της διαδρομής, το συνολικό κόστος για τη διαδρομή αποθηκεύεται σε μια συστοιχία. Τέλος, επιστρέφεται το συνολικό κόστος σε όλες τις διαδρομές.

```
% Calculate cost
            for j = 1:route_len-1
                distance = dist_matrix(routes{i}(j),
                    routes{i}(j+1));
                FCR = FCR_fun(veh_load);      % Fuel
                    Consumption Rate along route j -> j
                    +1
                fuel_cost = distance * FCR;

                route_costs(i) = route_costs(i) +
                    fuel_cost;
                veh_load = veh_load - demand(routes{i
                    }(j+1));
            end
        end
        total_cost = sum(route_costs);
    else
        cost = 0;
        route_len = size(routes, 2);
```

Εάν η είσοδος δεν είναι πίνακας κελιών (που υποδεικνύει μόνο μία διαδρομή), η συνάρτηση απλώς υπολογίζει το κόστος για τη συγκεκριμένη διαδρομή. Τα βήματα είναι παρόμοια: υπολογίζει το φορτίο του οχήματος, υπολογίζει το κόστος καυσίμου για κάθε τμήμα της διαδρομής και επιστρέφει το συνολικό κόστος.

```
% Calculate initial load
        veh_load = 0;
        for node = 2:route_len-1
            veh_load = veh_load + demand(routes(node))
                ;
        end

        % Calculate cost
        for j =1:route_len-1
            distance = dist_matrix(routes(j), routes(j
                +1));
            FCR = FCR_fun(veh_load) ;      % Fuel
                Consumption Rate along route j -> j+1
            fuel_cost = distance * FCR;

            cost = cost + fuel_cost;
            veh_load = veh_load - demand(routes(j+1));
        end
        total_cost = cost;
```

Μετά τον υπολογισμό του κόστους καυσίμου για όλα τα τμήματα όλων των διαδρομών (ή μόνο της μεμονωμένης διαδρομής), η συνάρτηση επιστρέφει το συνολικό κόστος, το οποίο είναι το άθροισμα του κόστους του μεμονωμένου τμήματος. Εάν υπάρχουν πολλές διαδρομές, επιστρέφει επίσης μια σειρά από μεμονωμένα κόστη διαδρομής.

4.2.7 Distance_calculator.m

Η συνάρτηση distance_calculator είναι υπεύθυνη για τον υπολογισμό του συνολικού χρόνου ταξιδιού για ένα σύνολο διαδρομών με βάση τον πίνακα απόστασης που περιέχει τις αποστάσεις μεταξύ κάθε ζεύγους κόμβων.

```
function [totalTime, rtime] = distance_calculator(
    routes,dist_matrix)
```

Εάν οι διαδρομές εισόδου είναι ένας πίνακας κελιών (που σημαίνει ότι υπάρχουν πολλές διαδρομές), η συνάρτηση πρώτα μετράει πόσες μη κενές διαδρομές υπάρχουν. Στη συνέχεια, αρχικοποιεί έναν πίνακα, rtime, για να αποθηκεύσει το χρόνο ταξιδιού για κάθε διαδρομή.

```
if iscell(routes) == 1
    routesNo = sum(cellfun(@(x)~isempty(x),routes));
    rtime = zeros(routesNo,1);
```

Στη συνέχεια, η συνάρτηση επαναλαμβάνεται σε κάθε διαδρομή. Για κάθε διαδρομή, αθροίζει τις αποστάσεις μεταξύ διαδοχικών κόμβων χρησιμοποιώντας το dist_matrix, το οποίο αποθηκεύει την απόσταση μεταξύ κάθε ζεύγους κόμβων. Ο συνολικός χρόνος ταξιδιού για την τρέχουσα διαδρομή συσσωρεύεται στο αντίστοιχο στοιχείο του χρόνου r.

```
for sn = 1:routesNo
            for pnt = 1:size(routes{sn}, 2)-1
                rtime(sn) = rtime(sn) + dist_matrix(
                    routes{sn}(pnt),routes{sn}(pnt+1));
            end
        end
        totalTime = sum(rtime);
```

Μετά τον υπολογισμό των χρόνων ταξιδιού για όλες τις επιμέρους διαδρομές, ο συνολικός χρόνος για όλες τις διαδρομές σε συνδυασμό υπολογίζεται αθροίζοντας τις τιμές στη διάταξη rtime.

Εάν οι διαδρομές εισόδου δεν είναι ένας πίνακας κελιών (δηλαδή, υπάρχει μόνο μία διαδρομή), η συνάρτηση εκτελεί την ίδια διαδικασία, αλλά για μία μόνο διαδρομή. Αρχικοποιεί το rtime στο μηδέν και στη συνέχεια αθροίζει τις αποστάσεις μεταξύ διαδοχικών κόμβων σε μία διαδρομή. Στη συνέχεια, ο συνολικός χρόνος αποθηκεύεται στο totalTime και ο χρόνος μεμονωμένης διαδρομής επιστρέφεται ως rtime.

```
else
        rtime = 0;
        for pnt =1:size(routes,2)-1
            rtime = rtime + dist_matrix(routes(pnt),
                routes(pnt+1));
        end
        totalTime = rtime;
    end
```

Η συνάρτηση τελικά επιστρέφει τον συνολικό χρόνο ταξιδιού για όλες τις διαδρομές (ή μόνο τη μεμονωμένη διαδρομή) στο totalTime και επίσης επιστρέφει το χρόνο ταξιδιού για κάθε μεμονωμένη διαδρομή σε rtime.

4.2.8 GraphPlotter.m

Η συνάρτηση graphPlotter οπτικοποιεί τις διαδρομές σε ένα 2D γράφημα.

Αρχικά μετράει τον αριθμό των μη κενών διαδρομών και αρχικοποιεί τις απαραίτητες μεταβλητές. Για κάθε διαδρομή, εξάγει τους κόμβους source (s1) και target (t2) και στη συνέχεια δημιουργεί ένα κατευθυνόμενο γράφημα . Οι κόμβοι εμφανίζονται με μαύρο χρώμα.

```
function graphPlotter(routes, coords)
    weights = [];
    nodeNo = size(coords,1);
    n_routes = sum(cellfun(@(x)~isempty(x),routes));
    hold on
        for i = 1:n_routes
            s1 = routes{i}(1, 1:end-1);
            t2 = routes{i}(1, 2:end);
            G = digraph(s1, t2, weights, nodeNo);
            plot(G, 'XData', coords(:,1), 'YData',
               coords(:,2), 'NodeColor', 'k')
        end
    hold off
end
```

5. ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ

Με στόχο τα αποτελέσματα που λαμβάνονται από τις προσομοιώσεις των δεδομένων να ανταποκρίνονται όσο το δυνατόν καλύτερα στην πραγματικότητα, επιλέχθηκε ως όχημα ένα σύνηθες φορτηγό Mercedes – Benz Atego1523. Τα χαρακτηριστικά του οχήματος είναι τα ακόλουθα:

Μικτό βάρος οχήματος = 15000kg

Βάρος οχήματος χωρίς φορτίο = περ. 5000kg

Καθαρό βάρος φόρτωσης = περ. 10000kg

(Λόγω κανονικοποίησης των δεδομένων- kg & km- τα παραπάνω διαφοροποιούνται για κάθε παράδειγμα)

Όπως προαναφέρθηκε, οι παράμετροι που επηρεάζουν τον αλγόριθμο ACO είναι οι α και β που καθορίζουν την επιλογή με περισσότερη έμφαση στη φερομόνη ή στην ευρετική πληροφορία κάθε τόξου αντίστοιχα ενώ το ρ αποτελεί τον ρυθμό εξάτμισης της φερομόνης.

Για να καθοριστούν οι βέλτιστες τιμές των παραμέτρων α, β και ρ έγιναν δοκιμές για διαφορετικούς συνδυασμούς τιμών και σετ δεδομένων, τα αποτελέσματα των οποίων παρατίθενται στη συνέχεια.

Ορίζονται επίσης ο αριθμός των μυρμηγκιών (batch size) που θα δημιουργηθούν και οι επαναλήψεις (iters) που θα πραγματοποιηθούν για την εύρεση της λύσης:

batch_size = 20

iters = 10

5.1 Παράδειγμα 1

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   51
Χωρητικότητα οχήματος:    160
Χρονικό όριο διαδρομής:   999999
χρόνος εξυπηρέτησης του κόμβου:  0
Βάρος οχήματος χωρίς φορτίο = 5000kg
Καθαρό βάρος φόρτωσης  = 10000kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.1.1 α= 1 β = 1 ρ = 0.4

```
Η υλοποίηση του αλγορίθμου γίνεται για τιμές:
a= 1   b = 1  ρ = 0.4
```

Επαναλήψεις:

Iteration: 1 -- batch score: 200.11

Iteration: 2 -- batch score: 211.28

Iteration: 3 -- batch score: 204.57

Iteration: 4 -- batch score: 190.85

Iteration: 5 -- batch score: 211.27

Iteration: 6 -- batch score: 198.33

Iteration: 7 -- batch score: 194.09

Iteration: 8 -- batch score: 205.31

Iteration: 9 -- batch score: 203.67

Iteration: 10 -- batch score: 211.31

best score: 190.85

Total Tour distance: 504.27

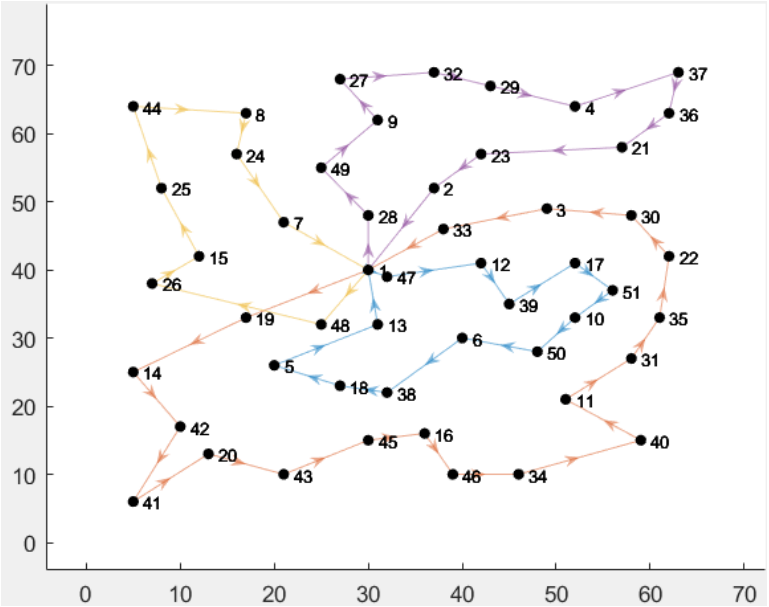
Βέλτιστη διαδρομή:

Route 1: 1 47 12 39 17 51 10 50 6 38 18 5 13 1

Route 2: 1 19 14 42 41 20 43 45 16 46 34 40 11 31 35 22 30 3 33 1

Route 3: 1 48 26 15 25 44 8 24 7 1

Route 4: 1 28 49 9 27 32 29 4 37 36 21 23 2 1



Γράφημα 1: Διαδρομές δεδομένων 5.1.1

5.1.2 α=0.2 β=10ρ=0.4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 209.11

Iteration: 2 -- batch score: 214.64

Iteration: 3 -- batch score: 210.81

Iteration: 4 -- batch score: 198.79

Iteration: 5 -- batch score: 205.31

Iteration: 6 -- batch score: 201.73

Iteration: 7 -- batch score: 212.79

Iteration: 8 -- batch score: 207.58

Iteration: 9 -- batch score: 206.43

Iteration: 10 -- batch score: 205.76

best score: 198.79

Total Tour distance: 528.19

Βέλτιστη διαδρομή:

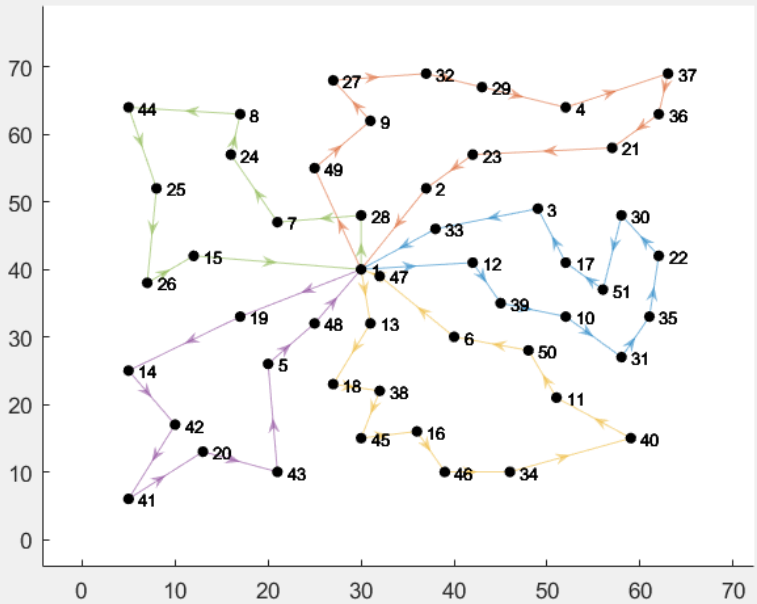
Route 1: 1 12 39 10 31 35 22 30 51 17 3 33 1

Route 2: 1 49 9 27 32 29 4 37 36 21 23 2 1

Route 3: 1 13 18 38 45 16 46 34 40 11 50 6 47 1

Route 4: 1 19 14 42 41 20 43 5 48 1

Route 5: 1 28 7 24 8 44 25 26 15 1



Γράφημα 2: Διαδρομές για δεδομένα 5.1.2

5.1.3 α=0.2 β=10 ρ=0.2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 209.11

Iteration: 2 -- batch score: 211.39

Iteration: 3 -- batch score: 198.94

Iteration: 4 -- batch score: 200.13

Iteration: 5 -- batch score: 207.92

Iteration: 6 -- batch score: 211.92

Iteration: 7 -- batch score: 200.75

Iteration: 8 -- batch score: 199.83

Iteration: 9 -- batch score: 201.97

Iteration: 10 -- batch score: 205.52

best score: 198.94

Total Tour distance: 528.69

Βέλτιση διαδρομή:

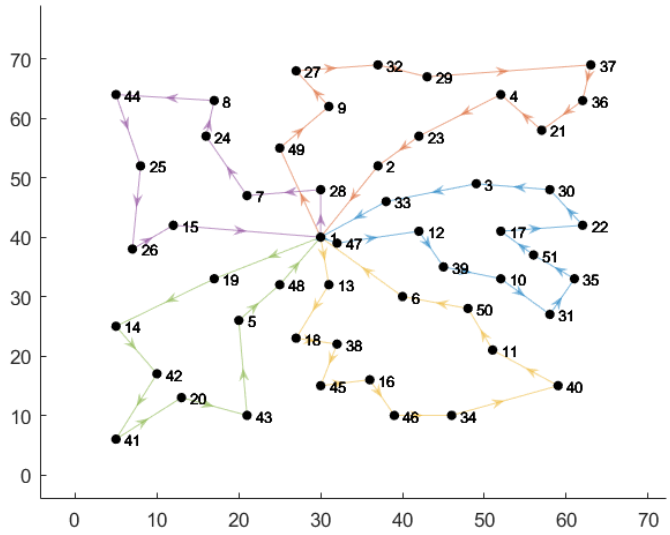
Route 1: 1 47 12 39 10 31 35 51 17 22 30 3 33 1

Route 2: 1 49 9 27 32 29 37 36 21 4 23 2 1

Route 3: 1 13 18 38 45 16 46 34 40 11 50 6 1

Route 4: 1 28 7 24 8 44 25 26 15 1

Route 5: 1 19 14 42 41 20 43 5 48 1



Γράφημα 3: Διαδρομές για δεδομένα 5.1.3

5.1.4 Σύγκριση Αποτελεσμάτων

Παρατηρήθηκε πως για α=1, β=1 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (190.85lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (504.17km) με τον ελάχιστο αριθμό διαδρομών (4). Η ρύθμιση του ρ δεν επηρέασε στην τελική λύση.

5.2 Παράδειγμα 2

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:    76
Χωρητικότητα οχήματος:     140
Χρονικό όριο διαδρομής:    999999
χρόνος εξυπηρέτησης του κόμβου:  0
Βάρος οχήματος χωρίς φορτίο = 4375kg
Καθαρό βάρος φόρτωσης  = 8750kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.2.1 α=1 β=1 ρ=0.4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 298.43

Iteration: 2 -- batch score: 294.72

Iteration: 3 -- batch score: 289.01

Iteration: 4 -- batch score: 293.20

Iteration: 5 -- batch score: 278.55

Iteration: 6 -- batch score: 281.05

Iteration: 7 -- batch score: 284.62

Iteration: 8 -- batch score: 284.29

Iteration: 9 -- batch score: 291.96

Iteration: 10 -- batch score: 298.04

best score: 278.55

Total Tour distance: 852.42

Βέλτιστη διαδρομή:

Route 1: 1 69 63 23 65 43 2 74 1

Route 2: 1 35 47 53 14 28 5 1

Route 3: 1 8 54 39 11 32 66 67 12 60 15 36 1

Route 4: 1 68 27 59 73 40 10 51 19 25 50 17 52 1

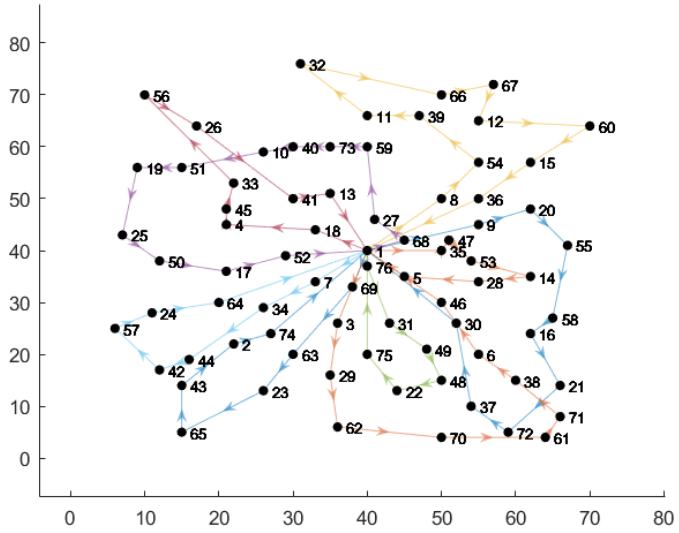
Route 5: 1 76 31 49 48 22 75 1

Route 6: 1 7 34 44 42 57 24 64 1

Route 7: 1 18 4 45 33 56 26 41 13 1

Route 8: 1 9 20 55 58 16 21 72 37 30 1

Route 9: 1 3 29 62 70 61 71 38 6 46 1



Γράφημα 4: Διαδρομές για δεδομένα 5.2.1

5.2.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 292.69

Iteration: 2 -- batch score: 291.75

Iteration: 3 -- batch score: 288.75

Iteration: 4 -- batch score: 285.39

Iteration: 5 -- batch score: 292.24

Iteration: 6 -- batch score: 287.50

Iteration: 7 -- batch score: 284.51

Iteration: 8 -- batch score: 285.29

Iteration: 9 -- batch score: 284.55

Iteration: 10 -- batch score: 285.19

best score: 284.51

Total Tour distance: 873.48

Βέλτιστη διαδρομή:

Route 1: 1 18 41 1

Route 2: 1 5 46 49 31 76 1

Route 3: 1 68 47 55 14 28 53 35 1

Route 4: 1 27 8 36 54 15 20 9 1

Route 5: 1 69 3 75 29 62 22 48 1

Route 6: 1 13 59 73 40 10 33 45 4 1

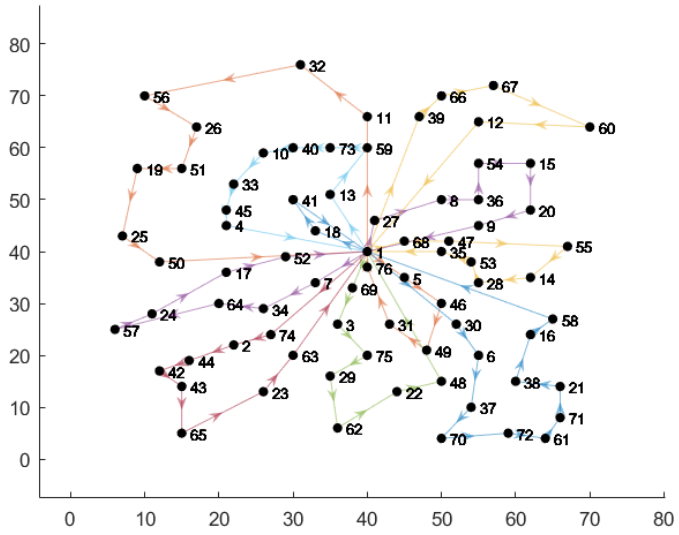
Route 7: 1 74 2 44 42 43 65 23 63 1

Route 8: 1 30 6 37 70 72 61 71 21 38 16 58 1

Route 9: 1 11 32 56 26 51 19 25 50 1

Route 10: 1 39 66 67 60 12 1

Route 11: 1 7 34 64 57 24 17 52 1



Γράφημα 5: Διαδρομές για δεδομένα 5.2.2

5.2.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 292.69

Iteration: 2 -- batch score: 291.75

Iteration: 3 -- batch score: 288.75

Iteration: 4 -- batch score: 288.52

Iteration: 5 -- batch score: 291.34

Iteration: 6 -- batch score: 291.86

Iteration: 7 -- batch score: 289.56

Iteration: 8 -- batch score: 289.20

Iteration: 9 -- batch score: 296.33

Iteration: 10 -- batch score: 286.65

best score: 286.65

Total Tour distance: 880.00

Βέλτιστη διαδρομή:

Route 1: 1 18 41 13 1

Route 2: 1 68 35 47 53 28 46 1

Route 3: 1 30 49 6 16 58 14 55 20 9 1

Route 4: 1 76 31 5 1

Route 5: 1 69 3 29 63 74 64 34 1

Route 6: 1 27 73 40 10 33 45 4 52 1

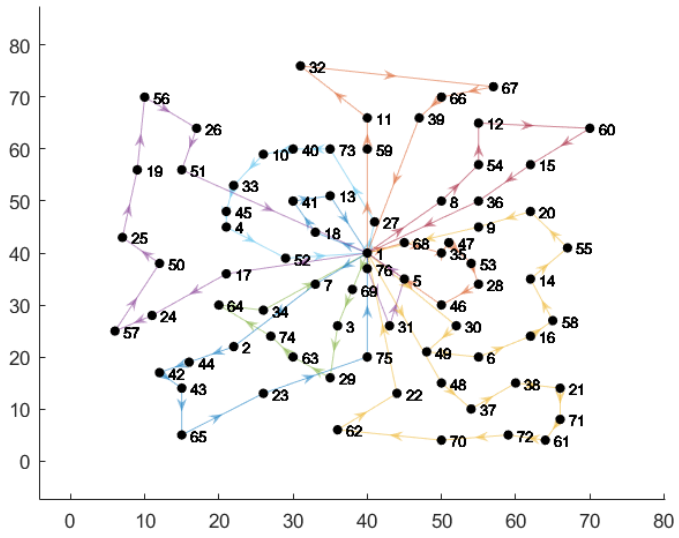
Route 7: 1 8 54 12 60 15 36 1

Route 8: 1 7 2 44 42 43 65 23 75 1

Route 9: 1 59 11 32 67 66 39 1

Route 10: 1 48 37 38 21 71 61 72 70 62 22 1

Route 11: 1 17 24 57 50 25 19 56 26 51



Γράφημα 6: Διαδρομές για δεδομένα 5.2.3

5.2.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=1, β=1 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (278.85lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (852.42km) με τον ελάχιστο αριθμό διαδρομών (9). Η ελαχιστοποίηση του ρ (εξάτμιση φερομόνης) δεν βελτίωσε τη λύση.

5.3

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   101
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   999999
χρόνος εξυπηρέτησης του κόμβου:  0
Βάρος οχήματος χωρίς φορτίο =  6250kg
Καθαρό βάρος φόρτωσης = 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.3.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 426.25

Iteration: 2 -- batch score: 434.29

Iteration: 3 -- batch score: 438.11

Iteration: 4 -- batch score: 428.53

Iteration: 5 -- batch score: 426.15

Iteration: 6 -- batch score: 425.27

Iteration: 7 -- batch score: 421.19

Iteration: 8 -- batch score: 439.14

Iteration: 9 -- batch score: 421.58

Iteration: 10 -- batch score: 429.32

best score: 421.19

Total Tour distance: 884.10

Βέλτιστη διαδρομή:

Route 1: 1 28 70 2 51 4 30 25 69 81 13 1

Route 2: 1 89 63 12 65 50 37 48 47 83 8 53 1

Route 3: 1 90 19 84 6 62 17 87 39 44 16 58 3 59 1

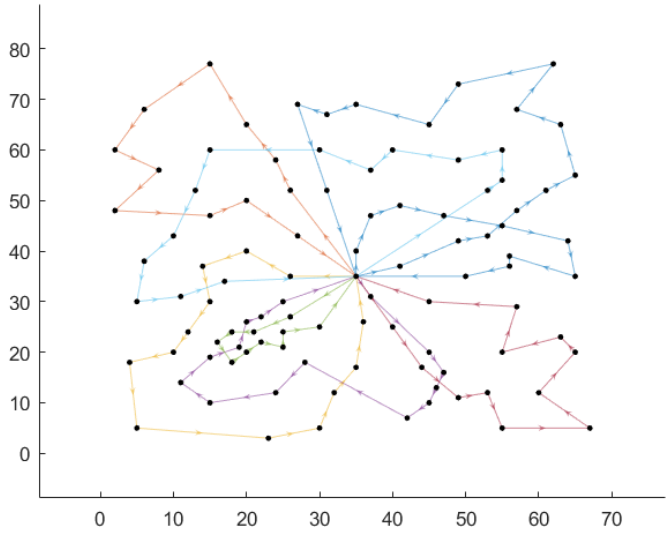
Route 4: 1 54 22 73 75 23 42 88 43 15 45 92 99 100 97 7 1

Route 5: 1 95 60 94 86 101 38 93 98 96 14 1

Route 6: 1 34 82 10 52 31 71 11 20 49 9 46 18 85 61 1

Route 7: 1 41 74 76 57 24 68 40 26 56 5 55 27 1

Route 8: 1 29 77 78 80 79 35 36 72 66 67 21 33 91 64 32 1



Γράφημα 7: Διαδρομές για δεδομένα 5.2.3

5.3.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 408.18

Iteration: 2 -- batch score: 418.88

Iteration: 3 -- batch score: 425.95

Iteration: 4 -- batch score: 423.49

Iteration: 5 -- batch score: 416.78

Iteration: 6 -- batch score: 420.04

Iteration: 7 -- batch score: 410.33

Iteration: 8 -- batch score: 422.43

Iteration: 9 -- batch score: 414.57

Iteration: 10 -- batch score: 415.54

best score: 408.18

Total Tour distance: 857.20

Βέλτιστη διαδρομή:

Route 1: 1 54 22 73 75 76 24 68 40 5 27 1

Route 2: 1 51 77 78 4 80 34 82 79 35 36 10 52 2 70 28 1

Route 3: 1 29 13 81 69 30 25 55 56 26 57 23 42 74 41 1

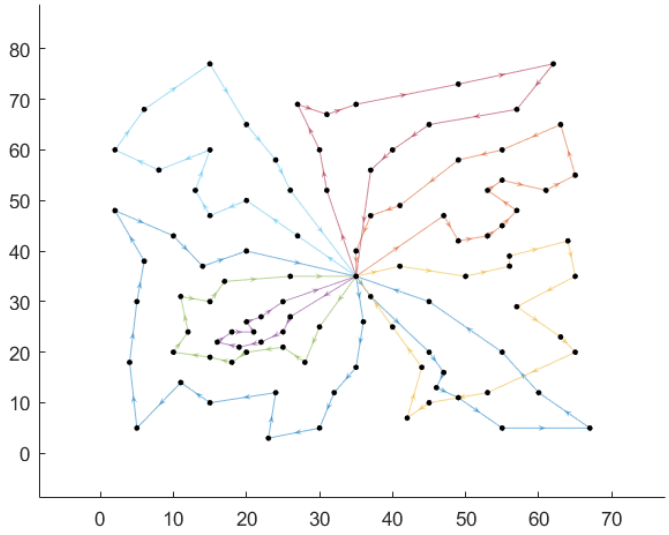
Route 4: 1 95 96 93 99 86 94 60 100 97 7 1

Route 5: 1 14 88 98 38 101 92 17 62 85 6 61 90 1

Route 6: 1 53 8 83 49 20 48 37 50 65 12 63 89 1

Route 7: 1 32 11 64 91 33 67 66 72 21 31 71 1

Route 8: 1 59 3 58 16 44 43 15 45 39 87 18 46 47 9 84 19 1



Γράφημα 8: Διαδρομές για δεδομένα 5.3.2

5.3.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 408.18

Iteration: 2 -- batch score: 418.77

Iteration: 3 -- batch score: 423.28

Iteration: 4 -- batch score: 415.12

Iteration: 5 -- batch score: 419.87

Iteration: 6 -- batch score: 421.72

Iteration: 7 -- batch score: 415.85

Iteration: 8 -- batch score: 418.60

Iteration: 9 -- batch score: 419.13

Iteration: 10 -- batch score: 414.52

best score: 408.18

Total Tour distance: 857.20

Βέλτιστη διαδρομή:

Route 1: 1 54 22 73 75 76 24 68 40 5 27 1

Route 2: 1 51 77 78 4 80 34 82 79 35 36 10 52 2 70 28 1

Route 3: 1 29 13 81 69 30 25 55 56 26 57 23 42 74 41 1

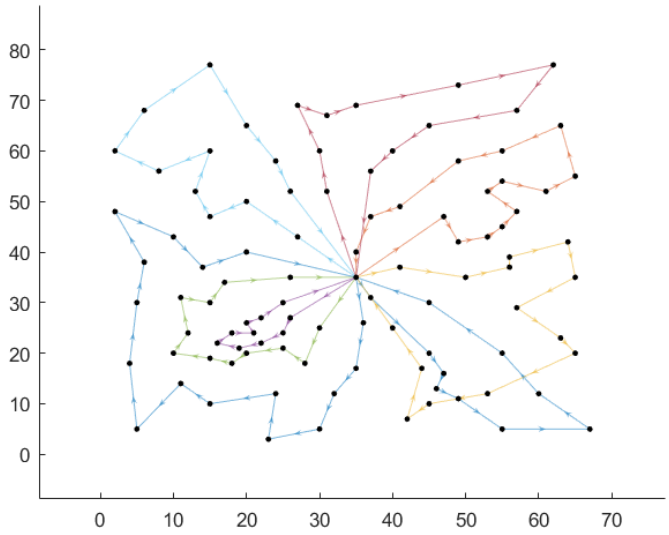
Route 4: 1 95 96 93 99 86 94 60 100 97 7 1

Route 5: 1 14 88 98 38 101 92 17 62 85 6 61 90 1

Route 6: 1 53 8 83 49 20 48 37 50 65 12 63 89 1

Route 7: 1 32 11 64 91 33 67 66 72 21 31 71 1

Route 8: 1 59 3 58 16 44 43 15 45 39 87 18 46 47 9 84 19 1



Γράφημα 9: Διαδρομές για δεδομένα 5.3.3

5.3.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (421.19lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (857.20km) με τον ελάχιστο αριθμό διαδρομών (8). Η τιμή του ρ είναι ανεξάρτητή ως προς τη τελική λύση.

5.4

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   151
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   999999
χρόνος εξυπηρέτησης του κόμβου:  0
Βάρος οχήματος χωρίς φορτίο = 6250kg
Καθαρό βάρος φόρτωσης = 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.4.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 600.62

Iteration: 2 -- batch score: 576.20

Iteration: 3 -- batch score: 606.34

Iteration: 4 -- batch score: 576.22

Iteration: 5 -- batch score: 598.63

Iteration: 6 -- batch score: 581.58

Iteration: 7 -- batch score: 587.29

Iteration: 8 -- batch score: 593.67

Iteration: 9 -- batch score: 571.85

Iteration: 10 -- batch score: 563.80

best score: 563.80

Total Tour distance: 1183.25

Βέλτιστη διαδρομή:

Route 1: 1 113 148 119 61 90 1

Route 2: 1 147 128 149 63 12 65 50 144 37 48 125 83 19 1

Route 3: 1 7 105 94 86 62 17 142 45 120 15 143 43 1

Route 4: 1 14 118 96 93 38 99 101 92 87 114 85 6 95 1

Route 5: 1 138 145 88 98 60 100 97 53 107 89 32 28 1

Route 6: 1 84 126 9 115 8 11 109 33 132 67 52 2 133 1

Route 7: 1 77 78 4 80 130 136 72 66 137 36 35 79 122 30 25 26 56 150 1

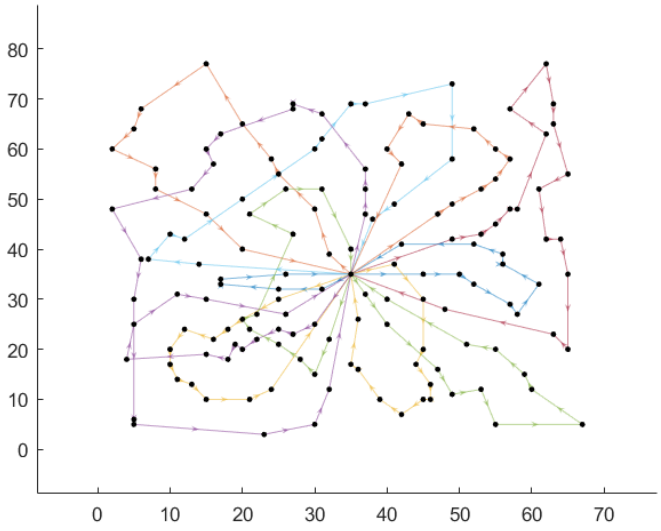
Route 8: 1 139 13 110 55 131 135 81 151 69 117 112 1

Route 9: 1 51 103 34 82 121 10 104 21 129 31 123 1

Route 10: 1 29 27 22 74 75 134 23 42 146 116 3 59 1

Route 11: 1 70 102 71 91 64 127 108 20 124 49 47 46 18 141 39 44 16 58 1

Route 12: 1 54 41 73 76 57 24 68 40 140 5 111 106 1



Γράφημα 10: Διαδρομές δεδομένων 5.4.1

5.4.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Iteration: 1 -- batch score: 530.34

Iteration: 2 -- batch score: 534.98

Iteration: 3 -- batch score: 530.78

Iteration: 4 -- batch score: 539.47

Iteration: 5 -- batch score: 525.42

Iteration: 6 -- batch score: 530.04

Iteration: 7 -- batch score: 545.97

Iteration: 8 -- batch score: 520.81

Iteration: 9 -- batch score: 527.79

Iteration: 10 -- batch score: 529.65

best score: 520.81

Total Tour distance: 1093.63

Βέλτιστη διαδρομή:

Route 1: 1 106 41 22 74 73 75 134 23 42 146 116 3 59 54 1

Route 2: 1 139 13 110 55 131 135 25 30 122 81 151 69 117 1

Route 3: 1 113 95 96 93 99 86 94 60 7 1

Route 4: 1 111 5 140 40 57 76 24 68 26 56 150 27 1

Route 5: 1 14 88 145 43 143 45 142 17 92 101 38 98 118 1

Route 6: 1 112 51 103 34 82 121 52 123 31 71 102 70 28 1

Route 7: 1 29 77 78 4 80 130 79 35 136 36 137 66 67 2 133 1

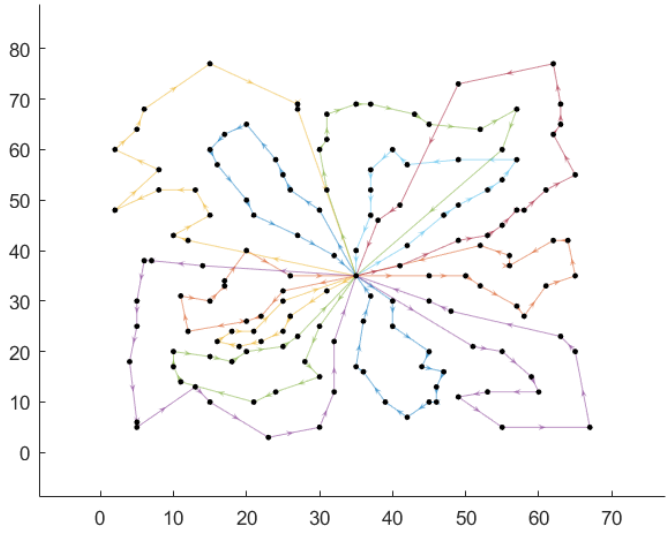
Route 8: 1 128 89 149 63 12 108 20 124 8 107 53 147 1

Route 9: 1 148 97 100 105 62 85 6 119 61 19 90 1

Route 10: 1 115 9 83 49 125 47 48 37 144 50 65 64 127 1

Route 11: 1 84 126 46 18 114 87 141 39 120 15 44 16 58 138 1

Route 12: 1 32 11 109 91 33 132 129 21 104 72 10 1



Γράφημα 11:Διαδρομές δεδομένων 5.4.2

5.4.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 530.34

Iteration: 2 -- batch score: 531.60

Iteration: 3 -- batch score: 538.40

Iteration: 4 -- batch score: 540.32

Iteration: 5 -- batch score: 536.94

Iteration: 6 -- batch score: 533.85

Iteration: 7 -- batch score: 534.21

Iteration: 8 -- batch score: 537.83

Iteration: 9 -- batch score: 541.90

Iteration: 10 -- batch score: 534.88

best score: 530.34

Total Tour distance: 1114.02

Βέλτιστη διαδρομή:

Route 1: 1 128 89 149 63 12 108 20 124 8 107 53 147 1

Route 2: 1 29 139 13 110 151 81 69 78 117 77 103 51 1

Route 3: 1 54 41 22 73 57 76 134 23 75 74 116 3 59 1

Route 4: 1 113 95 96 60 100 105 97 148 1

Route 5: 1 14 88 43 143 15 120 45 142 17 92 99 98 118 1

Route 6: 1 28 133 70 102 71 31 123 52 10 121 82 80 4 1

Route 7: 1 27 150 131 55 135 25 30 122 130 79 35 36 136 34 112 1

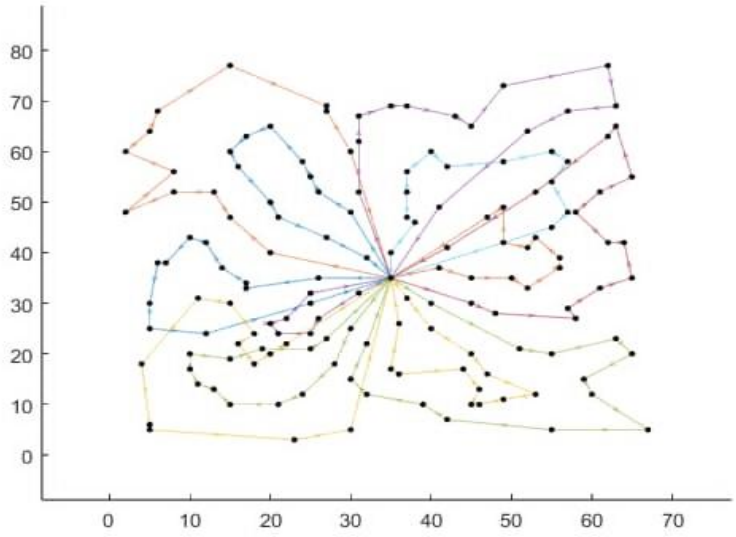
Route 8: 1 90 119 61 84 115 9 126 46 18 114 62 7 1

Route 9: 1 19 83 49 125 47 48 37 144 50 65 64 127 11 1

Route 10: 1 93 38 101 86 94 6 85 87 141 39 44 16 1

Route 11: 1 32 109 91 33 132 129 21 67 66 137 72 104 2 1

Route 12: 1 106 111 5 56 26 140 40 68 24 42 146 58 145 138 1



Γράφημα 12:Διαδρομές δεδομένων 5.4.3

5.4.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (520.81lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (1093.63km) με τον ελάχιστο αριθμό διαδρομών (12). Οι διαδρομές παραμένουν ίδιες για όλες τις παραμέτρους.

5.5

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   200
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   999999
χρόνος εξυπηρέτησης του κόμβου:  0
το βάρος οχήματος χωρίς φορτίο: 6250kg
το καθαρό βάρος φόρτωσης: 12500kg
και η ζήτηση κάθε κόμβου.

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.5.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 739.51

Iteration: 2 -- batch score: 716.66

Iteration: 3 -- batch score: 740.54

Iteration: 4 -- batch score: 730.53

Iteration: 5 -- batch score: 743.55

Iteration: 6 -- batch score: 748.90

Iteration: 7 -- batch score: 767.04

Iteration: 8 -- batch score: 758.09

Iteration: 9 -- batch score: 736.13

Iteration: 10 -- batch score: 747.14

best score: 716.66

Total Tour distance: 1501.36

Βέλτιστη διαδρομή:

Route 1: 1 59 153 74 172 134 23 42 146 58 145 88 14 1

Route 2: 1 60 194 92 142 87 114 18 174 85 119 1

Route 3: 1 51 103 158 159 13 110 196 116 179 3 1

Route 4: 1 28 70 102 31 129 67 66 137 72 10 121 82 170 122 30 25 164 135 131 40 68 24 187 57 198 1

Route 5: 1 148 6 86 99 38 101 193 45 141 39 44 16 173 1

Route 6: 1 29 117 78 4 80 130 79 35 165 136 36 162 104 52 1

Route 7: 1 113 184 95 138 75 76 73 22 41 54 1

Route 8: 1 157 7 97 105 100 94 62 17 192 120 15 143 43 98 118 1

Route 9: 1 112 185 197 77 177 190 11 32 191 1

Route 10: 1 90 167 84 200 126 46 107 154 147 1

Route 11: 1 106 181 111 156 5 180 150 27 1

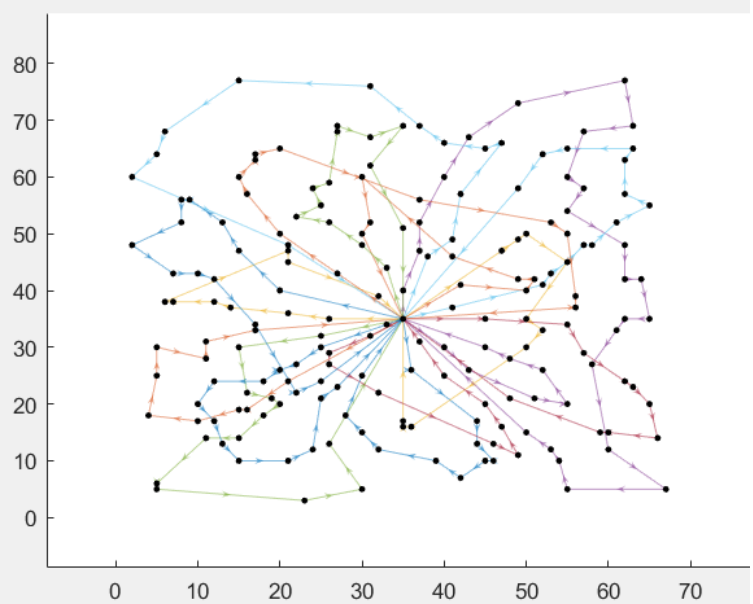
Route 12: 1 168 128 89 183 149 63 160 127 64 91 33 109 163 1

Route 13: 1 133 2 123 189 21 161 132 182 65 50 144 37 195 1

Route 14: 1 155 139 178 55 166 56 26 171 188 140 199 1

Route 15: 1 19 83 49 169 48 125 47 175 9 115 61 93 152 96 1

Route 16: 1 53 8 124 20 108 176 12 71 34 186 69 151 81 1



Γράφημα 13:Διαδρομές δεδομένων 5.5.1

5.5.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 678.41

Iteration: 2 -- batch score: 694.38

Iteration: 3 -- batch score: 683.92

Iteration: 4 -- batch score: 674.67

Iteration: 5 -- batch score: 678.20

Iteration: 6 -- batch score: 679.68

Iteration: 7 -- batch score: 696.52

Iteration: 8 -- batch score: 698.76

Iteration: 9 -- batch score: 677.15

Iteration: 10 -- batch score: 684.34

best score: 674.67

Total Tour distance: 1415.68

Βέλτιστη διαδρομή:

Route 1: 1 157 113 148 184 95 152 93 98 96 14 1

Route 2: 1 153 59 181 150 196 110 178 69 117 185 1

Route 3: 1 147 168 128 191 32 163 70 133 28 1

Route 4: 1 29 197 78 4 159 77 177 112 1

Route 5: 1 90 115 9 175 46 126 200 84 61 119 94 1

Route 6: 1 155 139 111 5 156 140 188 40 199 106 1

Route 7: 1 53 154 107 195 8 183 149 63 160 11 190 89 1

Route 8: 1 7 97 100 105 60 86 194 92 193 101 99 38 118 138 1

Route 9: 1 13 81 151 135 164 25 30 122 170 79 35 165 130 80 186 1

Route 10: 1 88 145 173 43 143 15 120 45 192 142 17 62 174 1

Route 11: 1 54 41 172 23 134 187 57 198 73 22 1

Route 12: 1 19 83 49 124 20 108 176 12 127 64 91 33 132 1

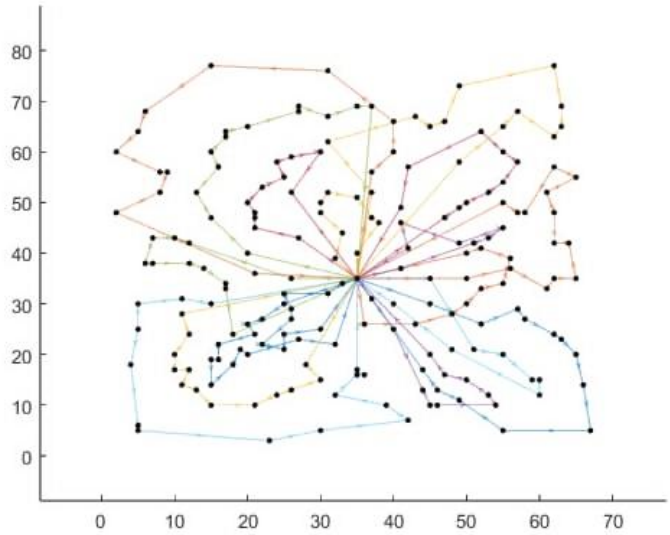
Route 13: 1 6 85 18 114 87 141 39 44 16 42 146 58 179 116 3 1

Route 14: 1 51 103 158 34 82 121 10 104 123 2 1

Route 15: 1 27 180 55 131 166 56 26 171 68 24 76 75 74 1

Route 16: 1 71 31 161 182 65 50 144 37 48 169 125 47 167 1

Route 17: 1 52 162 72 136 36 137 66 67 189 21 129 109 102 1



Γράφημα 14:Διαδρομές δεδομένων 5.5.2

5.5.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 678.41

Iteration: 2 -- batch score: 698.46

Iteration: 3 -- batch score: 682.44

Iteration: 4 -- batch score: 683.27

Iteration: 5 -- batch score: 677.67

Iteration: 6 -- batch score: 671.47

Iteration: 7 -- batch score: 691.64

Iteration: 8 -- batch score: 666.64

Iteration: 9 -- batch score: 684.90

Iteration: 10 -- batch score: 673.70

Best score: 666.64

Total Tour distance: 1399.26

Βέλτιστη διαδρομή:

Route 1: 1 157 113 184 152 93 96 95 1

Route 2: 1 54 153 59 106 181 27 150 196 110 139 29 1

Route 3: 1 28 168 128 191 32 163 70 133 177 1

Route 4: 1 77 197 117 78 4 159 130 80 186 82 34 112 1

Route 5: 1 148 100 105 94 86 92 194 101 99 60 97 7 1

Route 6: 1 90 19 154 107 195 8 183 63 160 149 89 1

Route 7: 1 102 190 11 109 33 132 161 129 21 189 104 52 1

Route 8: 1 155 185 69 81 151 178 55 131 166 56 26 156 111 1

Route 9: 1 121 10 162 72 136 36 137 66 67 31 123 1

Route 10: 1 119 6 62 17 142 192 45 120 193 38 1

Route 11: 1 51 103 158 165 35 79 170 122 30 25 164 135 13 1

Route 12: 1 14 88 145 173 43 143 44 16 58 179 116 3 138 1

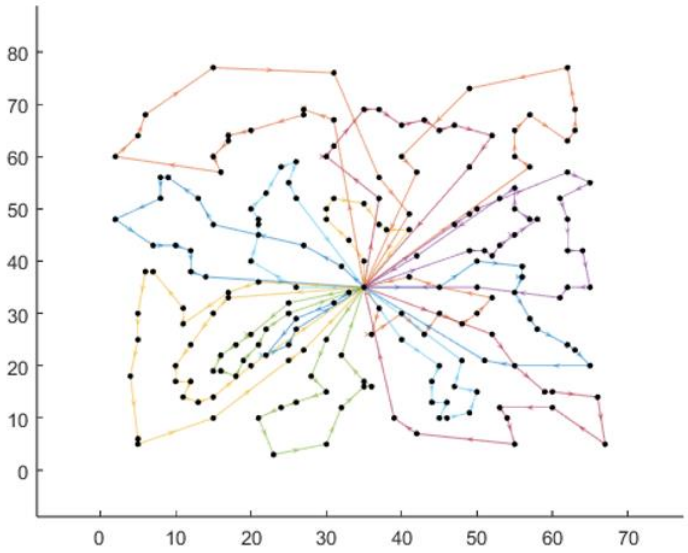
Route 13: 1 41 22 74 172 75 23 134 76 198 73 199 1

Route 14: 1 180 5 140 188 171 68 40 57 187 24 42 146 1

Route 15: 1 147 53 83 49 169 48 125 47 175 9 115 200 84 1

Route 16: 1 91 64 127 12 176 108 20 124 37 144 50 65 182 71 2 1

Route 17: 1 167 61 174 85 126 46 18 114 87 141 39 15 98 118 1



Γράφημα 15:Διαδρομές δεδομένων 5.5.3

5.5.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.2 έχουμε την λιγότερη κατανάλωση καυσίμου (666.64lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (1399.26km) παρά το γεγονός πως δεν έχουμε τον ελάχιστο αριθμό διαδρομών (16) της πρώτης παραλλαγής, αλλά μία παραπάνω (17). Η ρύθμιση του ρ μίκρυνε την απόσταση χωρίς να αλλάζει το σύνολο των διαδρομών.

5.6

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:    51
Χωρητικότητα οχήματος:     160
Χρονικό όριο διαδρομής:    200
χρόνος εξυπηρέτησης του κόμβου:  10
Το βάρος οχήματος χωρίς φορτίο: 5000kg
Το καθαρό βάρος φόρτωσης: 10000kg
Και η ζήτηση κάθε κόμβου,

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.6.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 408.53

Iteration: 2 -- batch score: 402.69

Iteration: 3 -- batch score: 404.79

Iteration: 4 -- batch score: 403.91

Iteration: 5 -- batch score: 402.32

Iteration: 6 -- batch score: 413.82

Iteration: 7 -- batch score: 406.03

Iteration: 8 -- batch score: 400.71

Iteration: 9 -- batch score: 404.22

Iteration: 10 -- batch score: 410.72

best score: 400.71

Total Tour distance: 567.37

Βέλτιστη διαδρομή:

Route 1: 1 13 18 38 45 16 46 34 40 11 1

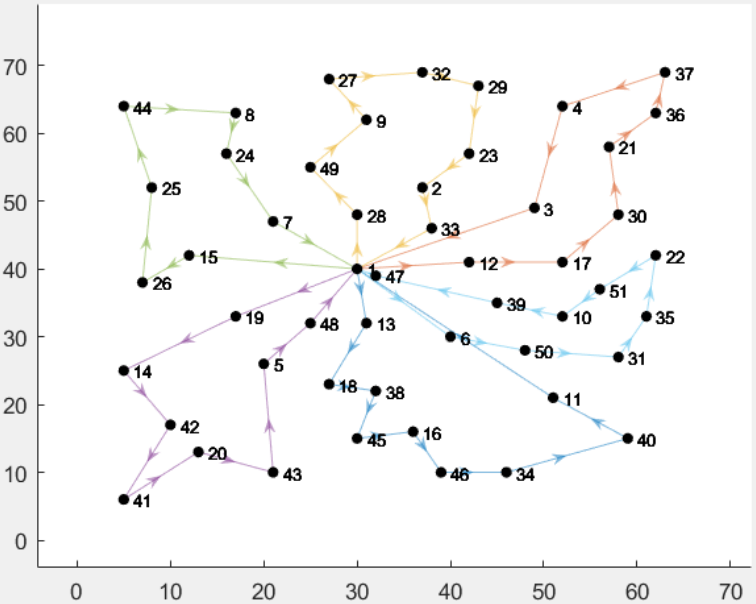
Route 2: 1 12 17 30 21 36 37 4 3 1

Route 3: 1 28 49 9 27 32 29 23 2 33 1

Route 4: 1 19 14 42 41 20 43 5 48 1

Route 5: 1 15 26 25 44 8 24 7 1

Route 6: 1 6 50 31 35 22 51 10 39 47 1



Γράφημα 16:Διαδρομές δεδομένων 5.6.1

5.6.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 405.67

Iteration: 2 -- batch score: 411.66

Iteration: 3 -- batch score: 404.55

Iteration: 4 -- batch score: 403.44

Iteration: 5 -- batch score: 409.83

Iteration: 6 -- batch score: 410.63

Iteration: 7 -- batch score: 411.10

Iteration: 8 -- batch score: 402.25

Iteration: 9 -- batch score: 403.07

Iteration: 10 -- batch score: 401.59

best score: 401.59

Total Tour distance: 569.90

Βέλτιστη διαδρομή:

Route 1: 1 13 38 16 46 34 40 11 50 6 1

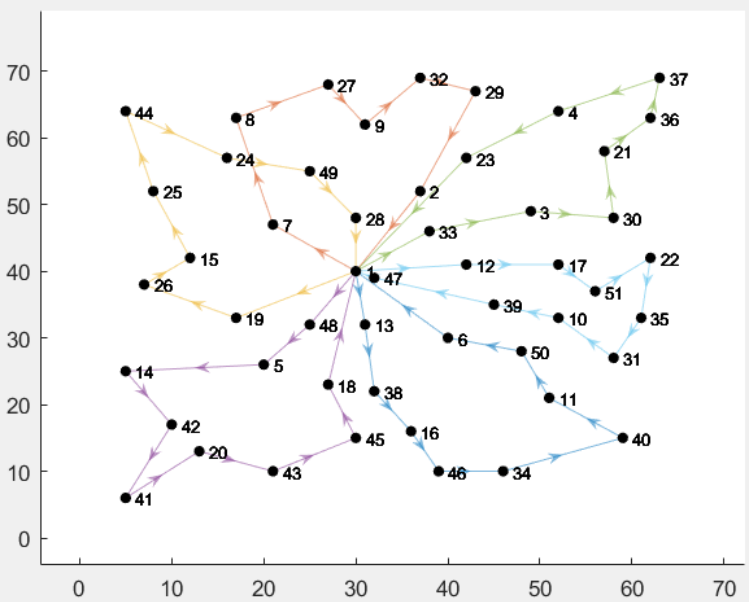
Route 2: 1 7 8 27 9 32 29 2 1

Route 3: 1 19 26 15 25 44 24 49 28 1

Route 4: 1 48 5 14 42 41 20 43 45 18 1

Route 5: 1 33 3 30 21 36 37 4 23 1

Route 6: 1 12 17 51 22 35 31 10 39 47 1



Γράφημα 17:Διαδρομές δεδομένων 5.6.2

5.6.3 α=0.2 β=10 ρ=0.2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 405.67

Iteration: 2 -- batch score: 408.65

Iteration: 3 -- batch score: 403.42

Iteration: 4 -- batch score: 402.62

Iteration: 5 -- batch score: 409.52

Iteration: 6 -- batch score: 406.45

Iteration: 7 -- batch score: 400.76

Iteration: 8 -- batch score: 411.72

Iteration: 9 -- batch score: 408.48

Iteration: 10 -- batch score: 411.09

best score: 400.76

Total Tour distance: 567.33

Βέλτιστη διαδρομή:

Route 1: 1 6 50 11 40 34 46 16 18 48 1

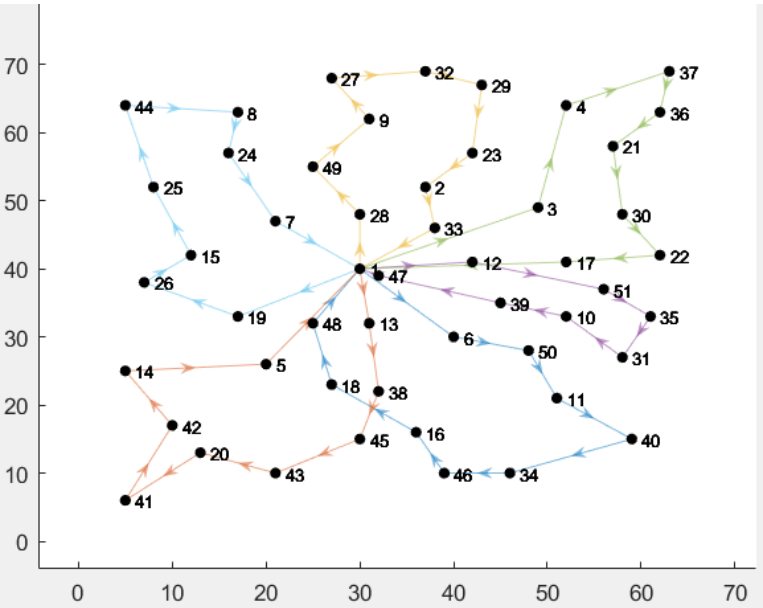
Route 2: 1 13 38 45 43 20 41 42 14 5 1

Route 3: 1 28 49 9 27 32 29 23 2 33 1

Route 4: 1 12 51 35 31 10 39 47 1

Route 5: 1 3 4 37 36 21 30 22 17 1

Route 6: 1 19 26 15 25 44 8 24 7 1



Γράφημα 18:Διαδρομές δεδομένων 5.6.3

5.6.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως και για τις 3 περιπτώσεις τα αποτελέσματα είναι πρακτικά ίδια. Για α=1, β=1 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (400.71lt) με διαφορά 50 ml από τη δεύτερη ενώ η μικρότερη συνολική χιλιομετρική απόσταση (567.33km) επιτυγχάνεται στην τρίτη περίπτωση με διαφορά μόλις 40 μέτρα. Ο αριθμός διαδρομών (6) είναι ίδια σε όλες τις περιπτώσεις. Με γνώμονα την ελαχιστοποίηση κατανάλωσης καυσίμου θα επιλέξουμε την πρώτη περίπτωση (α=1 β=1 ρ=0.4).

5.7

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   76
Χωρητικότητα οχήματος:    140
Χρονικό όριο διαδρομής:   160
χρόνος εξυπηρέτησης του κόμβου:  10
Το βάρος οχήματος χωρίς φορτίο: 4375kg
Το καθαρό βάρος φόρτωσης: 8750kg
Και η ζήτηση κάθε κόμβου,

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.7.1 α=1 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 560.09

Iteration: 2 -- batch score: 561.69

Iteration: 3 -- batch score: 560.22

Iteration: 4 -- batch score: 554.50

Iteration: 5 -- batch score: 553.23

Iteration: 6 -- batch score: 555.05

Iteration: 7 -- batch score: 553.90

Iteration: 8 -- batch score: 556.89

Iteration: 9 -- batch score: 561.24

Iteration: 10 -- batch score: 561.89

best score: 553.23

Total Tour distance: 950.26

Βέλτιστη διαδρομή:

Route 1: 1 31 22 70 72 37 48 49 1

Route 2: 1 35 53 55 14 28 1

Route 3: 1 41 33 51 19 56 26 1

Route 4: 1 69 3 29 62 75 76 1

Route 5: 1 5 46 30 6 1

Route 6: 1 64 24 57 42 44 74 1

Route 7: 1 7 34 2 43 65 23 63 1

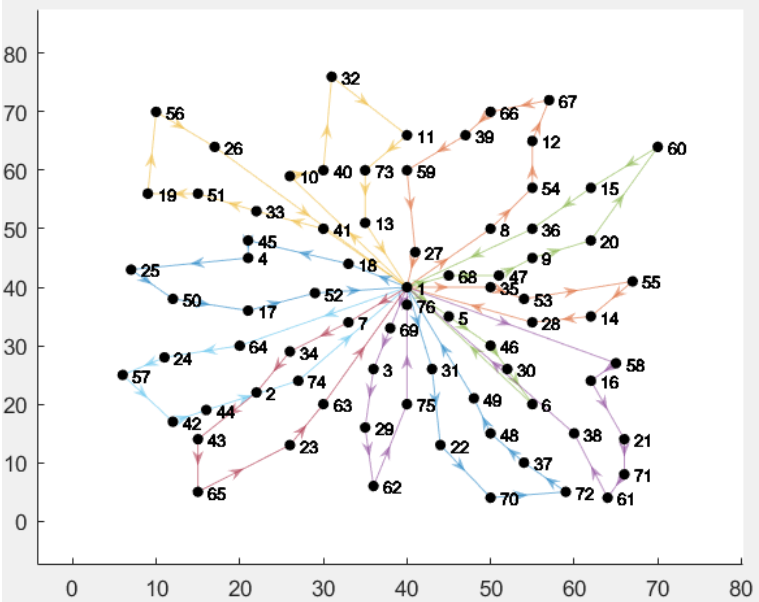
Route 8: 1 18 45 4 25 50 17 52 1

Route 9: 1 8 54 12 67 66 39 59 27 1

Route 10: 1 10 40 32 11 73 13 1

Route 11: 1 58 16 21 71 61 38 1

Route 12: 1 68 47 9 20 60 15 36 1



Γράφημα 19:Διαδρομές δεδομένων 5.7.1

5.7.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 560.09

Iteration: 2 -- batch score: 559.59

Iteration: 3 -- batch score: 561.79

Iteration: 4 -- batch score: 563.11

Iteration: 5 -- batch score: 558.59

Iteration: 6 -- batch score: 562.04

Iteration: 7 -- batch score: 558.46

Iteration: 8 -- batch score: 571.26

Iteration: 9 -- batch score: 562.30

Iteration: 10 -- batch score: 561.94

best score: 558.46

Total Tour distance: 966.84

Βέλτιστη διαδρομή:

Route 1: 1 41 33 45 4 52 1

Route 2: 1 35 53 55 14 58 16 28 1

Route 3: 1 49 48 37 72 61 71 1

Route 4: 1 76 69 3 63 23 2 34 7 1

Route 5: 1 68 47 9 20 60 15 36 1

Route 6: 1 8 54 12 67 66 39 27 1

Route 7: 1 18 51 19 56 26 10 1

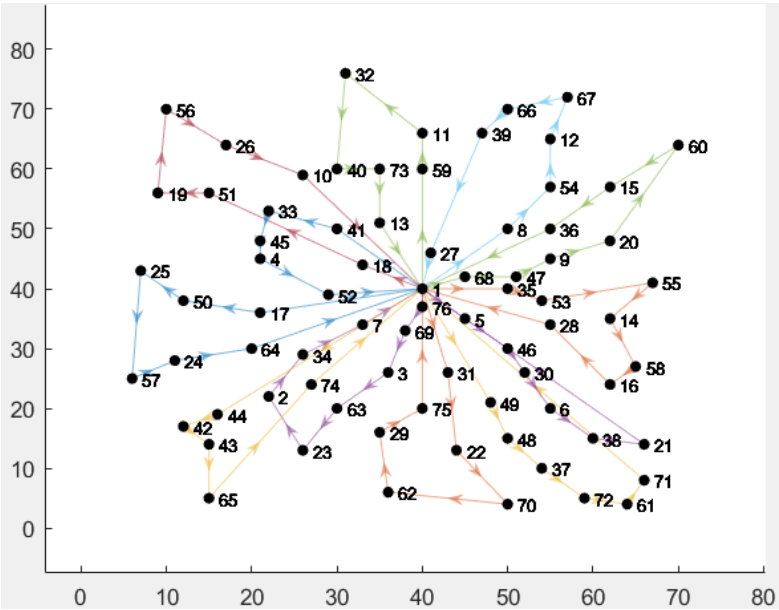
Route 8: 1 17 50 25 57 24 64 1

Route 9: 1 31 22 70 62 29 75 1

Route 10: 1 44 42 43 65 74 1

Route 11: 1 5 46 30 6 38 21 1

Route 12: 1 59 11 32 40 73 13 1



Γράφημα 20:Διαδρομές δεδομένων 5.7.2

5.7.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 560.09

Iteration: 2 -- batch score: 559.64

Iteration: 3 -- batch score: 558.76

Iteration: 4 -- batch score: 561.26

Iteration: 5 -- batch score: 564.21

Iteration: 6 -- batch score: 561.16

Iteration: 7 -- batch score: 562.95

Iteration: 8 -- batch score: 563.96

Iteration: 9 -- batch score: 559.07

Iteration: 10 -- batch score: 553.49

best score: 553.49

Total Tour distance: 950.45

Βέλτιστη διαδρομή:

Route 1: 1 76 69 1

Route 2: 1 33 19 51 10 40 73 13 1

Route 3: 1 3 75 29 62 22 48 31 1

Route 4: 1 64 24 57 42 44 74 1

Route 5: 1 18 41 45 4 25 50 17 52 1

Route 6: 1 49 37 70 72 61 30 1

Route 7: 1 35 53 28 58 14 55 47 1

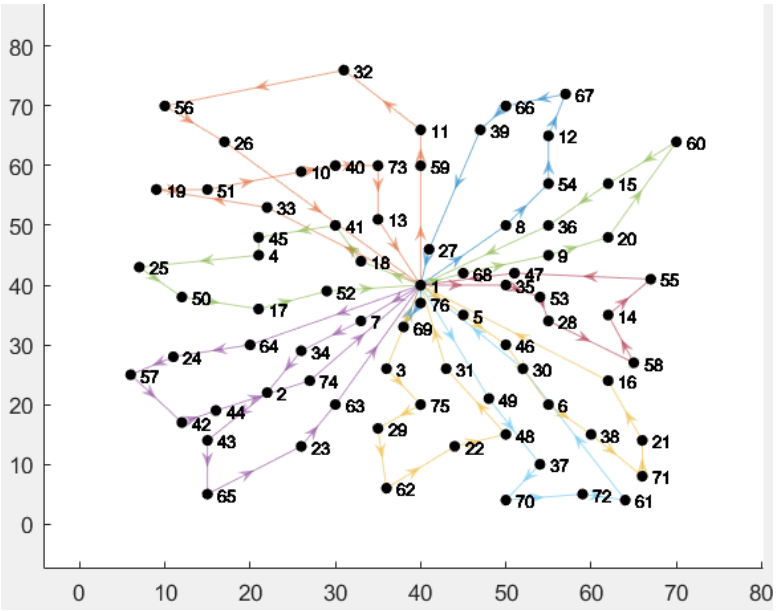
Route 8: 1 8 54 12 67 66 39 27 1

Route 9: 1 59 11 32 56 26 1

Route 10: 1 5 46 6 38 71 21 16 1

Route 11: 1 7 34 2 43 65 23 63 1

Route 12: 1 68 9 20 60 15 36 1



Γράφημα 21:Διαδρομές δεδομένων 5.7.3

5.7.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=1, β=1 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (553.23lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (950.26km) με τον ελάχιστο αριθμό διαδρομών (12) ο οποίος είναι ίδιος σε όλες τις περιπτώσεις. Η ρύθμιση του ρ (τρίτη περίπτωση) έχει ως αποτέλεσμα να πλησιάσει στη βέλτιστη λύση της πρώτης περίπτωσης.

5.8

Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών: 101
Χωρητικότητα οχήματος: 200
Χρονικό όριο διαδρομής: 230
χρόνος εξυπηρέτησης του κόμβου: 10
Το βάρος οχήματος χωρίς φορτίο: 6250kg
Το καθαρό βάρος φόρτωσης: 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.

5.8.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 921.07

Iteration: 2 -- batch score: 932.11

Iteration: 3 -- batch score: 921.83

Iteration: 4 -- batch score: 900.47

Iteration: 5 -- batch score: 918.48

Iteration: 6 -- batch score: 906.08

Iteration: 7 -- batch score: 925.24

Iteration: 8 -- batch score: 921.09

Iteration: 9 -- batch score: 924.50

Iteration: 10 -- batch score: 920.21

best score: 900.47

Total Tour distance: 891.95

Βέλτιστη διαδρομή:

Route 1: 1 13 81 69 4 80 79 35 82 34 51 1

Route 2: 1 14 96 98 88 43 101 92 99 38 93 60 95 1

Route 3: 1 62 17 87 39 45 15 44 16 58 3 59 1

Route 4: 1 7 97 100 94 86 6 85 18 46 84 61 90 1

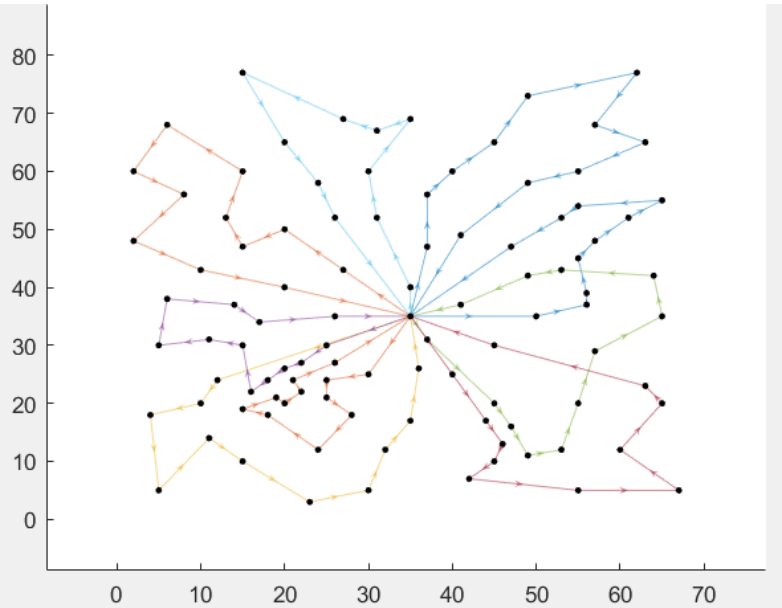
Route 5: 1 54 22 73 76 57 5 55 25 30 78 77 29 1

Route 6: 1 28 32 11 33 91 64 65 12 63 89 1

Route 7: 1 41 74 75 23 42 24 68 40 26 56 27 1

Route 8: 1 70 71 31 21 67 66 72 36 10 52 2 1

Route 9: 1 53 8 83 49 20 50 37 48 47 9 19 1



Γράφημα 22:Διαδρομές δεδομένων 5.8.1

5.8.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις;

Iteration: 1 -- batch score: 918.18

Iteration: 2 -- batch score: 927.87

Iteration: 3 -- batch score: 918.98

Iteration: 4 -- batch score: 924.24

Iteration: 5 -- batch score: 916.72

Iteration: 6 -- batch score: 919.47

Iteration: 7 -- batch score: 920.61

Iteration: 8 -- batch score: 911.16

Iteration: 9 -- batch score: 924.47

Iteration: 10 -- batch score: 929.25

best score: 911.16

Total Tour distance: 915.11

Βέλτιστη διαδρομή:

Route 1: 1 7 100 60 94 86 62 92 101 99 38 93 98 96 1

Route 2: 1 77 78 4 80 82 79 35 30 25 81 69 1

Route 3: 1 54 41 74 75 76 23 42 58 3 59 1

Route 4: 1 29 27 1

Route 5: 1 95 6 85 18 46 47 9 84 61 90 1

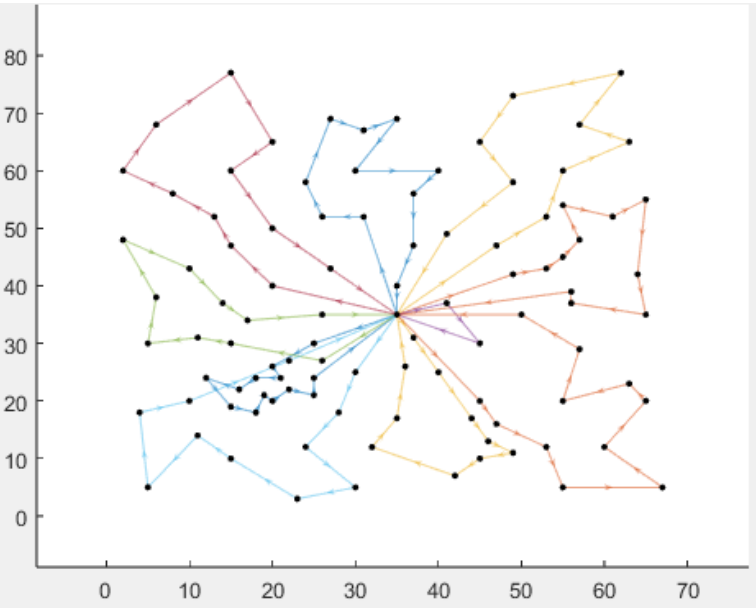
Route 6: 1 14 88 43 16 44 15 45 39 87 17 97 1

Route 7: 1 19 83 49 48 37 50 65 12 20 8 53 1

Route 8: 1 32 89 63 64 91 33 11 31 71 70 28 1

Route 9: 1 22 73 57 24 68 40 26 56 5 55 13 1

Route 10: 1 51 34 10 36 72 66 67 21 52 2 1



Γράφημα 23:Διαδρομές δεδομένων 5.8.2

5.8.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 918.18

Iteration: 2 -- batch score: 920.60

Iteration: 3 -- batch score: 914.08

Iteration: 4 -- batch score: 914.81

Iteration: 5 -- batch score: 914.86

Iteration: 6 -- batch score: 920.19

Iteration: 7 -- batch score: 924.77

Iteration: 8 -- batch score: 926.27

Iteration: 9 -- batch score: 918.64

Iteration: 10 -- batch score: 922.74

best score: 914.08

Total Tour distance: 920.30

Βέλτιστη διαδρομή:

Route 1: 1 19 84 9 49 48 37 50 65 64 63 1

Route 2: 1 77 78 4 80 34 82 79 35 30 25 81 69 1

Route 3: 1 90 61 6 85 18 46 47 83 8 53 1

Route 4: 1 95 97 100 60 94 99 86 92 101 98 88 14 59 54 1

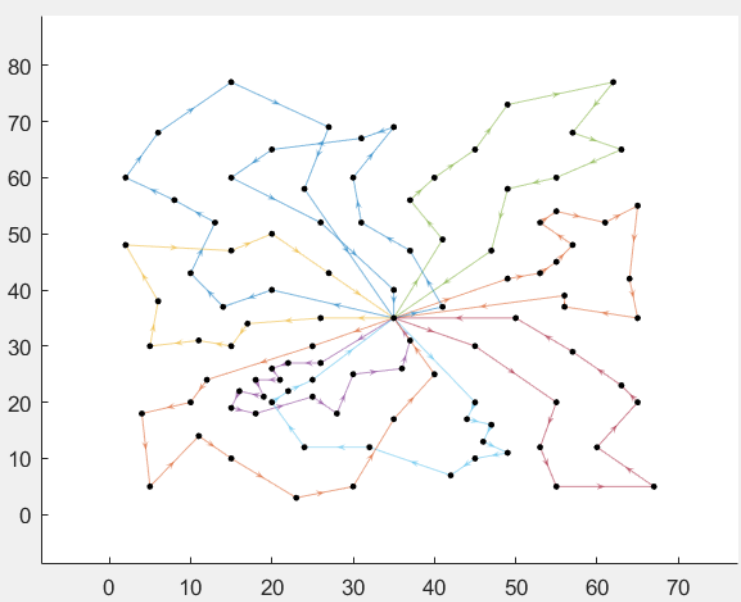
Route 5: 1 2 71 31 21 67 66 72 36 10 52 51 1

Route 6: 1 22 74 73 75 76 23 42 58 43 38 93 96 1

Route 7: 1 27 5 57 24 68 40 26 56 55 13 1

Route 8: 1 29 70 32 11 33 91 12 20 89 28 1

Route 9: 1 7 62 17 87 39 45 15 44 16 3 41 1



Γράφημα 24:Διαδρομές δεδομένων 5.8.3

5.8.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=1, β=1 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (900.47lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (891.95km) με τον ελάχιστο αριθμό διαδρομών (9). Η ρύθμιση του ρ βελτίωσε τον αριθμό των διαδρομών ενώ αύξησε τη κατανάλωση και την χιλιομετρική απόσταση.

5.9

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   151
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   200
χρόνος εξυπηρέτησης του κόμβου:  10
Το βάρος οχήματος χωρίς φορτίο:  6250kg
Το καθαρό βάρος φόρτωσης: 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.9.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 1352.98

Iteration: 2 -- batch score: 1358.61

Iteration: 3 -- batch score: 1350.94

Iteration: 4 -- batch score: 1329.96

Iteration: 5 -- batch score: 1340.44

Iteration: 6 -- batch score: 1351.03

Iteration: 7 -- batch score: 1355.33

Iteration: 8 -- batch score: 1354.37

Iteration: 9 -- batch score: 1326.80

Iteration: 10 -- batch score: 1341.48

best score: 1326.80

Total Tour distance: 1292.50

Βέλτιστη διαδρομή:

Route 1: 1 109 91 33 132 129 67 104 21 2 1

Route 2: 1 97 105 100 86 17 142 120 15 101 38 93 96 1

Route 3: 1 113 95 7 148 1

Route 4: 1 128 107 8 124 49 48 125 83 53 147 1

Route 5: 1 60 94 62 114 87 141 39 45 92 99 1

Route 6: 1 73 76 57 24 68 40 140 26 56 1

Route 7: 1 28 32 11 127 64 65 12 63 149 89 1

Route 8: 1 151 81 69 78 103 123 31 71 102 70 133 1

Route 9: 1 59 138 88 145 58 16 44 143 43 98 118 14 1

Route 10: 1 51 34 82 136 36 137 66 72 52 1

Route 11: 1 90 61 119 6 85 18 46 126 84 1

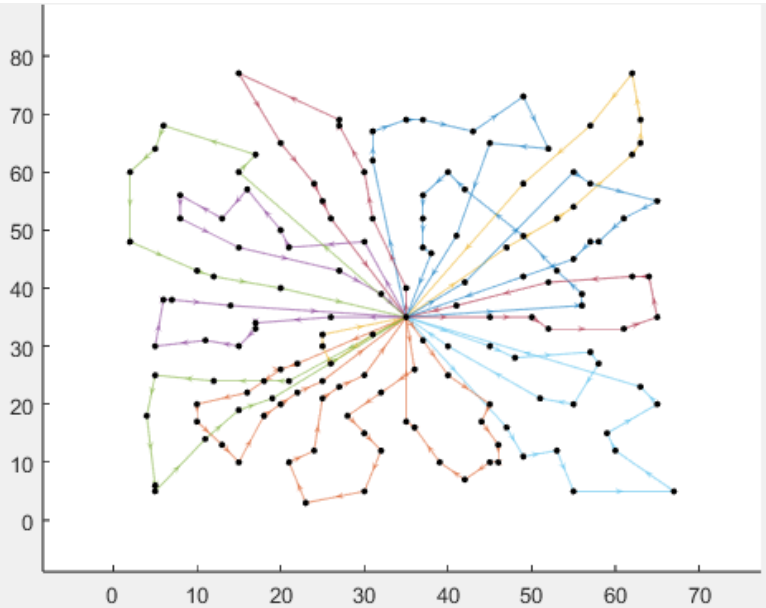
Route 12: 1 20 108 50 144 37 47 9 115 19 1

Route 13: 1 27 150 55 131 5 111 106 1

Route 14: 1 139 13 110 135 25 30 122 117 29 1

Route 15: 1 112 10 121 35 79 130 80 4 77 1

Route 16: 1 54 41 22 74 75 134 23 42 146 116 3 1



Γράφημα 25:Διαδρομές δεδομένων 5.9.1

5.9.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 1314.89

Iteration: 2 -- batch score: 1344.09

Iteration: 3 -- batch score: 1345.13

Iteration: 4 -- batch score: 1337.58

Iteration: 5 -- batch score: 1323.48

Iteration: 6 -- batch score: 1343.61

Iteration: 7 -- batch score: 1338.76

Iteration: 8 -- batch score: 1329.37

Iteration: 9 -- batch score: 1332.33

Iteration: 10 -- batch score: 1329.44

best score: 1314.89

Total Tour distance: 1266.24

Βέλτιστη διαδρομή:

Route 1: 1 139 13 69 122 30 80 77 29 1

Route 2: 1 113 7 97 100 105 60 98 14 138 59 54 1

Route 3: 1 28 70 102 71 31 129 67 21 123 133 1

Route 4: 1 128 32 109 132 33 91 64 127 11 1

Route 5: 1 148 99 86 92 142 62 6 119 61 84 90 1

Route 6: 1 89 149 63 12 108 20 124 8 107 53 147 1

Route 7: 1 95 96 93 38 101 45 120 15 143 43 88 118 1

Route 8: 1 106 41 74 111 131 55 135 25 151 81 110 1

Route 9: 1 19 115 9 47 46 126 18 114 85 1

Route 10: 1 27 150 73 75 76 134 23 42 146 116 3 1

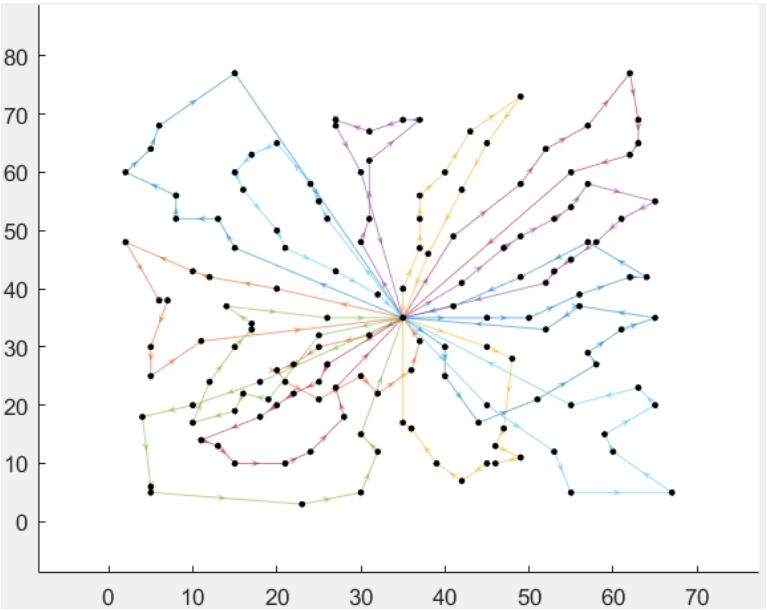
Route 11: 1 112 51 103 34 82 121 35 79 130 4 78 117 1

Route 12: 1 94 17 87 141 39 44 16 58 145 1

Route 13: 1 22 57 24 68 40 140 26 56 5 1

Route 14: 1 2 52 104 72 66 137 36 136 10 1

Route 15: 1 83 49 125 48 37 144 50 65 1



Γράφημα 26:Διαδρομές δεδομένων 5.9.2

5.9.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 1314.89

Iteration: 2 -- batch score: 1320.42

Iteration: 3 -- batch score: 1327.08

Iteration: 4 -- batch score: 1330.15

Iteration: 5 -- batch score: 1320.44

Iteration: 6 -- batch score: 1332.22

Iteration: 7 -- batch score: 1327.19

Iteration: 8 -- batch score: 1325.87

Iteration: 9 -- batch score: 1337.28

Iteration: 10 -- batch score: 1339.88

best score: 1314.89

Total Tour distance: 1266.24

Βέλτιστη διαδρομή:

Route 1: 1 139 13 69 122 30 80 77 29 1

Route 2: 1 113 7 97 100 105 60 98 14 138 59 54 1

Route 3: 1 28 70 102 71 31 129 67 21 123 133 1

Route 4: 1 128 32 109 132 33 91 64 127 11 1

Route 5: 1 148 99 86 92 142 62 6 119 61 84 90 1

Route 6: 1 89 149 63 12 108 20 124 8 107 53 147 1

Route 7: 1 95 96 93 38 101 45 120 15 143 43 88 118 1

Route 8: 1 106 41 74 111 131 55 135 25 151 81 110 1

Route 9: 1 19 115 9 47 46 126 18 114 85 1

Route 10: 1 27 150 73 75 76 134 23 42 146 116 3 1

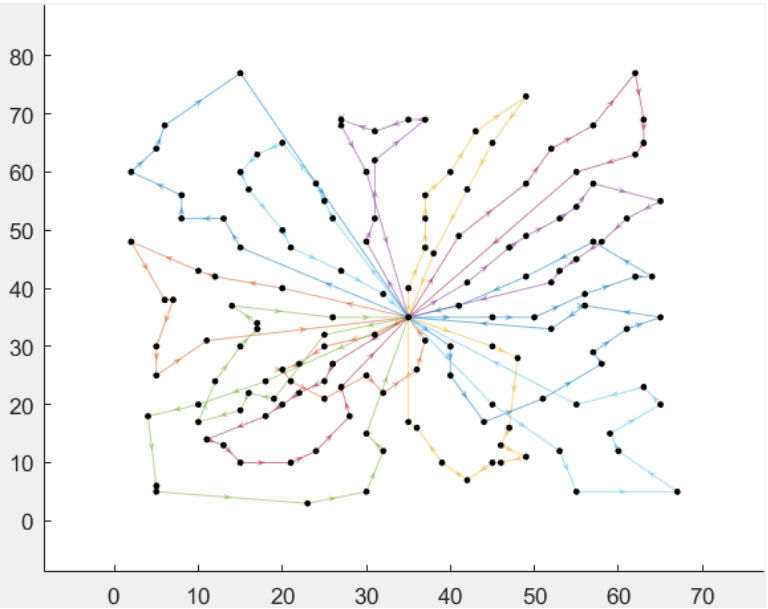
Route 11: 1 112 51 103 34 82 121 35 79 130 4 78 117 1

Route 12: 1 94 17 87 141 39 44 16 58 145 1

Route 13: 1 22 57 24 68 40 140 26 56 5 1

Route 14: 1 2 52 104 72 66 137 36 136 10 1

Route 15: 1 83 49 125 48 37 144 50 65 1



Γράφημα 27:Διαδρομές δεδομένων 5.9.3

5.9.4 Σύγκριση αποτελεσμάτων

Έχουμε βέλτιστη λύση με ελάχιστη κατανάλωση καυσίμου (1314.89lt) και ελάχιστη χιλιομετρική απόσταση (1266.24km) για α=0.2και β=10 με ελάχιστο αριθμό διαδρομών (15) . Η ρύθμιση του ρ δεν επηρεάζει την βέλτιστη λύση.

5.10

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   200
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   200
Χρόνος εξυπηρέτησης του κόμβου:  10
Το βάρος οχήματος χωρίς φορτίο:  6250kg
Το καθαρό βάρος φόρτωσης: 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.10.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 1714.57

Iteration: 2 -- batch score: 1733.36

Iteration: 3 -- batch score: 1741.03

Iteration: 4 -- batch score: 1722.35

Iteration: 5 -- batch score: 1721.63

Iteration: 6 -- batch score: 1696.56

Iteration: 7 -- batch score: 1717.73

Iteration: 8 -- batch score: 1727.88

Iteration: 9 -- batch score: 1724.10

Iteration: 10 -- batch score: 1734.20

best score: 1696.56

Total Tour distance: 1574.55

Βέλτιστη διαδρομή:

Route 1: 1 27 150 180 156 5 140 188 171 26 56 166 131 1

Route 2: 1 97 94 86 120 45 87 114 18 85 6 7 1

Route 3: 1 89 149 12 65 50 144 37 195 1

Route 4: 1 183 63 176 108 20 124 8 107 53 147 1

Route 5: 1 155 139 110 178 164 122 30 25 135 55 196 1

Route 6: 1 14 145 173 43 143 44 16 58 179 116 3 1

Route 7: 1 185 197 117 78 159 4 69 151 81 13 1

Route 8: 1 181 199 73 75 134 23 42 146 172 74 41 54 1

Route 9: 1 177 123 21 189 104 10 121 79 130 186 1

Route 10: 1 154 83 49 169 48 125 47 175 9 115 19 1

Route 11: 1 168 32 33 132 161 71 102 70 28 1

Route 12: 1 90 148 84 61 119 100 105 60 152 98 88 138 1

Route 13: 1 112 51 103 77 29 1

Route 14: 1 106 111 40 68 24 187 57 76 198 22 1

Route 15: 1 95 96 99 101 193 141 39 15 38 93 118 1

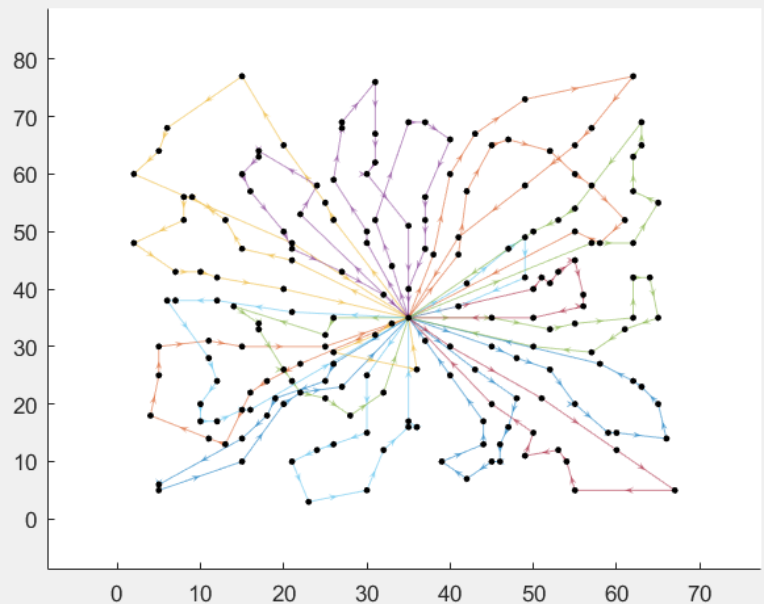
Route 16: 1 133 31 129 67 66 72 162 52 2 1

Route 17: 1 157 113 184 153 59 1

Route 18: 1 128 191 160 127 64 182 91 109 11 190 163 1

Route 19: 1 80 170 35 165 136 36 137 82 34 158 1

Route 20: 1 167 200 126 46 174 62 17 142 192 92 194 1



Γράφημα 28:Διαδρομές δεδομένων 5.10.1

5.10.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 1695.84

Iteration: 2 -- batch score: 1709.12

Iteration: 3 -- batch score: 1705.03

Iteration: 4 -- batch score: 1695.20

Iteration: 5 -- batch score: 1691.27

Iteration: 6 -- batch score: 1721.09

Iteration: 7 -- batch score: 1708.12

Iteration: 8 -- batch score: 1699.35

Iteration: 9 -- batch score: 1709.97

Iteration: 10 -- batch score: 1697.21

best score: 1691.27

Total Tour distance: 1564.39

Βέλτιστη διαδρομή:

Route 1: 1 157 113 90 154 107 195 53 147 1

Route 2: 1 199 111 156 5 75 172 134 23 42 146 116 1

Route 3: 1 167 119 61 84 85 174 18 114 17 62 148 1

Route 4: 1 54 41 22 73 198 57 187 40 188 140 180 27 1

Route 5: 1 184 95 96 152 93 38 194 92 99 100 105 97 7 1

Route 6: 1 128 191 32 163 102 70 133 28 1

Route 7: 1 106 181 150 196 110 178 151 81 69 117 185 112 1

Route 8: 1 177 51 103 158 34 82 186 197 77 29 1

Route 9: 1 14 88 173 43 143 15 120 45 193 101 98 118 1

Route 10: 1 153 59 3 179 145 138 1

Route 11: 1 168 71 33 132 161 129 189 21 31 123 1

Route 12: 1 52 162 10 121 165 35 79 170 130 1

Route 13: 1 94 86 6 200 126 46 175 9 115 19 1

Route 14: 1 89 183 149 63 160 127 64 91 109 190 11 1

Route 15: 1 2 67 66 137 36 136 72 104 1

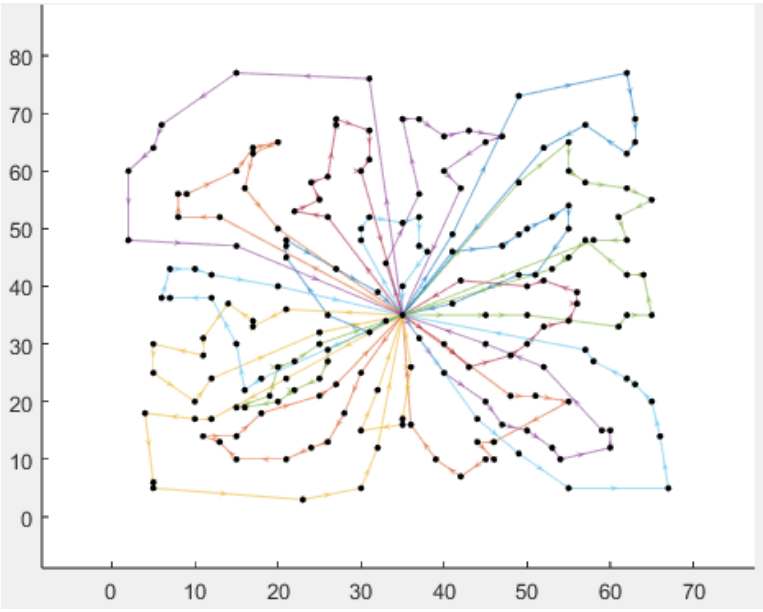
Route 16: 1 49 125 48 169 20 176 12 108 124 8 1

Route 17: 1 60 192 142 87 141 39 44 16 58 1

Route 18: 1 182 65 50 144 37 47 83 1

Route 19: 1 155 139 13 135 164 25 30 122 80 159 4 78 1

Route 20: 1 74 76 24 68 171 26 56 166 131 55 1



Γράφημα 29:Διαδρομές δεδομένων 5.10.2

5.10.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 1695.84

Iteration: 2 -- batch score: 1700.27

Iteration: 3 -- batch score: 1710.39

Iteration: 4 -- batch score: 1716.09

Iteration: 5 -- batch score: 1717.97

Iteration: 6 -- batch score: 1707.94

Iteration: 7 -- batch score: 1717.99

Iteration: 8 -- batch score: 1688.97

Iteration: 9 -- batch score: 1709.47

Iteration: 10 -- batch score: 1711.51

best score: 1688.97

Total Tour distance: 1557.49

Βέλτιστη διαδρομή:

Route 1: 1 184 60 94 92 193 120 17 62 100 105 97 7 1

Route 2: 1 147 154 107 195 183 63 160 11 190 71 102 70 1

Route 3: 1 155 139 151 81 69 159 4 78 197 117 185 77 28 1

Route 4: 1 168 191 32 33 132 161 129 21 123 2 177 1

Route 5: 1 157 113 95 96 152 93 98 118 14 153 59 1

Route 6: 1 41 74 198 76 134 75 172 23 42 146 116 3 1

Route 7: 1 106 181 150 55 135 25 164 178 110 13 1

Route 8: 1 29 54 138 145 58 143 43 173 88 1

Route 9: 1 148 119 61 84 200 126 46 175 9 115 167 90 1

Route 10: 1 83 125 49 124 20 108 176 12 149 89 128 1

Route 11: 1 103 34 82 121 136 36 162 10 158 51 1

Route 12: 1 133 163 109 91 182 65 64 127 1

Route 13: 1 99 38 101 194 142 87 114 18 174 85 6 1

Route 14: 1 112 186 80 130 165 35 79 170 30 122 1

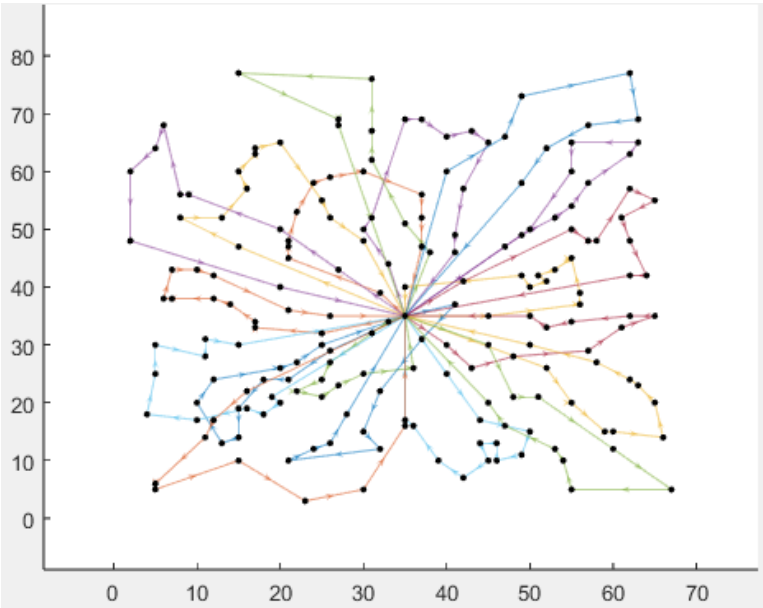
Route 15: 1 31 189 67 66 137 72 104 52 1

Route 16: 1 86 192 45 141 39 15 44 16 179 1

Route 17: 1 180 156 5 140 188 171 26 56 166 131 196 1

Route 18: 1 53 8 169 48 50 144 37 47 19 1

Route 19: 1 27 199 111 40 68 24 187 57 73 22 1



Γράφημα 30:Διαδρομές δεδομένων 5.10.3

5.10.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.2 έχουμε την λιγότερη κατανάλωση καυσίμου (1688.97lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (1557.49km) με ελάχιστο αριθμό διαδρομών (19). Παρατηρούμε πως για μεγάλο αριθμό κόμβων πλησιάζουμε στο μέγιστο των πιθανών διαδρομών-μυρμηγκιών(20) ενώ η διασπορά των κόμβων έχει μεγάλη σημασία για την κατανάλωση του καυσίμου. Η ρύθμιση του ρ βελτιστοποίησε τη λύση.

5.11

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   121
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   999999
Χρόνος εξυπηρέτησης του κόμβου:  0
Το βάρος του οχήματος χωρίς φορτίο:    6250kg
Καθαρό βάρος φόρτωσης:    12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.11.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 601.12

Iteration: 2 -- batch score: 622.48

Iteration: 3 -- batch score: 597.34

Iteration: 4 -- batch score: 628.46

Iteration: 5 -- batch score: 620.98

Iteration: 6 -- batch score: 595.70

Iteration: 7 -- batch score: 588.39

Iteration: 8 -- batch score: 590.73

Iteration: 9 -- batch score: 597.78

Iteration: 10 -- batch score: 582.23

best score: 582.23

Total Tour distance: 1221.00

Βέλτιστη διαδρομή:

Route 1: 1 120 1

Route 2: 1 88 112 86 113 19 119 7 8 10 11 5 4 2 3 118 82 83 1

Route 3: 1 18 17 20 26 23 25 28 31 34 35 37 32 29 33 36 30 27 24 21 22 115 1

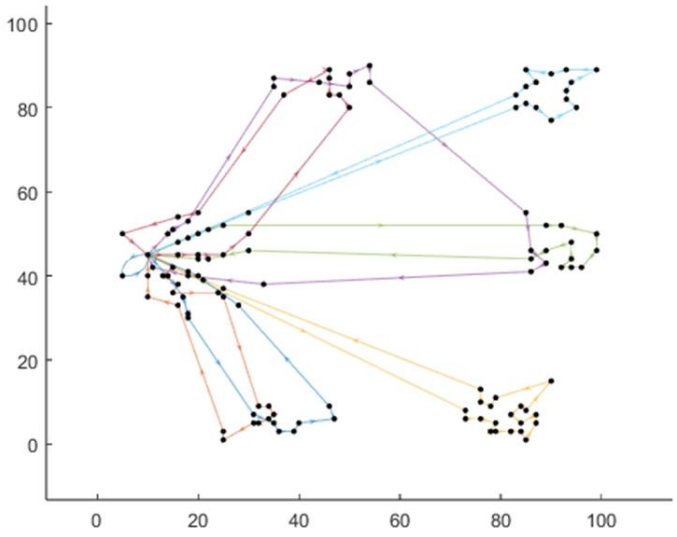
Route 4: 1 106 107 105 70 71 72 78 79 81 80 41 40 42 38 110 90 89 1

Route 5: 1 103 102 100 101 117 44 46 52 51 50 47 45 48 49 43 39 116 95 94 1

Route 6: 1 99 53 55 58 60 66 62 63 65 67 64 61 57 59 56 54 1

Route 7: 1 96 97 98 111 69 77 74 75 73 76 68 104 108 121 1

Route 8: 1 87 85 114 84 6 12 16 15 14 13 9 109 91 92 93 1



Γράφημα 31:Διαδρομές δεδομένων 5.11.1

5.11.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 518.96

Iteration: 2 -- batch score: 542.63

Iteration: 3 -- batch score: 538.12

Iteration: 4 -- batch score: 539.42

Iteration: 5 -- batch score: 513.17

Iteration: 6 -- batch score: 527.89

Iteration: 7 -- batch score: 527.72

Iteration: 8 -- batch score: 507.50

Iteration: 9 -- batch score: 534.10

Iteration: 10 -- batch score: 529.00

best score: 507.50

Total Tour distance: 1064.63

Βέλτιστη διαδρομή:

Route 1: 1 83 112 87 93 90 92 91 115 119 110 116 98 95 94 96 1

Route 2: 1 88 86 113 85 114 84 118 82 120 1

Route 3: 1 121 70 71 72 75 73 76 79 81 80 78 69 77 74 68 1

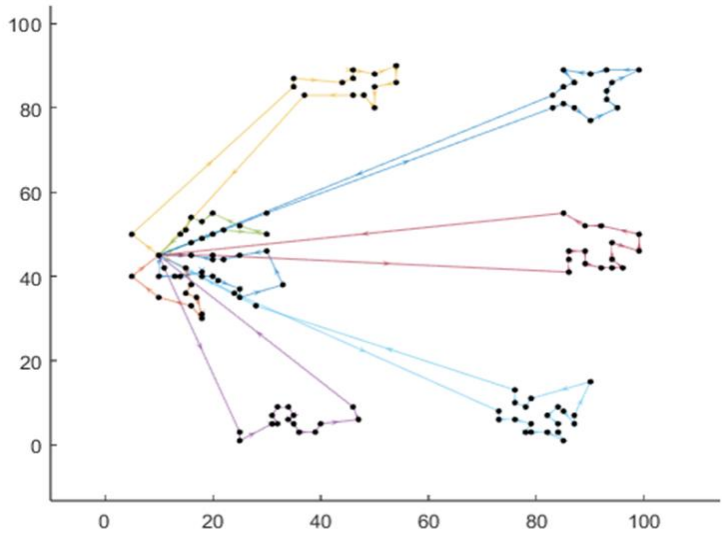
Route 4: 1 89 3 2 4 5 6 7 8 10 11 12 16 15 14 13 9 1

Route 5: 1 103 102 100 101 111 117 104 105 108 107 106 1

Route 6: 1 19 18 17 20 26 23 25 28 34 31 32 29 33 36 35 37 30 27 24 21 22 109 1

Route 7: 1 97 38 39 40 43 42 45 47 50 48 49 51 52 46 44 41 1

Route 8: 1 99 53 55 58 60 66 62 63 65 67 64 61 57 59 56 54 1



Γράφημα 32:Διαδρομές δεδομένων 5.11.2

5.11.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 518.96

Iteration: 2 -- batch score: 544.18

Iteration: 3 -- batch score: 522.22

Iteration: 4 -- batch score: 525.30

Iteration: 5 -- batch score: 537.62

Iteration: 6 -- batch score: 505.73

Iteration: 7 -- batch score: 544.43

Iteration: 8 -- batch score: 532.90

Iteration: 9 -- batch score: 528.33

Iteration: 10 -- batch score: 529.38

best score: 505.73

Total Tour distance: 1061.12

Βέλτιστη διαδρομή:

Route 1: 1 83 112 87 86 113 85 114 84 118 82 120 1

Route 2: 1 103 102 100 101 104 105 108 107 106 121 1

Route 3: 1 115 18 17 20 23 25 28 31 34 35 37 30 33 36 32 29 26 24 27 21 22 1

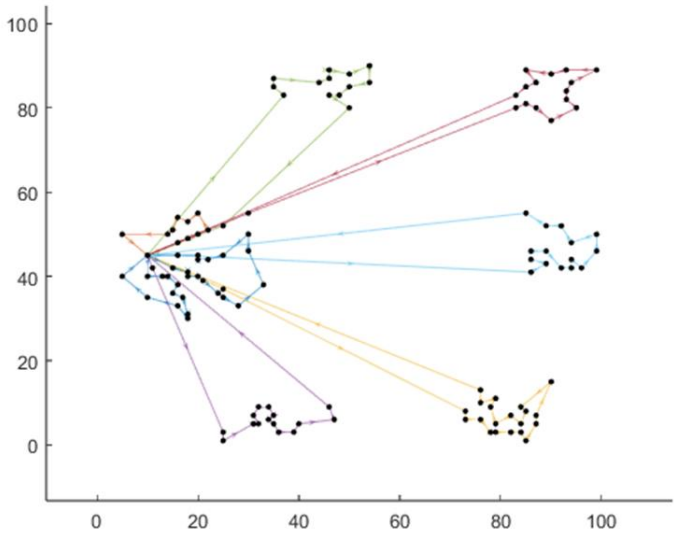
Route 4: 1 89 3 2 4 5 6 7 8 10 11 12 16 15 14 13 9 1

Route 5: 1 68 70 71 72 75 73 76 79 81 80 78 77 74 69 117 1

Route 6: 1 96 38 42 39 40 43 45 47 48 50 51 52 49 46 44 41 1

Route 7: 1 99 53 55 58 60 66 62 63 65 67 64 61 57 59 56 54 1

Route 8: 1 88 93 90 92 91 19 119 109 110 116 111 98 95 94 97 1



Γράφημα 33:Διαδρομές δεδομένων 5.11.3

5.11.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.2 έχουμε την λιγότερη κατανάλωση καυσίμου (505.73lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (1061.12km) με ελάχιστο αριθμό διαδρομών (8). Σημαντική βελτίωση είχαμε με την ελαχιστοποίηση του α (από 1 σε 0.2) και μεγιστοποίηση του β (από 1 σε 10). Η ρύθμιση του ρ βελτιστοποίησε τη λύση.

5.12

```
Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών:   101
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   999999
```

```
χρόνος εξυπηρέτησης του κόμβου:  0
Το βάρος του οχήματος χωρίς φορτίο:    6250kg
Το καθαρό βάρος φόρτωσης: 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.12.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 449.23

Iteration: 2 -- batch score: 454.19

Iteration: 3 -- batch score: 445.39

Iteration: 4 -- batch score: 453.60

Iteration: 5 -- batch score: 459.82

Iteration: 6 -- batch score: 446.43

Iteration: 7 -- batch score: 446.16

Iteration: 8 -- batch score: 424.31

Iteration: 9 -- batch score: 446.48

Iteration: 10 -- batch score: 460.03

best score: 424.31

Total Tour distance: 890.52

Βέλτιστη διαδρομή:

Route 1: 1 64 81 80 78 74 71 72 77 79 82 1

Route 2: 1 68 66 75 73 58 65 62 63 67 1

Route 3: 1 33 32 36 34 38 39 40 37 35 1

Route 4: 1 22 23 26 27 12 10 8 4 6 1

Route 5: 1 99 97 96 95 93 94 98 101 100 1

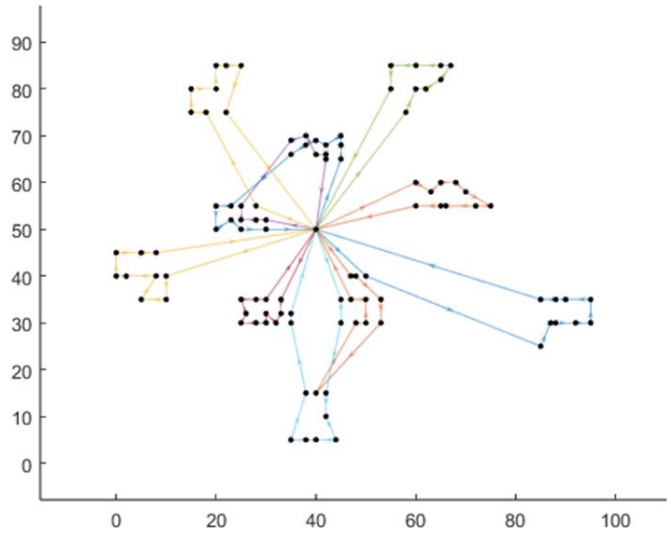
Route 6: 1 70 69 56 55 54 57 59 61 60 41 42 1

Route 7: 1 44 43 45 47 46 49 52 51 53 50 48 1

Route 8: 1 76 2 3 5 7 9 11 29 31 30 28 25 21 1

Route 9: 1 92 90 89 86 85 83 84 87 88 91 1

Route 10: 1 24 18 19 20 16 17 15 13 14 1



Γράφημα 34:Διαδρομές δεδομένων 5.12.1

5.12.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 419.68

Iteration: 2 -- batch score: 416.16

Iteration: 3 -- batch score: 399.82

Iteration: 4 -- batch score: 394.80

Iteration: 5 -- batch score: 418.74

Iteration: 6 -- batch score: 411.55

Iteration: 7 -- batch score: 409.54

Iteration: 8 -- batch score: 400.13

Iteration: 9 -- batch score: 398.95

Iteration: 10 -- batch score: 422.38

best score: 394.80

Total Tour distance: 828.54

Βέλτιστη διαδρομή:

Route 1: 1 76 2 3 5 4 8 7 9 10 12 1

Route 2: 1 21 23 25 26 28 30 31 29 27 24 22 1

Route 3: 1 68 66 63 75 73 62 65 69 67 70 1

Route 4: 1 44 43 42 41 45 47 46 49 52 51 53 50 48 1

Route 5: 1 92 90 89 86 85 83 84 87 88 91 1

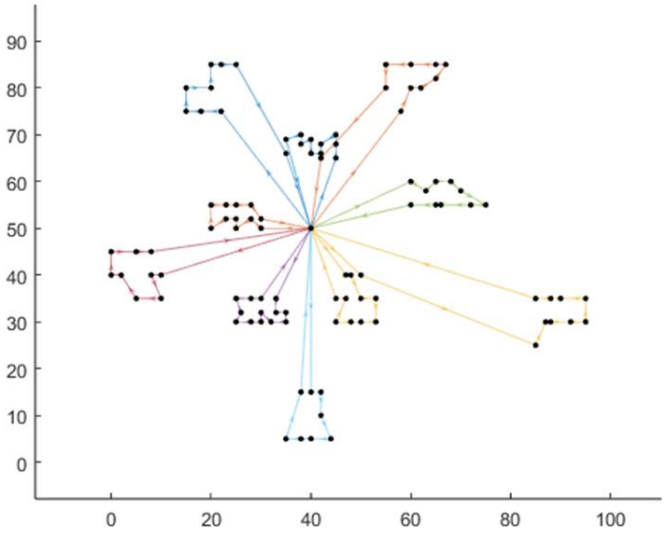
Route 6: 1 58 56 55 54 57 59 61 60 1

Route 7: 1 33 34 32 36 38 39 40 37 35 1

Route 8: 1 14 18 19 20 16 17 15 13 11 1

Route 9: 1 99 97 96 95 93 94 98 101 100 6 1

Route 10: 1 64 81 80 78 74 71 72 77 79 82 1



Γράφημα 35:Διαδρομές δεδομένων 5.12.2

5.12.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 419.68

Iteration: 2 -- batch score: 416.16

Iteration: 3 -- batch score: 395.07

Iteration: 4 -- batch score: 409.90

Iteration: 5 -- batch score: 408.95

Iteration: 6 -- batch score: 422.48

Iteration: 7 -- batch score: 394.67

Iteration: 8 -- batch score: 404.79

Iteration: 9 -- batch score: 405.55

Iteration: 10 -- batch score: 408.20

best score: 394.67

Total Tour distance: 828.15

Βέλτιστη διαδρομή:

Route 1: 1 22 23 24 27 29 31 30 28 26 25 21 1

Route 2: 1 68 66 64 63 75 73 62 65 69 67 70 1

Route 3: 1 76 2 5 4 6 8 7 10 12 11 1

Route 4: 1 44 42 41 43 45 47 46 49 52 51 53 50 48 1

Route 5: 1 91 88 87 86 89 90 92 1

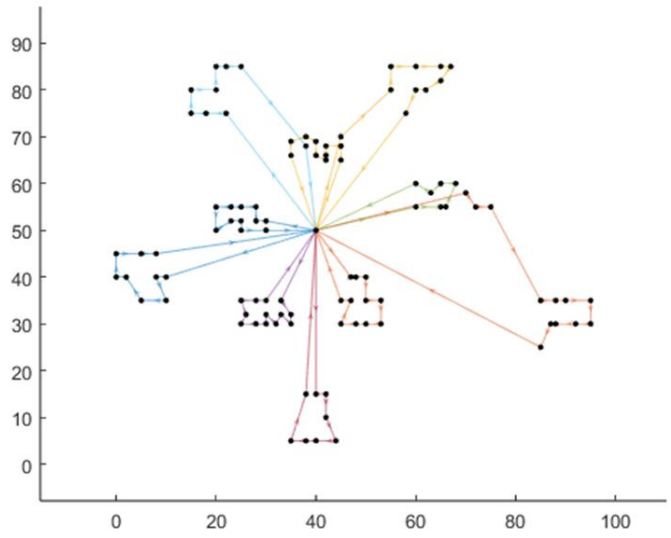
Route 6: 1 14 18 19 20 16 17 15 13 9 1

Route 7: 1 58 56 55 54 57 59 61 60 1

Route 8: 1 33 34 32 36 38 39 40 37 35 1

Route 9: 1 85 84 83 82 79 77 72 71 74 78 80 81 1

Route 10: 1 3 100 101 98 94 93 95 96 97 99 1



Γράφημα 36:Διαδρομές δεδομένων 5.12.3

5.12.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.2 έχουμε την λιγότερη κατανάλωση καυσίμου (394.67lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (828.15km) με ελάχιστο αριθμό διαδρομών (10). Η ρύθμιση του ρ μίκρυνε ελάχιστα την απόσταση και την κατανάλωση χωρίς να αλλάζει το σύνολο των διαδρομών.

5.13

Τα δεδομένα του προβλήματος είναι:

Αριθμός κόμβων-πελατών: 121
Χωρητικότητα οχήματος: 200
Χρονικό όριο διαδρομής: 720
χρόνος εξυπηρέτησης του κόμβου: 50
Το βάρος του οχήματος χωρίς φορτίο: 6250kg
Το καθαρό βάρος φόρτωσης: 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.

5.13.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 3635.91

Iteration: 2 -- batch score: 3608.81

Iteration: 3 -- batch score: 3631.96

Iteration: 4 -- batch score: 3633.74

Iteration: 5 -- batch score: 3633.92

Iteration: 6 -- batch score: 3639.32

Iteration: 7 -- batch score: 3644.10

Iteration: 8 -- batch score: 3638.05

Iteration: 9 -- batch score: 3618.93

Iteration: 10 -- batch score: 3636.16

best score: 3608.81

Total Tour distance: 1608.39

Βέλτιστη διαδρομή:

Route 1: 1 96 97 98 95 94 87 112 1

Route 2: 1 120 82 84 3 2 4 5 6 7 114 118 89 1

Route 3: 1 103 102 100 101 104 105 108 107 106 121 1

Route 4: 1 90 115 110 38 45 47 50 48 42 39 1

Route 5: 1 117 99 58 60 66 62 57 81 80 69 1

Route 6: 1 54 56 59 61 64 67 65 63 55 53 1

Route 7: 1 70 71 68 72 75 76 73 79 78 77 74 1

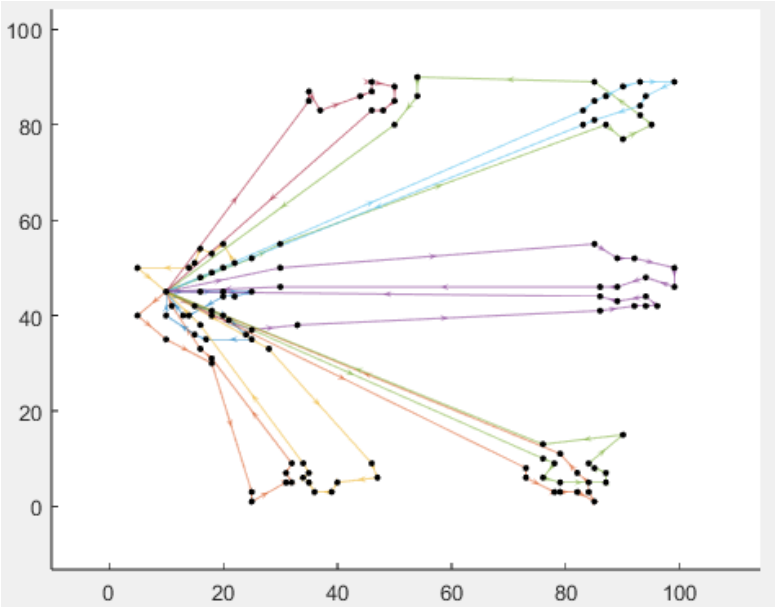
Route 8: 1 88 93 92 91 19 119 85 113 83 1

Route 9: 1 18 17 23 25 28 34 31 32 29 27 1

Route 10: 1 109 9 13 14 15 16 12 11 10 8 86 1

Route 11: 1 111 41 44 46 52 51 49 43 40 116 1

Route 12: 1 21 24 20 26 35 37 36 33 30 22 1



Γράφημα 37:Διαδρομές δεδομένων 5.13.1

5.13.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 3606.38

Iteration: 2 -- batch score: 3612.88

Iteration: 3 -- batch score: 3625.06

Iteration: 4 -- batch score: 3624.90

Iteration: 5 -- batch score: 3629.38

Iteration: 6 -- batch score: 3609.87

Iteration: 7 -- batch score: 3610.04

Iteration: 8 -- batch score: 3625.37

Iteration: 9 -- batch score: 3606.09

Iteration: 10 -- batch score: 3622.77

best score: 3606.09

Total Tour distance: 1599.93

Βέλτιστη διαδρομή:

Route 1: 1 87 88 93 95 98 116 111 117 97 1

Route 2: 1 103 107 106 1

Route 3: 1 121 108 70 71 76 73 75 72 68 104 105 1

Route 4: 1 120 83 112 90 92 91 115 19 119 109 110 94 96 1

Route 5: 1 9 13 14 15 16 12 5 4 2 3 82 1

Route 6: 1 86 85 6 11 10 8 7 84 114 118 113 89 1

Route 7: 1 54 56 59 62 63 65 67 64 61 57 1

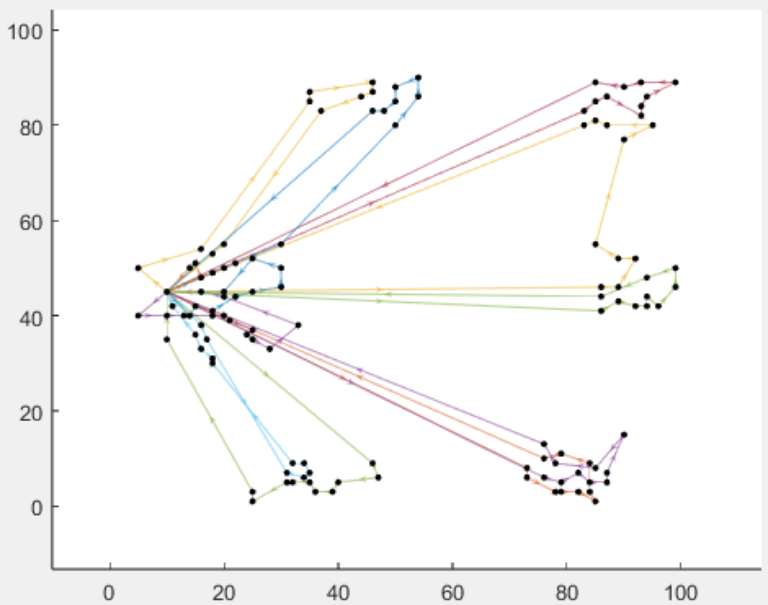
Route 8: 1 102 100 101 99 69 80 81 79 78 77 74 1

Route 9: 1 18 17 23 25 28 34 31 33 27 21 1

Route 10: 1 40 43 46 44 41 60 66 58 55 53 1

Route 11: 1 20 26 29 32 35 37 30 36 24 22 1

Route 12: 1 38 42 45 47 48 50 51 52 49 39 1



Γράφημα 38:Διαδρομές δεδομένων 5.13.2

5.13.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 3606.38

Iteration: 2 -- batch score: 3612.88

Iteration: 3 -- batch score: 3625.06

Iteration: 4 -- batch score: 3614.97

Iteration: 5 -- batch score: 3612.78

Iteration: 6 -- batch score: 3614.39

Iteration: 7 -- batch score: 3622.83

Iteration: 8 -- batch score: 3615.64

Iteration: 9 -- batch score: 3620.66

Iteration: 10 -- batch score: 3614.39

best score: 3606.38

Total Tour distance: 1602.67

Βέλτιστη διαδρομή:

Route 1: 1 96 97 94 95 98 116 111 101 106 121 1

Route 2: 1 88 90 93 1

Route 3: 1 120 83 113 85 19 115 91 86 87 112 89 1

Route 4: 1 82 3 2 4 5 11 10 8 7 84 114 118 1

Route 5: 1 17 27 26 23 25 28 32 29 30 22 1

Route 6: 1 70 71 68 72 75 76 73 79 81 80 78 1

Route 7: 1 38 42 45 47 50 51 52 49 48 39 1

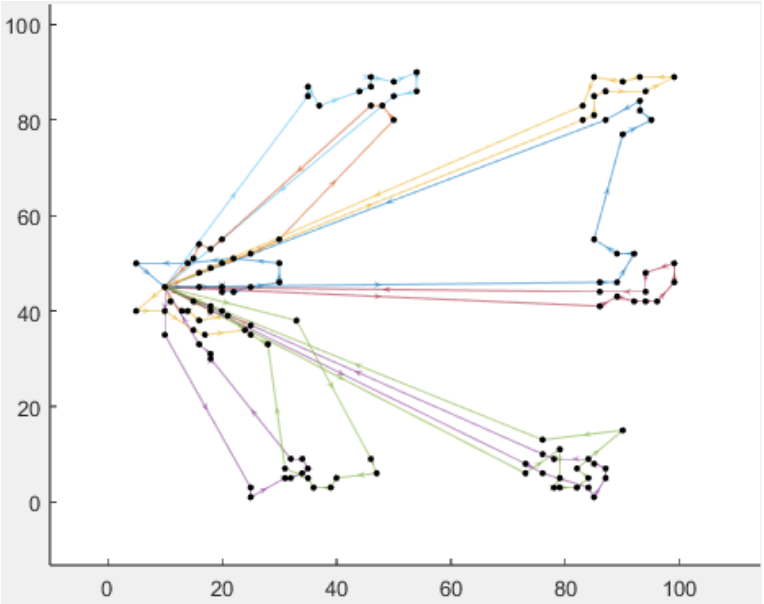
Route 8: 1 40 43 46 44 41 60 66 62 63 58 1

Route 9: 1 103 102 100 117 99 69 77 74 104 105 108 107 1

Route 10: 1 53 55 56 59 65 67 64 61 57 54 1

Route 11: 1 18 20 31 34 35 37 36 33 24 21 1

Route 12: 1 110 9 13 14 15 16 12 6 109 119 92 1



Γράφημα 39:Διαδρομές δεδομένων 5.13.3

5.13.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=0.2, β=10 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (3606.09lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (1599.93km) με ελάχιστο αριθμό διαδρομών (12). Παρατηρούμε πως η μεγάλη διασπορά των κόμβων έχει μεγάλη σημασία για την κατανάλωση του καυσίμου καθώς καλύπτεται μεγάλη απόσταση ενώ το όχημα είναι γεμάτο ώστε να εξυπηρετήσει «ομάδες» πελατών που βρίσκονται σε μικρή απόσταση μεταξύ τους. Αρκεί να παρατηρήσουμε τα γραφήματα των παραπάνω περιπτώσεων. Η ρύθμιση του ρ δεν βελτιστοποίησε τη λύση.

5.14

```
Τα δεδομένα του προβλήματος είναι

Αριθμός κόμβων-πελατών:   101
Χωρητικότητα οχήματος:    200
Χρονικό όριο διαδρομής:   1040
χρόνος εξυπηρέτησης του κόμβου:  90
Το βάρος οχήματος χωρίς φορτίο: 6250kg
Το καθαρό βάρος φόρτωσης: 12500kg
Και η ζήτηση κάθε κόμβου

Οι συντεταγμένες κάθε κόμβου-πελάτη απεικονίζονται με γράφημα στο αποτέλεσμα.
```

5.14.1 α=1 β=1 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 1 b = 1 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 4742.66

Iteration: 2 -- batch score: 4725.79

Iteration: 3 -- batch score: 4719.61

Iteration: 4 -- batch score: 4739.48

Iteration: 5 -- batch score: 4723.10

Iteration: 6 -- batch score: 4734.44

Iteration: 7 -- batch score: 4722.57

Iteration: 8 -- batch score: 4737.75

Iteration: 9 -- batch score: 4742.12

Iteration: 10 -- batch score: 4735.72

best score: 4719.61

Total Tour distance: 910.77

Βέλτιστη διαδρομή:

Route 1: 1 9 10 14 16 20 17 15 13 12 11 1

Route 2: 1 90 89 86 85 83 84 87 88 91 1

Route 3: 1 24 18 19 31 29 27 28 53 50 48 1

Route 4: 1 58 60 61 59 57 54 55 56 69 70 1

Route 5: 1 100 101 98 94 93 95 96 97 99 92 1

Route 6: 1 30 33 34 32 36 38 39 40 37 35 1

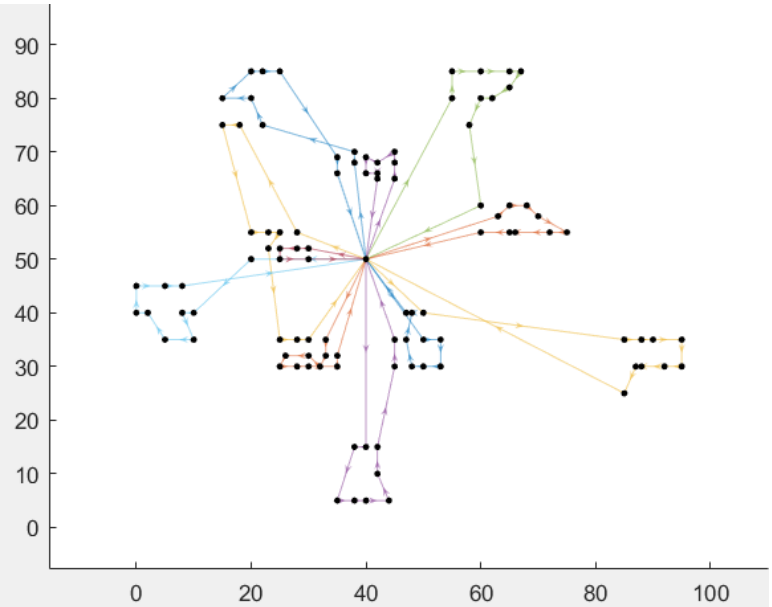
Route 7: 1 22 23 26 25 21 1

Route 8: 1 63 75 73 62 65 67 66 68 1

Route 9: 1 44 43 45 47 51 52 49 46 41 42 1

Route 10: 1 64 82 79 77 72 71 74 78 80 81 1

Route 11: 1 76 2 3 5 7 8 4 6 1



Γράφημα 40:Διαδρομές δεδομένων 5.14.1

5.14.2 α=0.2 β=10 ρ=0,4

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.4

Επαναλήψεις:

Iteration: 1 -- batch score: 4743.13

Iteration: 2 -- batch score: 4742.08

Iteration: 3 -- batch score: 4738.12

Iteration: 4 -- batch score: 4749.95

Iteration: 5 -- batch score: 4733.36

Iteration: 6 -- batch score: 4729.39

Iteration: 7 -- batch score: 4733.66

Iteration: 8 -- batch score: 4736.61

Iteration: 9 -- batch score: 4734.06

Iteration: 10 -- batch score: 4745.89

best score: 4729.39

Total Tour distance: 941.88

Βέλτιστη διαδρομή:

Route 1: 1 14 18 16 13 15 17 20 19 24 1

Route 2: 1 22 23 26 27 29 31 30 28 25 21 1

Route 3: 1 48 47 43 45 46 49 52 51 53 50 1

Route 4: 1 68 66 63 67 70 1

Route 5: 1 76 2 3 100 101 5 4 6 1

Route 6: 1 92 90 89 86 85 83 84 87 88 91 1

Route 7: 1 8 7 9 1

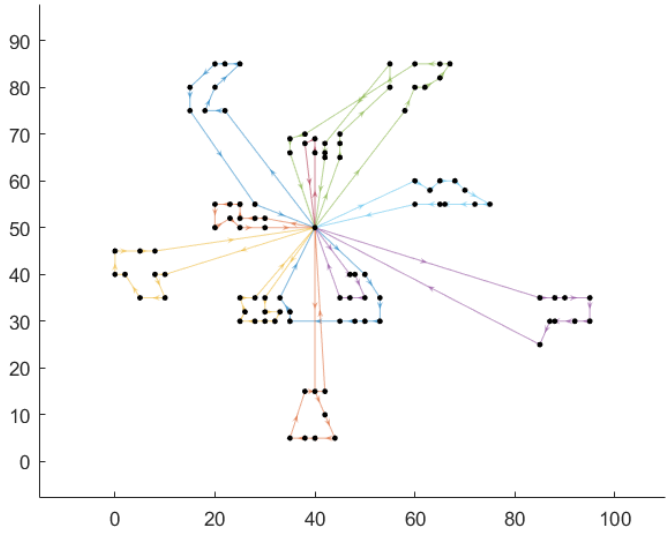
Route 8: 1 64 75 73 62 65 69 41 42 44 1

Route 9: 1 58 55 54 57 59 61 60 56 1

Route 10: 1 33 34 32 36 38 39 40 37 35 1

Route 11: 1 82 79 77 72 71 74 78 80 81 1

Route 12: 1 99 97 96 95 93 94 98 10 12 11 1



Γράφημα 41:Διαδρομές δεδομένων 5.14.2

5.14.3 α=0.2 β=10 ρ=0,2

Η υλοποίηση του αλγορίθμου γίνεται για τιμές:

a= 0.2 b = 10 ρ = 0.2

Επαναλήψεις:

Iteration: 1 -- batch score: 4726.15

Iteration: 2 -- batch score: 4755.84

Iteration: 3 -- batch score: 4738.95

Iteration: 4 -- batch score: 4743.64

Iteration: 5 -- batch score: 4737.78

Iteration: 6 -- batch score: 4733.95

Iteration: 7 -- batch score: 4745.88

Iteration: 8 -- batch score: 4738.23

Iteration: 9 -- batch score: 4760.39

Iteration: 10 -- batch score: 4758.84

best score: 4726.15

Total Tour distance: 928.52

Βέλτιστη διαδρομή:

Route 1: 1 8 7 3 2 5 4 6 1

Route 2: 1 35 37 34 33 53 50 48 1

Route 3: 1 22 23 29 31 19 20 18 27 24 1

Route 4: 1 44 43 45 47 51 52 49 46 41 42 1

Route 5: 1 66 64 75 63 62 65 67 68 1

Route 6: 1 14 16 17 15 13 10 9 12 11 1

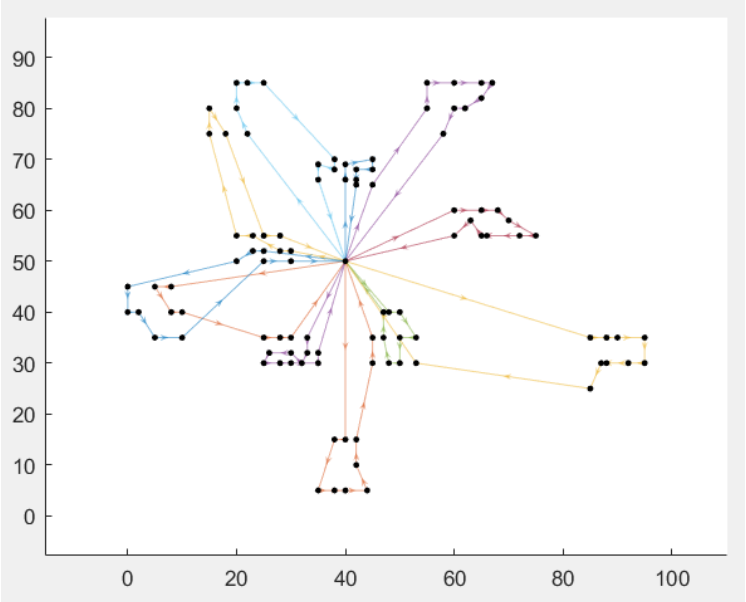
Route 7: 1 92 89 86 85 83 84 87 88 90 91 1

Route 8: 1 26 28 30 40 39 38 36 32 25 21 1

Route 9: 1 58 60 61 59 57 54 55 56 69 70 1

Route 10: 1 82 79 77 72 71 74 78 80 81 73 1

Route 11: 1 76 100 101 98 94 93 95 96 97 99 1



Γράφημα 42:Διαδρομές δεδομένων 5.14.3

5.14.4 Σύγκριση αποτελεσμάτων

Παρατηρήθηκε πως για α=1, β=1 και ρ=0.4 έχουμε την λιγότερη κατανάλωση καυσίμου (4719.61lt) καθώς επίσης επιτυγχάνεται η μικρότερη συνολική χιλιομετρική απόσταση (910.77km) με ελάχιστο αριθμό διαδρομών (11). Παρατηρούμε πως η μεγάλη διασπορά των κόμβων έχει μεγάλη σημασία για την κατανάλωση του καυσίμου καθώς καλύπτεται μεγάλη απόσταση ενώ το όχημα είναι γεμάτο ώστε να εξυπηρετήσει «ομάδες» πελατών που βρίσκονται σε μικρή απόσταση μεταξύ τους. Αρκεί να παρατηρήσουμε τα γραφήματα των παραπάνω περιπτώσεων. Η ρύθμιση του ρ συνέβαλε στην βελτίωση της χιλιομετρικής απόστασης και στον αριθμό των διαδρομών και λιγότερο στην συνολική κατανάλωση καυσίμου.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτή την εργασία, διερευνήσαμε την εφαρμογή του αλγόριθμου Ant Colony Optimization (ACO) στο Κλειστό Πρόβλημα Δρομολόγησης Πράσινων Οχημάτων (G-CVRP) με περιορισμούς χωρητικότητας. Στόχος μας ήταν να ελαχιστοποιήσουμε το συνολικό κόστος, λαμβάνοντας υπόψη την κατανάλωση καυσίμου, την απόσταση των διαδρομών και τον χρόνο εξυπηρέτησης του συστήματος. Το μοντέλο που αναπτύχθηκε ενσωματώνει λειτουργίες ρεαλιστικού ρυθμού κατανάλωσης καυσίμου (FCR), οι οποίες προσαρμόζονται δυναμικά με βάση το φορτίο του οχήματος.

Μέσω εκτεταμένων δοκιμών, αναλύσαμε τον αντίκτυπο των βασικών παραμέτρων ACO - σημασία φερομόνης (α), ευρετικός παράγοντας (β) και ρυθμός εξάτμισης (ρ) - στην ποιότητα της λύσης και την ταχύτητα σύγκλισης. Τα αποτελέσματα έδειξαν ότι μια ισορροπημένη διαμόρφωση παραμέτρων επηρεάζει σημαντικά την αποτελεσματικότητα του αλγορίθμου. Οι υψηλότερες τιμές του α βελτίωσαν τη σύγκλιση αλλά κινδύνευαν με πρόωρη στασιμότητα, ενώ η αύξηση του β οδήγησε σε πιο ντετερμινιστική συμπεριφορά, ευνοώντας μικρότερες διαδρομές. Ο ρυθμός εξάτμισης (ρ) έπαιξε κρίσιμο ρόλο στη διατήρηση της ποικιλομορφίας και στην αποφυγή τοπικών βέλτιστων.

Συνολικά, η προσέγγισή μας βελτιστοποίησε με επιτυχία την «πράσινη» δρομολόγηση των οχημάτων ενσωματώνοντας ζητήματα σχετικά με την απόδοση καυσίμου, συμβάλλοντας στον βιώσιμο σχεδιασμό της εφοδιαστικής. Μελλοντικές εργασίες θα μπορούσαν να διερευνήσουν υβριδικούς μεθευρετικούς αλγορίθμους, περιορισμούς κυκλοφορίας στον πραγματικό κόσμο και εναλλακτικά μοντέλα FCR για περαιτέρω βελτίωση της απόδοσης της δρομολόγησης.

7.ΒΙΒΛΙΟΓΡΑΦΙΑ

Canhong Lin, K.L. Choy , G.T.S. Ho, S.H. Chung, H.Y. Lam «Survey of Green Vehicle Routing Problem: Past and future trends»

Reza Moghdani , Khodakaram Salimifard , Emrah Demir, Abdelkader Benyettou «The green vehicle routing problem: A systematic literature review»

W.F. Tan, Lai Soon Lee, Zanariah Abdul Majid, Hsin-Vonn Seow «Ant Colony Optimization for Capacitated Vehicle Routing Problem»,

Xiao, Y., Zhao, Q., Kaku, I., & Xu, Y. (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. Computers & operations research, 39(7), 1419-1431.

Ιωάννης Μαρινάκης, Μαγδαληνή Μαρινάκη, Αθανάσιος Μυγδαλάς «Προβλήματα Δρομολόγησης Οχημάτων στη Διαχείριση της Εφοδιαστικής Αλυσίδας»,

https://el.wikiversity.org/wiki/Εφοδιαστική_Αλυσίδα_Σχεδιασμού_Συστημάτων#Ορισμός_της_Διαχείρισης_Εφοδιαστικής_Αλυσίδας_-_Supply_Chain_Management

ΕΙΚΟΝΕΣ

ΣΚΕΥΟΦΥΛΑΞ ΠΑΝΑΓΙΩΤΗΣ «Εξελικτικοί αλγόριθμοι για το ανοικτό – κλειστό πρόβλημα δρομολόγησης οχημάτων.»

https://www.researchgate.net/figure/An-example-of-the-2-opt-algorithm_fig2_353479905

https://www.researchgate.net/figure/The-1-1-exchange-move-for-single-and-multiple-routes_fig3_257551274