ΕΠΙΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΧΩΡΟΘΕΤΗΣΗΣ ΕΓΚΑΤΑΣΤΑΣΕΩΝ ΚΑΙ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΜΙΜΗΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ

Μανινάκης Ανδρέας[1]
Πολυτεχνείο Κρήτης
Τμήμα Μηχανικών Παραγωγής και Διοίκησης
†
Επιβλέπων καθηγητής:
Δρ. Ιωάννης Μαρινάκης,[2] Λέκτορας
Πολυτεχνείο Κρήτης
Τμήμα Μηχανικών Παραγωγής και Διοίκησης
Εργαστήριο Συστημάτων Υποστήριξης Αποφάσεων

Απρίλιος 2011

[1]email: amaninak@gmail.com
[2]Πολυτεχνειούπολη, Χανιά Κρήτης ΤΚ 73100, Τηλ: 2821037288, email: marinakis@ergasya.tuc.gr

Περίληψη

Στόχος αυτής της πτυχιακής εργασίας είναι η μελέτη και η επίλυση του σύνθετου προβλήματος χωροθέτησης εγκαταστάσεων και δρομολόγησης οχημάτων (LRP) με τη χρήση γενετικών και μιμητικών αλγορίθμων. Το πρόβλημα αυτό περιγράφει την περίπτωση όπου η ζήτηση γεωγραφικά κατανεμημένων πελατών πρέπει να ικανοποιηθεί με προϊόντα που διανέμονται από οχήματα διανομής. Τα οχήματα αυτά μπορούν να ξεκινούν από διάφορες διαθέσιμες τοποθεσίες για την εγκατάσταση αποθηκών ή παραγωγικών κέντρων. Η λύση πρέπει να απαντά στο που πρέπει να εγκατασταθούν τα κέντρα διανομής και ποιες θα πρέπει να είναι οι διαδρομές που θα ακολουθήσουν τα οχήματα, έτσι ώστε να εξυπηρετήσουν όλους τους πελάτες με το ελάχιστο δυνατό κόστος. Δεδομένου της πολυπλοκότητας του προβλήματος, οι κλασσικές μέθοδοι και στρατηγικές που εφαρμόζονται συχνά στην επιχειρησιακή έρευνα αποδεικνύονται μη αποδοτικές και συχνά καταλήγουν σε τοπικό ελάχιστο, το οποίο μπορεί να απέχει σημαντικά από την βέλτιστη λύση. Για αυτό το λόγο επιλέχθηκε η χρήση της μεθευρετικής εξελικτικής στρατηγικής των μιμητικών αλγορίθμων για την επίλυση του προβλήματος. Μιμούμενοι τη διαδικασία της εξέλιξης στη φύση, όπου αναπτύσσονται νέοι και καλύτεροι πληθυσμοί γενιά προς γενιά, δημιουργούμε πληθυσμούς λύσεων που διασταυρώνονται μεταξύ τους και μεταλλάσσονται με στόχο τη δημιουργία νέων, καλύτερων γενεών. Για την επιτάχυνση της διαδικασίας μετάβασης σε νέες καλύτερες λύσεις σε κάθε γενιά χρησιμοποιήθηκε η ευρετική μέθοδος της τοπικής αναζήτησης 2-Opt. Λόγω της φύσης του προβλήματος χωροθέτησης εγκαταστάσεων και δρομολόγησης οχημάτων μελετήθηκε τόσο η επίλυση των δύο φάσεων (χωροθέτησης και δρομολόγησης) ξεχωριστά, καθώς και η ταυτόχρονη επίλυση τους. Η εργασία αυτή αφορά περιπτώσεις όπου εξετάζεται η μείωση του κόστους διανομής με τη χρήση νέων αποθηκών και πως θα διαμορφωθεί το νέο δίκτυο διανομής ή η δημιουργία νέων παραγωγικών κέντρων και ποια ή ποιες θα ήταν οι βέλτιστες τοποθεσίες για να εξυπηρετηθούν με το ελάχιστο κόστος οι πελάτες της επιχείρησης. Δεδομένου ότι συχνά ξοδεύεται περισσότερο από το 20% των εσόδων από τις πωλήσεις στο φυσικό δίκτυο διανομής, η σημασία αυτής της εργασίας είναι αρκετά μεγάλη.

Περιεχόμενα

Κεφάλαιο 1

Το πρόβλημα χωροθέτησης εγκαταστάσεων και δρομολόγησης οχημάτων (LRP)

1.1 Γενική περιγραφή

1.1.1 Γενικά

Η επιλογή των τοποθεσιών για τις εγκαταστάσεις είναι μια από τις πιο κρίσιμες διοικητικές αποφάσεις. Το κόστος του συστήματος διανομής και το επίπεδο της εξυπηρέτησης πελατών που παρέχονται από το σύστημα επηρεάζονται σημαντικά από το πλήθος, το μέγεθος και τις τοποθεσίες κάθε κέντρου διανομής. Συμπεριλαμβάνοντας και τα κανάλια φυσικής διανομής, τα οποία έχουν σαν στόχο την παράδοση των προϊόντων σε καλή κατάσταση και τη σωστή χρονική στιγμή, προκύπτουν τα έξοδα μεταφοράς τα οποία αποτελούν συνήθως το μεγαλύτερο μέρος των εξόδων της εφοδιαστικής αλυσίδας[1]. Συνήθως οι αποφάσεις που αφορούν την τοποθεσία γίνονται σε στρατηγικό επίπεδο, ενώ οι αποφάσεις δρομολόγησης γίνονται επιχειρησιακά[4]. Μπορούμε να συμπεράνουμε ότι η μείωση των συνολικών εξόδων είναι κρίσιμη για τα οικονομικά μιας επιχείρησης και ταυτόχρονα μπορεί να είναι μια ιδιαίτερα πολύπλοκη διαδικασία. Η σύνδεση μεταξύ της τοποθεσίας των εγκαταστάσεων και της δρομολόγησης οχημάτων είναι γενικά αποδεκτή, ενώ ο διαχωρισμός τους αποδεικνύεται ότι ενδέχεται να οδηγήσει σε υποβέλτιστες λύσεις[2]. Αυτό θα το δούμε και στη συνέχεια αυτής της εργασίας. Παλιότερα η υπολογιστική πολυπλοκότητα του συνδυασμού του προβλήματος δρομολόγησης οχημάτων (VRP)[1] με το πρόβλημα χωροθέτησης εγκαταστάσεων (FLP[2] ή LAP[3]) ήταν μη πρακτική λόγω των τεχνολογικών περιορισμών της εποχής. Με την έλευση νέων ευρετικών και μεθευρετικών μεθόδων και αλγορίθμων, καθώς και με τις εξελίξεις της τεχνολογίας στον χώρο των ηλεκτρονικών υπολογιστών, η μελέτη και η επίλυση του προβλήματος χωροθέτησης και δρομολόγησης είναι πλέον αρκετά αποδοτική.

[1] Vehicle Routing Problem
[2] Facility Location Problem
[3] Location-Allocation Problem

Το πρόβλημα χωροθέτησης εγκαταστάσεων και δρομολόγησης οχημάτων (LRP) περιγράφει την περίπτωση όπου υπάρχουν διάφορες πιθανές τοποθεσίες για αποθήκες, οι οποίες εξυπηρετούν μέσω οχημάτων πελάτες που βρίσκονται γεωγραφικά διασκορπισμένοι. Θα πρέπει να βρεθούν οι βέλτιστες διαδρομές, τις οποίες θα ακολουθούν τα οχήματα που θα εξυπηρετούν τους πελάτες ικανοποιώντας τη ζήτηση τους. Ταυτόχρονα θα πρέπει να βρεθούν και οι βέλτιστες τοποθεσίες για τις αποθήκες, από τις οποίες ξεκινούν τα οχήματα για να εξυπηρετήσουν τους πελάτες, έτσι ώστε το συνολικό κόστος δρομολόγησης (απόσταση, καύσιμα, χρόνος κ.α.) και χωροθέτησης (κόστος λειτουργίας αποθηκών, ενοίκιο ή αγορά κ.α.) να είναι το ελάχιστο. Το σύνολο των αποφάσεων που πρέπει να ληφθούν συνοψίζεται στις ακόλουθες:

- που πρέπει να βρίσκονται τελικά οι αποθήκες από το σύνολο διαθέσιμων τοποθεσιών.
- ποιοι πελάτες θα εξυπηρετούνται από ποιες αποθήκες.
- ποιοι πελάτες θα ανατεθούν στην εκάστοτε διαδρομή.
- ποια θα είναι η σειρά με την οποία πρέπει να εξυπηρετηθούν οι πελάτες κάθε διαδρομής.

Η λήψη των αποφάσεων που περιγράψαμε παραπάνω κάνουν το πρόβλημα πιο πολύπλοκο από το κλασικό πρόβλημα χωροθέτησης εγκαταστάσεων (FLP) και το πρόβλημα δρομολόγησης οχημάτων (VRP), των οποίων η πολυπλοκότητα είναι NP-hard. Έτσι και η πολυπλοκότητα του προβλήματος (LRP) είναι NP-hard, που αυτομάτως σημαίνει ότι η χρήση των κλασσικών μεθόδων και στρατηγικών που χρησιμοποιούνται στην επιχειρησιακή έρευνα είναι μη αποδοτική και συχνά χαρακτηρίζεται από αδυναμία περαιτέρω βελτίωσης από τοπικά ελάχιστα. Για αυτό το λόγο χρησιμοποιούνται μεθευρετικοί αλγόριθμοι, όπως οι γενετικοί και οι μιμητικοί στην περίπτωσή μας. Οι μεθευρετικοί αλγόριθμοι υλοποιούν μια φυσική διαδικασία ή φαινόμενο συνδυάζοντας διαδικασίες τοπικής αναζήτησης και στρατηγικές υψηλότερου επιπέδου. Χαρακτηρίζονται από την προσαρμοστικότητά τους και την ευκολία μεταφοράς τους σε παράλληλη μορφή.

Οι πελάτες έχουν συγκεκριμένη ζήτηση σε προϊόντα, η οποία πρέπει να ικανοποιηθεί με μία και μοναδική επίσκεψη από τα οχήματα. Τα οχήματα και οι αποθήκες έχουν συγκεκριμένη χωρητικότητα. Μια διαδρομή ξεκινά από την αποθήκη όπου το όχημα είναι γεμάτο και επισκέπτεται τους πελάτες με τη σειρά που ορίζει η συγκεκριμένη διαδρομή ικανοποιώντας τη ζήτησή τους. Όταν το όχημα αδειάσει ή η υπάρχουσα ποσότητα προϊόντων που μεταφέρει δεν αρκεί για να ικανοποιήσει τη ζήτηση του πελάτη που έχει σειρά, επιστρέφει στην αποθήκη σηματοδοτώντας τη λήξη της τρέχουσας διαδρομής και την έναρξη μιας νέας μέχρι να ικανοποιηθούν όλοι οι πελάτες που αντιστοιχούν στη συγκεκριμένη αποθήκη. Η διαδικασία επαναλαμβάνεται για κάθε αποθήκη μέχρι να εξυπηρετηθούν όλοι οι πελάτες. Οι διαδρομές δεν είναι απαραίτητο να πραγματοποιούνται η μία μετά την άλλη με σειριακό τρόπο, αλλά μπορούν να χρησιμοποιούνται περισσότερα του ενός οχήματα για να πραγματοποιούν τις ορισμένες διαδρομές ταυτόχρονα. Με αυτό τον τρόπο μειώνεται ο συνολικός χρόνος εξυπηρέτησης των πελατών, αλλά όπως είναι αναμενόμενο αυξάνεται το κόστος λειτουργίας λόγω της χρήσης παραπάνω οχημάτων που απαιτούν με τη σειρά τους συντήρηση, οδηγούς κλπ.

1.1.2 Οι δύο φάσεις του LRP

Το LRP μπορεί να χωριστεί σε δύο υπό-προβλήματα ή φάσεις. Αυτά είναι το πρόβλημα χωροθέτησης εγκαταστάσεων (FLP) και το πρόβλημα δρομολόγησης οχημάτων (VRP) που είδαμε προηγουμένως. Το υπό-πρόβλημα της χωροθέτησης έχει σαν στόχο να βρει ποιες αποθήκες πρέπει να ανοίξουν και πως πρέπει να αντιστοιχηθούν οι πελάτες σε κάθε αποθήκη. Το κόστος που πρέπει να ελαχιστοποιηθεί αφορά το σταθερό κόστος λειτουργίας των ανοιχτών εγκαταστάσεων, το μεταβλητό κόστος των εγκαταστάσεων που είναι ανάλογο των προϊόντων που διακινούνται μέσω αυτών και της απόστασης μεταξύ των πελατών και των εγκαταστάσεων στις οποίες αντιστοιχούν. Το υπό-πρόβλημα πρόβλημα της δρομολόγησης έχει σαν δεδομένο την τοποθεσία που βρίσκονται οι εγκαταστάσεις και στόχος είναι η εύρεση των βέλτιστων διαδρομών που θα πρέπει να ακολουθήσουν τα οχήματα για να επισκεφθούν τους πελάτες και να ικανοποιήσουν τη ζήτησή τους. Το κόστος που πρέπει να ελαχιστοποιηθεί εδώ αφορά την απόσταση μεταξύ πελατών και εγκαταστάσεων που βρίσκονται στην ίδια διαδρομή, το οποίο είναι ανάλογο της απόστασης που διανύεται από τα οχήματα καθώς εκτελούν την διαδρομή.

Στην περίπτωση που επιλύονται ταυτόχρονα και οι φάσεις του προβλήματος πρέπει για κάθε λύση να πάρουμε τις αποφάσεις που είδαμε παραπάνω. Στην περίπτωση που οι δύο φάσεις του προβλήματος επιλύονται ξεχωριστά βρίσκουμε πρώτα τη βέλτιστη λύση στο πρόβλημα χωροθέτησης και στη συνέχεια ξεκινάμε τον υπολογισμό της βέλτιστης λύσης του προβλήματος δρομολόγησης βασιζόμενοι στη βέλτιστη λύση που έχουμε υπολογίσει από την προηγούμενη φάση. Ωστόσο δεν είναι απαραίτητο να λυθούν τα υπό-προβλήματα με αυτή τη σειρά[7]. Στην πρώτη περίπτωση έχουμε το πλεονέκτημα ότι φτάνουμε σε ένα χαμηλό κόστος σχετικά γρήγορα, αλλά για τον περαιτέρω υπολογισμό της βέλτιστης λύσης υπάρχουν πολλοί παράγοντες που επηρεάζουν το αποτέλεσμα καθιστώντας την μετάβαση σε χαμηλότερο κόστος πιο δύσκολη σε μεγάλου μεγέθους προβλήματα. Αντίθετα στην δεύτερη περίπτωση όπου έχουμε ξεχωριστή επίλυση των δύο φάσεων, περιορίζουμε τους βαθμούς ελευθερίας του προβλήματος λύνοντας πρώτα το πρόβλημα χωροθέτησης και στη συνέχεια της δρομολόγησης. Αυτό μπορεί να βοηθήσει στην εύρεση λύσεων με χαμηλό κόστος σε πολύπλοκα προβλήματα, αλλά κατά την υλοποίηση του με μιμητικούς αλγορίθμους μπορεί να χρειαστούν αρκετές παραπάνω γενιές για την τελική λύση σε σχετικά απλά προβλήματα. Ενώ σε αυτή την περίπτωση πάντα υπάρχει ο κίνδυνος η λύση που θα βρεθεί να είναι υπό-βέλτιστη χωρίς δυνατότητα βελτίωσης, λόγω του διαχωρισμού της δρομολόγησης από τη χωροθέτηση.

1.2 Μαθηματική μοντελοποίηση

Το κόστος του προβλήματος LRP αποτελείται από το συνδυασμό του κόστους λειτουργίας και διαχείρισης των εγκαταστάσεων με το κόστος μεταφοράς των εμπορευμάτων. Το κόστος εγκαταστάσεων αποτελείται από τα σταθερά και μεταβλητά κόστη. Το σταθερό κόστος εκφράζει τα έξοδα που απαιτούνται για τη δημιουργία της εγκατάστασης (αγορά, διαμόρφωση κλπ), ενώ τα μεταβλητό κόστος τα έξοδα λειτουργίας της εγκατάστασης και αποδίδεται σαν μια μοναδιαία γραμμική ποσότητα ανά μονάδα προϊόντος που εξέρχεται από την αποθήκη. Το κόστος μεταφοράς αποτελείται από τα κόστη προμήθειας εμπορευμάτων προς τα κέντρα διανομής και τα κόστη παράδοσης στους πελάτες. Το κόστος μεταφοράς των εμπορευμάτων προς τις αποθήκες από όπου θα γίνει η παράδοση είναι γραμμικό ως προς την ποσότητα που μεταφέρεται. Το κόστος παράδοσης αφορά τα οχήματα μεταφοράς από τις εγκαταστάσεις τα οποία εκτελούν διαδρομές

με περισσότερες του ενός πελάτες. Το συνολικό κόστος παράδοσης είναι αναλογικό ως προς την απόσταση που διανύεται από τα οχήματα διανομής.

1.2.1 Γενική μορφή του προβλήματος

Η γενική μορφή του προβλήματος LRP σύμφωνα με τους *J. Perl και M. S. Daskin*[2] ορίζεται ως εξής.[4] Δίνεται η τοποθεσία και η ζήτηση ενός συνόλου n πελατών, όπου καθένας έχει ανατεθεί σε μια τοπική αποθήκη που θα τον προμηθεύει περιοδικά, σύμφωνα με τη ζήτηση του. Επίσης δίνεται και το σύνολο m πιθανών τοποθεσιών για τις εγκαταστάσεις μαζί με το σταθερά και μεταβλητά τους κόστη. Το πρόβλημα είναι το πλήθος, μέγεθος και τοποθεσίες των αποθηκών, η κατανομή των πελατών στις ανοιχτές αποθήκες και οι διαδρομές παράδοσης, έτσι ώστε να ελαχιστοποιείται το άθροισμα του κόστους μεταφοράς και του κόστους των αποθηκών.

Η αντικειμενική συνάρτηση του προβλήματος είναι:

$$\min \sum_{i=n+1}^{n+m} F_i z_i + \sum_{i=1}^{S} \sum_{j=n+1}^{n+m} c_{ij} f_{ij} + \sum_{j=n+1}^{n+m} v_j \left(\sum_{i=1}^{n} q_i y_{ij} \right) + \sum_{k=1}^{K} \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} c_k d_{ij} x_{ijk} \tag{1.1}$$

υπό τους περιορισμούς:

$$\sum_{k=1}^{K} \sum_{j=1}^{n+m} x_{ijk} = 1, i = 1, ..., n \tag{1.2}$$

$$\sum_{i=1}^{n} q_i \left(\sum_{j=1}^{n+m} x_{ijk} \right) \leq C_k, k = 1, ..., K \tag{1.3}$$

$$\sum_{i=1}^{n+m} \sum_{j=1}^{n+m} d_{ij} x_{ijk} \leq D_k, k = 1, ..., K \tag{1.4}$$

$$\sum_{i \in N} \sum_{j \in \bar{N}} \sum_{j=1}^{K} x_{ijk} \geq 1, \forall N \subset \{1, ..., n+m\} | \{n+1, ..., n+m\} \subset N \tag{1.5}$$

$$\sum_{j=1}^{n+m} x_{ijk} - \sum_{j=1}^{n+m} x_{ijk} = 0, k = 1, ..., K, i = 1, ..., n+m \tag{1.6}$$

$$\sum_{j=n+1}^{n+m} \sum_{i=1}^{n} x_{ijk} \leq 1, k = 1, ..., K \tag{1.7}$$

$$\sum_{i=1}^{S} f_{ij} - \sum_{i=1}^{n} q_i y_{ij} = 0, j = n+1, ..., n+m \tag{1.8}$$

$$\sum_{i=1}^{S} f_{ij} - T_j z_j \leq 0, j = n+1, ..., n+m \tag{1.9}$$

[4] Η ονομασία που έδωσαν στο πρόβλημα ήταν Warehouse Location Routing Problem (WLRP)

$$\sum_{j=1}^{n+m} x_{ijk} + \sum_{l=1}^{n+m} x_{jlk} - y_{ij} \leq 1, i = 1, ..., n, j = n+1, ..., n+m, k = 1, ..., K \quad (1.10)$$

$$x_{ijk} \in \{0, 1\}, i = 1, ..., n + m, j = 1, ..., n + m, k = 1, ..., K \quad (1.11)$$

$$y_{ij} \in 0, 1, i = 1, ..., n + m, j = n + 1, ..., n + m \quad (1.12)$$

$$z_j \in \{0, 1\}, j = n + 1, ..., n + m \quad (1.13)$$

$$f_{ij} \geq 0, i = 1, ..., S, j = n + 1, ..., n + m \quad (1.14)$$

Δείκτες:

$i, j, l =$ δείκτες πελάτη ή αποθήκης

$k =$ δείκτης διαδρομής

Παράμετροι:

$d_{ij} =$ απόσταση μεταξύ σημείων i και j

$q_i =$ ζήτηση πελάτη i

$F_i =$ σταθερό κόστος αποθήκης i

$v_j =$ μεταβλητό κόστος ανά μονάδα μέσω που διανέμεται μέσω της αποθήκης j

$T_j =$ μέγεθος αποθήκης j

$c_{ij} =$ μοναδιαίο κόστος προμήθειας εμπορευμάτων από τον πηγή i στην αποθήκη j

$C_k =$ χωρητικότητα οχήματος (ή διαδρομής) k

$D_k =$ μέγιστο επιτρεπόμενο μήκος διαδρομής k

$c_k =$ κόστος ανά μονάδα μήκους για την μεταφορά προς τους πελάτες στη διαδρομή k

Μεταβλητές:

$$x_{ijk} = \begin{cases} 1 \text{ αν ο δείκτης } i \text{ προηγείται του } j \text{ στη διαδρομή } k \\ 0 \text{ αλλιώς} \end{cases}$$

$$y_{ij} = \begin{cases} 1 \text{ αν ο πελάτης } i \text{ εξυπηρετείται από την αποθήκη } j \\ 0 \text{ αλλιώς} \end{cases}$$

$$z_i = \begin{cases} 1 \text{ αν ο πελάτης } i \text{ εξυπηρετείται από την αποθήκη } j \\ 0 \text{ αλλιώς} \end{cases}$$

$f_{ij} =$ ποσότητα που μεταφέρεται από την πηγή i στην αποθήκη j

Οι όροι τις αντικειμενικής συνάρτησης 1.1 συμβολίζουν το σταθερό κόστος των αποθηκών, κόστος προμήθειας στις αποθήκες, μεταβλητό κόστος αποθηκών και κόστος παράδοσης στους πελάτες αντίστοιχα.

Ο περιορισμός 1.2 εξασφαλίζει ότι ο κάθε πελάτης θα εξυπηρετηθεί από μία μόνο διαδρομή. Επίσης αφήνει να εννοηθεί ότι η ζήτηση κάποιου πελάτη δεν μπορεί να είναι

μεγαλύτερη από τη χωρητικότητα του οχήματος. Σε μια τέτοια περίπτωση, η ζήτηση του συγκεκριμένου πελάτη εξυπηρετείται με απευθείας προμήθεια από την αποθήκη χρησιμοποιώντας το οχήματα προμήθειας των αποθηκών.

Ο περιορισμός 1.3 εξασφαλίζει ότι η ζήτηση των πελατών μιας διαδρομής k δε θα ξεπεράσει τη χωρητικότητα του οχήματος για τη συγκεκριμένη διαδρομή.

Ο περιορισμός 1.4 αφορά την περίπτωση όπου οι οδηγοί των οχημάτων δεν πρέπει να ξεπερνούν κάποιο συγκεκριμένο αριθμό ωρών συνεχόμενης εργασίας. Δεν προκύπτει από τον ορισμό του LRP, αλλά είναι μερικές φορές απαραίτητος για τον σχεδιασμό και την δημιουργία πρακτικών διαδρομών.

Ο περιορισμός 1.5 εξασφαλίζει ότι κάθε διαδρομή θα πρέπει να περιέχει μια αποθήκη από την οποία θα ξεκινούν και επιστρέφουν το όχημα της διαδρομής k.

Ο περιορισμός 1.6 δηλώνει ότι ένα όχημα παράδοσης προϊόντων αφήνει κάθε σημείο στο οποίο εισέρχεται.

Ο περιορισμός 1.7 εξασφαλίζει ότι μια διαδρομή δεν θα εξυπηρετείται από περισσότερες της μίας αποθήκης.

Ο περιορισμός 1.8 απαιτεί ότι η ροή προϊόντων προς τις αποθήκες από τους προμηθευτές ισούται με τη ροή προϊόντων από τις αποθήκες προς τους πελάτες.

Ο περιορισμός 1.9 περιορίζει την ροή από την αποθήκη προς τους πελάτες που εξυπηρετεί έτσι ώστε να μην ξεπερνούν την ποσότητα εμπορευμάτων που περιέχει.

Τέλος ο περιορισμός 1.10 συνδέει την χωροθέτηση με τη δρομολόγηση ορίζοντας ότι ένας πελάτης μπορεί να εξυπηρετείται από μια αποθήκη, μόνο αν υπάρχει μια διαδρομή που να τον περιέχει από την αποθήκη αυτή.

Οι σχέσεις 1.11-1.14 ορίζουν τις τιμές που θα παίρνουν οι μεταβλητές του προβλήματος.

Η παραπάνω μοντελοποίηση μπορεί να δημιουργήσει ένα πολύ σύνθετο υπολογιστικά πρόβλημα. Για να πάρουμε μια ιδέα πόσο πολύπλοκο μπορεί να γίνει, αρκεί να αναφέρουμε ένα απλό παράδειγμα με ένα μόνο προμηθευτή προς τις αποθήκες, τρεις πιθανές θέσεις για τις αποθήκες, 14 πελάτες και τέσσερις διαδρομές με συμμετρικές αποστάσεις. Αυτό μας δίνει 1201 ακέραιες μεταβλητές και 8455 περιορισμούς. Αν οι πελάτες γίνουν 100 και ο αριθμός των διαδρομών 25, τότε το πλήθος των ακέραιων μεταβλητών αμέσως γίνεται 265528 και ο αριθμός των περιορισμών ξεπερνά τους $6.33 \cdot 10^{29}$. Μπορούμε να καταλάβουμε την αναγκαιότητα της χρήσης ευρετικών και μεθευρετικών μεθόδων, όπως επίσης και την ανάγκη για παραδοχές που θα απλουστεύσουν το πρόβλημα.

1.2.2 Παραδοχές και η δημιουργία του προσαρμοσμένου μοντέλου

Κατανοώντας τις παραπάνω δυσκολίες, οι *J. Perl και M. S. Daskin*[2] προχώρησαν στη δημιουργία του προσαρμοσμένου LRP[5] με βάση κάποιες παραδοχές για την απλούστευση του μοντελοποιημένου προβλήματος. Θεωρούμε σταθερό το κόστος προμήθειας ανά μονάδα προϊόντος στην αποθήκη. Συνεπώς, το συνολικό κόστος προμήθειας είναι με τη σειρά του και αυτό σταθερό. Έτσι μπορούμε να εξετάζουμε τις αποθήκες σαν προμηθευτές αντί για κέντρα της αλυσίδας διανομής των εμπορευμάτων. Αυτό

[5] modified WLRP (MWLRP)

έχει και πρακτική σημασία, πέρα από την περίπτωση που τα εμπορεύματα διανέμονται κατευθείαν από τις εγκαταστάσεις στις οποίες παράγονται, επειδή η εμπειρία δείχνει ότι το κόστος διανομής στους πελάτες είναι 10 φορές μεγαλύτερο από το κόστος προμήθειας των αποθηκών. Επόμενη παραδοχή που κάνουμε είναι ότι ο στόλος διανομής αποτελείται από όμοια οχήματα. Αυτή είναι μια κοινή πρακτική στα μοντέλα δρομολόγησης οχημάτων (VRP) γιατί δίνει τη δυνατότητα να αποφύγουμε πρόσθετη πολυπλοκότητα που θα προέκυπτε από την ανάγκη να αντιστοιχήσουμε συγκεκριμένους πελάτες με συγκεκριμένο τύπο οχημάτων. Τέλος, παραδεχόμαστε ότι δεν υπάρχει περιορισμός στο μέγιστο μήκος που μπορεί να έχει κάποια διαδρομή. Αυτή η παραδοχή έχει καθαρά υπολογιστική σημασία, και μπορούμε να την απαλείψουμε για να αποφύγουμε πρόσθετο υπολογιστικό φόρτο.

Συνεπώς, η νέα αντικειμενική συνάρτηση του προσαρμοσμένου LRP γίνεται:

$$\min \sum_{i=n+1}^{n+m} F_i z_i + \sum_{j=n+1}^{n+m} v_j \left(\sum_{i=1}^{n} q_i y_{ij} \right) + \sum_{k=1}^{K} \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} c d_{ij} x_{ijk} \tag{1.15}$$

υπό:

$$\sum_{k=1}^{K} \sum_{j=1}^{n+m} x_{ijk} = 1, i = 1, ..., n \tag{1.16}$$

$$\sum_{i=1}^{n} q_i \left(\sum_{j=1}^{n+m} x_{ijk} \right) \leq C, k = 1, ..., K \tag{1.17}$$

$$\sum_{i \in N} \sum_{j \in N} \sum_{j=1}^{K} x_{ijk} \geq 1, \forall N \subset \{1, ..., n+m\} | \{n+1, ..., n+m\} \subset N \tag{1.18}$$

$$\sum_{j=1}^{n+m} x_{ijk} - \sum_{j=1}^{n+m} x_{ijk} = 0, k = 1, ..., K, i = 1, ..., n+m \tag{1.19}$$

$$\sum_{j=n+1}^{n+m} \sum_{i=1}^{n} x_{ijk} \leq 1, k = 1, ..., K \tag{1.20}$$

$$\sum_{i=1}^{n} q_i y_{ij} - T_j z_j \leq 0, j = n+1, ..., n+m \tag{1.21}$$

$$\sum_{j=1}^{n+m} x_{ijk} + \sum_{l=1}^{n+m} x_{jlk} - y_{ij} \leq 1, i = 1, ..., n, j = n+1, ..., n+m, k = 1, ..., K \tag{1.22}$$

$$x_{ijk} \in \{0, 1\}, i = 1, ..., n+m, j = 1, ..., n+m, k = 1, ..., K \tag{1.23}$$

$$y_{ij} \in 0, 1, i = 1, ..., n+m, j = n+1, ..., n+m \tag{1.24}$$

$$z_j \in \{0, 1\}, j = n+1, ..., n+m \tag{1.25}$$

Ο περιορισμός 1.16 είναι αντίστοιχος του περιορισμού 1.2 στη γενική μορφή του προβλήματος.

Ο περιορισμός 1.17 είναι αντίστοιχος του περιορισμού 1.3 στη γενική μορφή του προβλήματος.

Ο περιορισμός 1.18 είναι αντίστοιχος του περιορισμού 1.5 στη γενική μορφή του προβλήματος.

Ο περιορισμός 1.19 είναι αντίστοιχος του περιορισμού 1.6 στη γενική μορφή του προβλήματος.

Ο περιορισμός 1.20 είναι αντίστοιχος του περιορισμού 1.7 στη γενική μορφή του προβλήματος.

Από τη στιγμή που δεν έχουμε μεταφορά προϊόντων από κάποια πηγή σε αποθήκη ο όρος $\sum f_{ij}$ στους περιορισμούς 1.8 και 1.9 του γενικού προβλήματος παύει να υφίσταται. Προσαρμόζοντας αυτούς τους δύο περιορισμούς σύμφωνα με τα νέα δεδομένα προκύπτει ο περιορισμός 1.21 του προσαρμοσμένου προβλήματος. Αυτός ο περιορισμός εξασφαλίζει ότι η συνολική ζήτηση των πελατών που έχουν αντιστοιχηθεί στην εκάστοτε αποθήκη δεν ξεπερνά τη χωρητικότητά της.

Ο περιορισμός 1.22 είναι αντίστοιχος του περιορισμού 1.10 στη γενική μορφή του προβλήματος.

Για το σκοπό αυτής της εργασίας χρησιμοποιήθηκε το μοντέλο για το προσαρμοσμένο LRP. Έγιναν τροποποιήσεις (κυρίως στις μεταβλητές του προβλήματος) προκειμένου να έρθει σε κατάλληλη μορφή για να το διαχειριστούν οι διαδικασίες που αποτελούν τον μιμητικό αλγόριθμο. Οι τροποποιήσεις αυτές θα αναφερθούν παρακάτω κατά την υλοποίηση του αλγορίθμου που επιλύει το πρόβλημα.

Κεφάλαιο 2

Εξελικτικές στρατηγικές και αλγόριθμοι

2.1 Γενική περιγραφή

Όπως είδαμε στο προηγούμενο κεφάλαιο, οι απόλυτες μέθοδοι για την επίλυση του LRP δεν είναι υπολογιστικά πρακτικές για ρεαλιστικά μεγέθη προβλημάτων. Μεθευρετικοί αλγόριθμοι, όπως ο tabu search, οι γενετικοί αλγόριθμοι, η προσομοιωμένη ανόπτηση, νευρωνικά δίκτυα κλπ, έχουν χρησιμοποιηθεί για την επίλυση σημαντικών προβλημάτων συνδυαστικής βελτιστοποίησης. Στόχος αυτών των αλγορίθμων είναι η αναζήτηση του χώρου λύσεων πιο αποτελεσματικά από τις συμβατικές μεθόδους. Οπότε είναι ξεκάθαρη η δυναμική για τη χρήση τους σε δύσκολα συνδυαστικά προβλήματα όπως το LRP.

Οι πρώτοι γενετικοί αλγόριθμοι αναπτύχθηκαν και περιγράφηκαν πρώτα από τον J. Holland το 1960[1] και αργότερα από τον Goldberg το 1989[4]. Οι γενετικοί αλγόριθμοι προσομοιώνουν τη διαδικασία της εξέλιξης που συναντούμε στη φύση. Ένας πληθυσμός εξελίσσεται με την πάροδο των γενεών προκειμένου να προσαρμοστεί στο περιβάλλον του και να αντιμετωπίσει καλύτερα τα διάφορα προβλήματα που παρουσιάζονται. Ένας γενετικός αλγόριθμος είναι μια επαναληπτική στοχαστική διαδικασία όπου ένας σταθερός από άποψη πλήθους πληθυσμός πολλαπλασιάζεται για τη δημιουργία καλύτερων γενεών. Ο πληθυσμός των γενετικών αλγορίθμων αποτελείται από λύσεις του προβλήματος κατάλληλα κωδικοποιημένες σε μορφή χρωμοσωμάτων. Αντίθετα με τη φύση, κάθε χρωμόσωμα αντιπροσωπεύει και μια λύση. Ξεκινώντας από ένα αρχικό πληθυσμό από εφικτές λύσεις, οι λύσεις αυτές αξιολογούνται με συγκεκριμένο κριτήριο και με κατάλληλες διαδικασίες επιλογής και αναπαραγωγής που θα δούμε παρακάτω, πολλαπλασιάζονται δύο από αυτές (γονείς) και δημιουργούν δύο νέες λύσεις (απόγονους) μέχρι να δημιουργηθεί μια νέα γενιά χρωμοσωμάτων, όμοια σε πλήθος με την προηγούμενη. Ιδεατά, κάποια από αυτές τις λύσεις θα είναι καλύτερη από την καλύτερη λύση της προηγούμενης γενιάς. Έτσι μέσα στον πληθυσμό θα υπάρχουν καλύτερα γονίδια, τα οποία με τη σειρά τους θα οδηγήσουν στην δημιουργία ακόμα καλύτερων γενεών. Επιπλέον έχουμε και τη διαδικασία της μετάλλαξης. Ένας μικρός αριθμός του πληθυσμού μεταλλάσσεται τυχαία με σκοπό να εισαχθούν νέα γονίδια στον πληθυσμό και να αποτρέψουν τη λύση να είναι υποβέλτιστη. Η διαδικασία αυτή επαναλαμβάνεται

μέχρι να γίνει αληθής μια συνθήκη εξόδου, οπότε και τερματίζεται ο αλγόριθμος.

Οι μιμητικοί αλγόριθμοι αντιπροσωπεύουν έναν από τους πρόσφατα αναπτυσσόμενους τομείς έρευνας της εξελικτικής υπολογιστικής. Ο όρος μιμητικοί αλγόριθμοι χρησιμοποιείται ευρέως για την περιγραφή μεθόδων που αποτελούνται από τη συνεργεία μιας εξελικτικής ή πληθυσμιακής προσέγγισης με ξεχωριστή μάθηση για κάθε υποκείμενο ή διαδικασίες τοπικής βελτίωσης για την αναζήτηση του χώρου λύσεων του προβλήματος. Η τοπική αναζήτηση (local search) είναι μια μεθευρετική μέθοδος για την επίλυση δύσκολων και πολύπλοκων υπολογιστικά προβλημάτων βελτιστοποίησης. Γενικά μπορεί να περιγραφεί σαν μια μέθοδος όπου βρίσκεται μια λύση όπου μεγιστοποιείται ένα κριτήριο από ένα σύνολο πιθανών λύσεων. Οι αλγόριθμοι τοπικής αναζήτησης κινούνται από λύση σε λύση στο χώρο των πιθανών λύσεων (χώρος λύσεων του προβλήματος), μέχρι να βρεθεί μια λύση που ορίζεται ως βέλτιστη ή ξεπεραστεί κάποιος περιορισμός χρόνου εκτέλεσης.

2.2 Υλοποίηση των διαδικασιών των γενετικών αλγορίθμων

2.2.1 Κωδικοποίηση

Αν εξετάσουμε τον τρόπο με τον οποίο το γενετικό υλικό μας (DNA) αποθηκεύεται στα χρωμοσώματα, θα δούμε ότι όλα τα γονίδια που δημιουργούν έναν πολύπλοκο οργανισμό όπως ο άνθρωπος αποτελούνται από διαφορετικούς συνδυασμούς 4 αζωτούχων βάσεων. Προκειμένου να εφαρμόσουμε τους εξελικτικούς μηχανισμούς χρειάζεται να κωδικοποιήσουμε τις λύσεις μας με ανάλογο τρόπο για τη δημιουργία των χρωμοσωμάτων. Συνήθως, στους γενετικούς αλγορίθμους χρησιμοποιείται δυαδική κωδικοποίηση. Αξίζει να σημειωθεί εδώ ότι αντίθετα με το τι συναντάμε στη φύση, όπου το γενετικό υλικό ενός κυττάρου αποτελείται από πολλά χρωμοσώματα, εδώ χρησιμοποιούμε τον όρο αυθαίρετα. Έτσι ένα χρωμόσωμα αποτελεί την κωδικοποιημένη μορφή μιας λύσης.

Προκειμένου η λύση να κωδικοποιηθεί δυαδικά, αναπαρίσταται από ένα συνδυασμό των 0 και 1. Αυτή την κωδικοποίηση χρησιμοποιήσαμε για το διάνυσμα μεγέθους m των αποθηκών. Έτσι όταν μια αποθήκη είναι ανοιχτή συμβολίζεται με 1 και όταν είναι κλειστή με 0. Για παράδειγμα υποθέτουμε ότι έχουμε την περίπτωση τεσσάρων πιθανών θέσεων για τις αποθήκες. Στην περίπτωση που εξετάζουμε λειτουργούν οι εγκαταστάσεις στη θέση 2 και στη θέση 4. Αν κωδικοποιήσουμε την λύση σύμφωνα με τον τρόπο που περιγράψαμε παραπάνω, έχουμε:

Λύση:

$$\left.\begin{array}{l}\text{Αποθήκη 1 κλειστή}\\ \text{Αποθήκη 2 ανοιχτή}\\ \text{Αποθήκη 3 κλειστή}\\ \text{Αποθήκη 4 ανοιχτή}\end{array}\right\} [0101]$$

Για την δρομολόγηση των οχημάτων η δυαδική κωδικοποίηση δεν ήταν επαρκής. Θα μπορούσε να χρησιμοποιηθεί μόνο με μετατροπή των αριθμών από δεκαδικό σε δυαδικό σύστημα αρίθμησης και αντίστροφα, αλλά κάτι τέτοιο δεν είναι πρακτικό οπότε χρησιμοποιήθηκε δεκαδική κωδικοποίηση. Για την αναπαράσταση των διαδρομών έχουμε ένα *διάνυσμα δρομολόγησης* μεγέθους n που περιέχει τους πελάτες με τη σειρά που

τους επισκέπτονται τα οχήματα και ένα *διάνυσμα δείκτη* μεγέθους m όπως το διάνυσμα αποθηκών. Το διάνυσμα δείκτης στη θέση κάθε ανοιχτής αποθήκης έχει τη θέση i του διανύσματος δρομολόγησης όπου βρίσκεται ο πρώτος πελάτης που αντιστοιχεί στη συγκεκριμένη αποθήκη. Για να το καταλάβουμε αυτό, υποθέτουμε ότι έχουμε 5 πελάτες και τις ανοιχτές αποθήκες του προηγούμενου παραδείγματος. Έτσι έχουμε την ακόλουθη δρομολόγηση:

- Αποθήκη 2 $\rightarrow$ πελάτης 5 $\rightarrow$ πελάτης 2 $\rightarrow$ πελάτης 3 $\rightarrow$ αποθήκη 2
- Αποθήκη 4 $\rightarrow$ πελάτης 4 $\rightarrow$ πελάτης 1 $\rightarrow$ αποθήκη 4

Σύμφωνα με την παραπάνω περιγραφή τα διανύσματα δείκτη και δρομολόγησης είναι:

- διάνυσμα δρομολόγησης: $[52341]$
- διάνυσμα δείκτης: $[0104]$

Τέλος, για να γίνονται κάποιες διαδικασίες πιο εύκολα, χρησιμοποιείται ένα βοηθητικό *διάνυσμα αντιστοίχησης*, όπου έχει μέγεθος n και σε κάθε στοιχείο που αντιστοιχεί στον εκάστοτε πελάτη υπάρχει α αριθμός της αποθήκης στην οποία αντιστοιχεί. Αυτό το διάνυσμα χρησιμοποιείται για την αρχική δημιουργία του διανύσματος δρομολόγησης και για τον έλεγχο εγκυρότητας μιας λύσης. Ας δούμε ένα παράδειγμα:

Με βάση την παραπάνω λύση το διάνυσμα αντιστοίχησης θα είναι:

$$\text{Διάνυσμα αντιστοίχησης: } [22244]$$

Συνηθίζεται[4] η δρομολόγηση να γίνεται σε ένα διάνυσμα με κάθε πελάτη i να εκφράζεται σαν $m + i$. Έτσι η παραπάνω λύση θα συμβολιζόταν ως εξής: $[296724854]$. Το πρόβλημα είναι ότι με την αναπαράσταση μεγαλύτερων διαδρομών όπου εισάγεται και η παράμετρος της χωρητικότητας του οχήματος αυτός ο τρόπος δημιουργεί διανύσματα απρόβλεπτου μεγέθους, κάτι που κάνει δυσκολότερη την προγραμματιστική υλοποίηση. Έτσι προτιμήσαμε την παραπάνω υλοποίηση που μας δίνει το πλεονέκτημα των διανυσμάτων σταθερού μεγέθους και ευκολότερης αναπαράστασης των λύσεων.

2.2.2 Αρχικός πληθυσμός

Για την επίλυση του LRP χρησιμοποιήσαμε δύο διαφορετικές προσεγγίσεις με γενετικούς αλγόριθμους. Λύσαμε τα επιμέρους προβλήματα της χωροθέτησης και της δρομολόγησης ταυτόχρονα αλλά και ξεχωριστά με πρώτη τη χωροθέτηση και κατόπιν τη δρομολόγηση. Και στις δύο περιπτώσεις οι αρχικοί πληθυσμοί δημιουργήθηκαν τυχαία εξασφαλίζοντας όμως ότι οι λύσεις θα είναι εφικτές.

Στην πρώτη περίπτωση όπου οι δύο φάσεις του προβλήματος επιλύονται ταυτόχρονα, δημιουργείται αρχικός πληθυσμός με μέγεθος που επιλέγεται από το χρήστη. Για κάθε λύση ανοίγονται αποθήκες με τυχαίο τρόπο και τους ανατίθενται οι πελάτες με τυχαίο τρόπο, εξασφαλίζοντας όμως να μην ξεπερνούν τη χωρητικότητα κάθε αποθήκης με τη συνολική ζήτησή τους. Στη συνέχεια με βάση την αντιστοίχηση που έχει γίνει μεταξύ ανοιχτών εγκαταστάσεων και πελατών δημιουργείται μια τυχαία δρομολόγηση των πελατών.

Στην περίπτωση που λύνουμε ξεχωριστά τις δύο φάσεις του προβλήματος, θα χρειαστούμε δύο αρχικούς πληθυσμούς για κάθε υπό-πρόβλημα, αφού λύνουμε και τα δύο με

γενετικούς αλγορίθμους. Αρχικά για το υπό-πρόβλημα της χωροθέτησης των εγκαταστάσεων ανοίγουμε για κάθε λύση αποθήκες με τυχαίο τρόπο. Στη συνέχεια αντιστοιχίζονται πελάτες σε κάθε ανοιχτή αποθήκη φροντίζοντας να μην ξεπεραστεί η χωρητικότητα της εκάστοτε αποθήκης με τη ζήτηση τους. Για το υπό-πρόβλημα δρομολόγησης έχουμε ήδη λύση με ανοιχτές αποθήκες και τους πελάτες που τους αντιστοιχούν από το υπό-πρόβλημα της χωροθέτησης. Ο αρχικός πληθυσμός δημιουργείται δημιουργώντας τυχαία *διανύσματα δρομολόγησης* για την καλύτερη λύση της προηγούμενης φάσης.

2.2.3 Κριτήριο αξιολόγησης χρωμοσωμάτων (fitness)

Προκειμένου να διαχωριστούν οι καλές λύσεις από τις κακές θα πρέπει να αξιολογούνται με κάποιο κριτήριο. Έτσι κατά την εκτέλεση του αλγορίθμου οι καλύτερες λύσεις θα επιλέγονται συχνότερα για τη δημιουργία των επόμενων γενεών προσφέροντας τα καλύτερα γονίδια του τρέχοντος πληθυσμού. Για το πρόβλημα μας το κριτήριο αξιολόγησης είναι το κόστος της κωδικοποιημένης λύσης, το οποίο υπολογίζεται σύμφωνα με την αντικειμενική συνάρτηση του τροποποιημένου LRP (1.15). Έτσι για κάθε λύση υπολογίζεται το σταθερό κόστος των ανοιχτών αποθηκών, το μεταβλητό κόστος των αποθηκών ανάλογα με τον αριθμό των προϊόντων που ζητούν οι πελάτες που τους αντιστοιχούν και το κόστος διανομής το οποίο εξαρτάται από το μήκος της διαδρομής. Αφού υπολογιστεί το κόστος για κάθε χρωμόσωμα του πληθυσμού, τότε οι κωδικοποιημένες αυτές λύσεις ταξινομούνται με αύξουσα σειρά έτσι ώστε οι λύσεις με το μικρότερο κόστος να είναι πρώτες σε αντίθεση με αυτές που έχουν μεγαλύτερο κόστος. Είναι εύκολο να συμπεράνουμε ότι η λύση με το μικρότερο κόστος είναι η μηδενική, δηλαδή να μην ανοίξει καμία αποθήκη. Με αυτό τον τρόπο όμως δε θα έχει εξυπηρετηθεί κανείς από τους πελάτες. Για αυτό το λόγο εξασφαλίσαμε με τη χρήση ενός επιπλέον περιορισμού ότι για κάθε λύση πρέπει να είναι τουλάχιστον μια αποθήκη ανοιχτή και να εξυπηρετούνται όλοι οι πελάτες.

Βέβαια η συνάρτηση που χρησιμοποιούμε δεν υπολογίζει απλά το κόστος, αλλά εκτελεί και άλλες λειτουργίες που είναι σημαντικές για την υλοποίηση του προβλήματος. Το κόστος υπολογίζεται διαβάζοντας για κάθε ανοιχτή αποθήκη από το *διάνυσμα αποθηκών* σειριακά τους πελάτες του *διανύσματος δρομολόγησης* από το σημείο που αντιστοιχεί στην τρέχουσα αποθήκη μέχρι την επόμενη ανοιχτή, όπως φαίνεται από το *διάνυσμα δείκτη*. Εν τω μεταξύ σε κάθε βήμα ελέγχεται αν ο πελάτης που πρόκειται να εξυπηρετηθεί, σε συνδυασμό με αυτούς που έχουν ήδη εξυπηρετηθεί, ξεπερνά τη χωρητικότητα του οχήματος. Στην περίπτωση που η συνθήκη αυτή είναι αληθής τότε υπολογίζεται το κόστος από τον προηγούμενο πελάτη προς την τρέχουσα αποθήκη και από την τρέχουσα αποθήκη προς τον τρέχων πελάτη. Έτσι πέρα από τον υπολογισμό του κόστους, η συγκεκριμένη συνάρτηση διαχωρίζει τις επιμέρους διαδρομές που ξεκινούν από κάθε ανοιχτή αποθήκη με τέτοιο τρόπο ώστε η ζήτηση των πελατών κάθε διαδρομής να μην ξεπερνά τη χωρητικότητα του οχήματος.

Άλλη μια λειτουργία της συνάρτησης υπολογισμού του κόστους είναι ο έλεγχος κάθε λύσης αν είναι εφικτή. Σε περίπτωση που κάποιο χρωμόσωμα αποδεικνύεται ότι δεν αντιπροσωπεύει μια εφικτή λύση τότε του δίνεται ένα πολύ μεγάλο κόστος - ποινή. Τέτοιες λύσεις μπορούν να προκύψουν από τις διαδικασίες διασταύρωσης και μετάλλαξης που θα δούμε παρακάτω. Μέσω μιας διαδικασίας ελέγχου κατά την εκτέλεση του αλγορίθμου αυτές οι λύσεις απαλείφονται από τον πληθυσμό και αντικαθίστανται από την τρέχουσα βέλτιστη.

2.2.4 Αρχή του ελιτισμού

Η λογική πίσω από τους γενετικούς αλγόριθμους λέει ότι συνδυάζοντας τις καλύτερες λύσεις κάθε γενιάς δημιουργούμε μια νέα γενιά ευελπιστώντας να είναι καλύτερη από την προηγούμενη, δεδομένου ότι χρησιμοποιήθηκαν τα καλύτερα γονίδια της. Όμως αυτό δεν είναι απαραίτητο και αν δεν χρησιμοποιηθεί κάποιος μηχανισμός που να αποθηκεύει αυτή την λύση και να τη διατηρεί αμετάβλητη, μπορεί να χαθεί κατά τη δημιουργία της επόμενης γενιάς. Η αρχή του ελιτισμού δεν είναι τίποτα άλλο από αυτό το μηχανισμό. Στην ουσία μεταφέρεται στις πρώτες θέσεις της νέας γενιάς ένας πολύ μικρός αριθμός από τις καλύτερες λύσεις που έχουν υπολογιστεί μέχρι στιγμής μέχρι να βρεθεί κάποια νέα με μικρότερο κόστος. Σε αυτή την περίπτωση η καλύτερη αυτή λύση αντικαθίσταται με την νέα καλύτερη. Με αυτό τον τρόπο εξασφαλίζεται ότι πάντα θα υπάρχει μέσα στον πληθυσμό η καλύτερη λύση που έχει υπολογισθεί από την έναρξη της εκτέλεσης του αλγορίθμου και θα χρησιμοποιείται για τη δημιουργία νέων γενεών.

2.2.5 Διαδικασία επιλογής

Μία σημαντική διαδικασία για την εκτέλεση των γενετικών αλγορίθμων είναι η επιλογή των χρωμοσωμάτων που θα διασταυρωθούν για τη δημιουργία δύο νέων απογόνων. Υπάρχουν διάφορες μέθοδοι επιλογής που χρησιμοποιούνται στους γενετικούς αλγόριθμους (fitness proportionate selection ή roulette-wheel selection, stochastic universal sampling, tournament selection, truncation selection). Στη δικιά μας περίπτωση χρησιμοποιήσαμε τη μέθοδο επιλογής της ρουλέτας (roulette-wheel selection). Το κόστος των λύσεων κανονικοποιείται με τέτοιο τρόπο ώστε το κόστος της πρώτης (καλύτερης) λύσης να είναι ίσο με τη μονάδα και σταδιακά τα υπόλοιπα < 1. Στη συνέχεια μια γεννήτρια τυχαίων αριθμών γεννά έναν αριθμό $p : p \in [0, \sum f_i]$, όπου f_i το εκάστοτε κόστος της κάθε λύσης. Στη συνέχεια αθροίζουμε το κάθε f_i μέχρι να ξεπεράσουμε τον τυχαίο αριθμό p, οπότε και επιστρέφουμε το προηγούμενο i, το οποίο αντιστοιχεί στην τελευταία λύση που δεν έχει ξεπεραστεί ο p με το άθροισμα του κόστους της. Με αυτό τον τρόπο όσο μικρότερο είναι το κόστος μιας λύσης τόσο περισσότερες είναι οι πιθανότητες που έχει για να επιλεχθεί για αναπαραγωγή.

2.2.6 Διασταύρωση χρωμοσωμάτων (crossover)

Crossover ενός σημείου

Η διασταύρωση χρωμοσωμάτων είναι η κυριότερη διαδικασία με την οποία δημιουργούνται η απόγονοι που σχηματίζουν τη νέα γενιά. Αφού επιλεχθούν δύο χρωμοσώματα γονείς με κάποια συγκεκριμένη διαδικασία επιλογής όπως είδαμε πάνω επιλέγεται ένα τυχαίο σημείο του διανύσματος δρομολόγησης. Όλοι οι πελάτες από την αρχή του διανύσματος μέχρι το τυχαίο σημείο αντικαθίστανται από τους αντίστοιχους του διανύσματος του δεύτερου χρωμοσώματος. Αντίστοιχα, το ίδιο συμβαίνει και στο διάνυσμα του δεύτερου χρωμοσώματος. Ας δούμε ένα παράδειγμα χρησιμοποιώντας τη λύση που είδαμε παραπάνω και μια νέα.

Λύση 1:		Λύση 2:	
διάνυσμα δρομολόγησης:	[52341]	διάνυσμα δρομολόγησης:	[24531]
διάνυσμα αποθηκών:	[0101]	διάνυσμα αποθηκών:	[1100]
διάνυσμα δείκτη:	[0104]	διάνυσμα δείκτη:	[1300]

Τα διανύσματα δρομολόγησης τους θα διασταυρωθούν στο σημείο 3.

$$\begin{matrix}[52341]\\[24531]\end{matrix} \Longrightarrow \begin{matrix}[24541]\\[52331]\end{matrix}$$

Παρατηρούμε ότι με τη διαδικασία του crossover ενδέχεται να δημιουργηθούν διανύσματα που δεν είναι εφικτά επειδή εξυπηρετούν κάποιους πελάτες πάνω από μια φορά, ενώ κάποιοι άλλοι δεν εξυπηρετούνται καθόλου. Για αυτό το λόγο έχουμε εισάγει ένα μηχανισμό διόρθωσης στη συνάρτηση του crossover όπου οι πελάτες που επαναλαμβάνονται αντικαθίστανται από αυτούς που δεν υπάρχουν στο διάνυσμα.

Έτσι τα διανύσματα των παραπάνω απογόνων διορθώνονται με παρόμοιο τρόπο και οι λύσεις γίνονται:

Λύση 3:		Λύση 4:	
διάνυσμα δρομολόγησης:	[23541]	διάνυσμα δρομολόγησης:	[52431]
διάνυσμα αποθηκών:	[0101]	διάνυσμα αποθηκών:	[1100]
διάνυσμα δείκτη:	[0104]	διάνυσμα δείκτη:	[1300]

Αποκωδικοποιώντας όλες τις παραπάνω λύσεις στην περιγραφική τους μορφή έχουμε:

Γονείς:

- Λύση 1:
  Αποθήκη 2 ανοιχτή, αποθήκη 4 ανοιχτή
  Αποθήκη 2 → πελάτης 5 → πελάτης 2 → πελάτης 3 → αποθήκη 2
  Αποθήκη 4 → πελάτης 4 → πελάτης 1 → αποθήκη 4
- Λύση 2:
  Αποθήκη 1 ανοιχτή, αποθήκη 2 ανοιχτή
  Αποθήκη 1 → πελάτης 2 → πελάτης 4 → αποθήκη 1
  Αποθήκη 4 → πελάτης 5 → πελάτης 3 → πελάτης 1 → αποθήκη 4

Απόγονοι:

- Λύση 3:
  Αποθήκη 2 ανοιχτή, αποθήκη 4 ανοιχτή
  Αποθήκη 2 → πελάτης 2 → πελάτης 3 → πελάτης 5 → αποθήκη 2
  Αποθήκη 4 → πελάτης 4 → πελάτης 1 → αποθήκη 4
- Λύση 4:
  Αποθήκη 1 ανοιχτή, αποθήκη 2 ανοιχτή
  Αποθήκη 1 → πελάτης 5 → πελάτης 2 → αποθήκη 1
  Αποθήκη 4 → πελάτης 4 → πελάτης 3 → πελάτης 1 → αποθήκη 4

Αυτή η διαδικασία εφαρμόζεται και για τα διανύσματα των αποθηκών και αντιστοίχησης πελατών κατά την επίλυση του υπό-προβλήματος της χωροθέτησης εγκαταστάσεων.

Crossover 2 σημείων

Με παρόμοιο τρόπο μπορεί να γίνει διασταύρωση λύσεων 2 σημείων. Σε αυτή τη διαδικασία αντί να διασταυρώνεται το τμήμα των διανυσμάτων που ορίζεται από την αρχή μέχρι το τυχαίο σημείο, διασταυρώνεται το τμήμα μεταξύ δύο τυχαίων σημείων των διανύσματος δρομολόγησης. Ας δούμε ένα παράδειγμα όπου η λύση 1 και η λύση 2 που είδαμε παραπάνω διασταυρώνονται μεταξύ των σημείων 2 και 4 του διανύσματος δρομολόγησης. Έτσι έχουμε:

$$\begin{matrix}[52341] \\ [24531]\end{matrix} \implies \begin{matrix}[54531] \\ [22341]\end{matrix}$$

Επειδή οι λύσεις που προέκυψαν δεν είναι εφικτές, διορθώνονται και τα διανύσματα θέσης τους γίνονται:

Διάνυσμα δρομολόγησης λύσης 5: [54231]
Διάνυσμα δρομολόγησης λύσης 6: [25341]

2.2.7 Μετάλλαξη (mutation)

Η μετάλλαξη περιγράφει τη διαδικασία όπου ξαφνικά εμφανίζονται χαρακτηριστικά σε κάποιο άτομο του πληθυσμού χωρίς να υπάρχουν τα γονίδια στις προηγούμενες γενιές που να οδηγήσουν σε αυτά τα χαρακτηριστικά. Στη βιολογία περιγράφεται σαν αλλαγές στα γονίδια στο γενετικό υλικό ενός κυττάρου. Αν και στις περισσότερες περιπτώσεις οι αλλαγές που συμβαίνουν στις πρωτεϊνικές αλληλουχίες είναι ουδέτερες ή επιβλαβείς, υπάρχει το ενδεχόμενο να έχουν θετικό αποτέλεσμα σε ένα οργανισμό. Σε αυτή την περίπτωση μπορεί η μετάλλαξη να επιτρέπει στον οργανισμό να ανταπεξέρχεται σε συγκεκριμένες περιβαλλοντικές συνθήκες καλύτερα από τον υπόλοιπο πληθυσμό. Σε αυτή την περίπτωση η αλλαγή που έγινε μέσω της μετάλλαξης τείνει να συναντάται πιο συχνά στο σύνολο του πληθυσμού μέσω της φυσικής επιλογής. Στους γενετικούς αλγόριθμους η μετάλλαξη έχει το ρόλο να εισάγει νέα χαρακτηριστικά στον πληθυσμό έτσι ώστε να μπορεί να ερευνηθεί ολόκληρος ο χώρος λύσεων. Επειδή αυτές οι μεταλλάξεις μπορούν να οδηγήσουν είτε σε χειρότερες ή μη εφικτές λύσεις, η πιθανότητα να συμβεί μετάλλαξη είναι συνήθως μικρή. Παρ'όλα αυτά, τα τροποποιημένα διανύσματα μπορεί με τη διασταύρωση τους να οδηγήσουν σε νέες καλύτερες λύσεις που δεν ήταν προσβάσιμες προηγουμένως από τον πληθυσμό γιατί δεν υπήρχαν αυτά τα γονίδια. Η διαδικασία αυτή μπορεί να υλοποιηθεί με διάφορους τρόπους που συνήθως εξαρτώνται από την ίδια την υλοποίηση του προβλήματος. Εδώ μετάλλαξη θα έχουμε στα τρία διανύσματα που αποτελούν την κωδικοποίηση μιας λύσης.

Μετάλλαξη διανύσματος αποθηκών

Στο διάνυσμα αποθηκών η μετάλλαξη γίνεται σε ένα από τα στοιχεία του. Αν το τυχαίο στοιχείο που θα επιλεγεί συμβολίζει μια ανοιχτή αποθήκη, είναι δηλαδή μονάδα, τότε στη θέση του μπαίνει η τιμή 0 και η αποθήκη αυτή κλείνει. Στην αντίθετη περίπτωση αν το τυχαίο στοιχείο έχει την τιμή 0 αντικαθίσταται με την τιμή 1. Σημειώνεται ότι στην περίπτωση που το τυχαίο σημείο που θα επιλεγεί είναι η τελευταία ανοιχτή αποθήκη, τότε η συνάρτηση την κλείνει και ανοίγει μια άλλη. ΜΕ αυτό τον τρόπο διασφαλίζεται ότι τουλάχιστον μια αποθήκη θα είναι ανοιχτή.

Ας δούμε ένα παράδειγμα με το διάνυσμα αποθήκης της λύσης 1. Η μετάλλαξη θα συμβεί στη θέση 2.

$$[0101] \Longrightarrow [0001]$$

Όπως είναι αναμενόμενο μια τέτοια αλλαγή επηρεάζει και το διάνυσμα δείκτη. Με τη χρήση κατάλληλου μηχανισμού, στην περίπτωση που κλείνει μια αποθήκη γίνεται κατάλληλη τροποποίηση και του αντίστοιχου στοιχείου στο διάνυσμα δείκτη, ενώ το διάνυσμα διορθώνεται με τέτοιο τρόπο ώστε να αντιπροσωπεύει πάντα μια εφικτή λύση. Π.χ. Μια αλλαγή που θα μπορούσε να επιφέρει ένα τέτοιο πρόβλημα είναι να κλείσει μέσω μετάλλαξης η πρώτη ανοιχτή αποθήκη. Σε αυτή την περίπτωση θα αντικατασταθεί και το αντίστοιχο στοιχείο στο διάνυσμα δείκτη με την τιμή 0. Τότε όμως δε θα υπάρχει αποθήκη που να εξυπηρετεί τους πρώτους πελάτες του διανύσματος δρομολόγησης. Έτσι απαιτείται διόρθωση για να γίνει η λύση εφικτή και να εξυπηρετούνται όλοι οι πελάτες του προβλήματος.

Στην περίπτωση που ανοίγει μια νέα αποθήκη μέσω της διαδικασίας της μετάλλαξης, θα πρέπει να της αντιστοιχεί και ένα στοιχείο στο διάνυσμα δείκτη. Σε αυτή την περίπτωση επιλέγεται μια τυχαία θέση του διανύσματος δρομολόγησης μεταξύ των δύο ανοιχτών αποθηκών που βρίσκεται η νέα αποθήκη. Και σε αυτή την περίπτωση γίνεται διόρθωση του διανύσματος με παρόμοιο τρόπο έτσι ώστε να εξασφαλίζεται η ότι το διάνυσμα αυτό θα αντιπροσωπεύει εφικτή λύση.

Μετάλλαξη διανύσματος δρομολόγησης

Η μετάλλαξη εδώ μπορεί να συμβεί με δύο τρόπους. Ο πρώτος είναι η επιλογή δύο τυχαίων στοιχείων του διανύσματος δρομολόγησης ενός χρωμοσώματος και η αντιμετάθεση τους. Ο δεύτερος είναι η αντιστροφή των στοιχείων του διανύσματος δρομολόγησης, δηλαδή το τελευταίο να έρθει στην πρώτη θέση, το προτελευταίο στη δεύτερη θέση κλπ. Η διαδικασία αυτή ολοκληρώνεται μόλις αντιμετατεθούν όλα τα στοιχεία.

Εδώ γίνεται μετάλλαξη με τον πρώτο τρόπο στις θέσεις 2 και 4 του διανύσματος δρομολόγησης της λύσης 1.

$$[52341] \Longrightarrow [54321]$$

Ενώ εδώ γίνεται μετάλλαξη με τον δεύτερο τρόπο στο διάνυσμα δρομολόγησης της ίδιας λύσης.

$$[52341] \Longrightarrow [14325]$$

Μετάλλαξη διανύσματος δείκτη

Η μετάλλαξη συμβαίνει σε αυτή την περίπτωση σε ένα μη μηδενικό στοιχείο του διανύσματος δείκτη. Επιλέγεται ένα τυχαίο στοιχείο που αντιστοιχεί σε ανοιχτή αποθήκη και του δίνεται μια νέα τιμή μεταξύ των τιμών των άλλων μη μηδενικών στοιχείων του διανύσματος και των ορίων που μπορούν να παίρνουν οι τιμές του ($ret_i \in [1, n]$). Με αυτό τον τρόπο επιτυγχάνεται η μεταφορά πελατών που αντιστοιχούν σε μια ανοιχτή

αποθήκη σε μια άλλη ανοιχτή. Προφανώς, η μετάλλαξη αυτή δεν έχει νόημα όταν είναι ανοιχτή μια και μοναδική αποθήκη.

Ας δούμε ένα παράδειγμα μεταλλάσσοντας το διάνυσμα δείκτη της λύσης 1.

$$\begin{array}{lll} \text{Διάνυσμα αποθηκών:} & [0101] & \\ \text{Διάνυσμα δείκτη:} & [0104] & \Longrightarrow \quad [0103] \end{array} \quad [0101]$$

Παρατηρούμε ότι η μετάλλαξη έγινε στη θέση 4. Κάτι τέτοιο δε θα είχε νόημα να συμβεί στη θέση 2 επειδή πρόκειται για την πρώτη ανοιχτή αποθήκη, άρα η τιμή του στοιχείου που της αντιστοιχεί πρέπει να είναι αναγκαστικά 1. Βλέπουμε ότι πήρε τυχαία την τιμή 3, όπου $ret_4 = 3 \in (ret_2 = 1, n = 5] \subset \mathbb{N}_1$.

2.3 Υλοποίηση των διαδικασιών των μιμητικών αλγορίθμων

2.3.1 Η μέθοδος τοπικής αναζήτησης 2-Opt

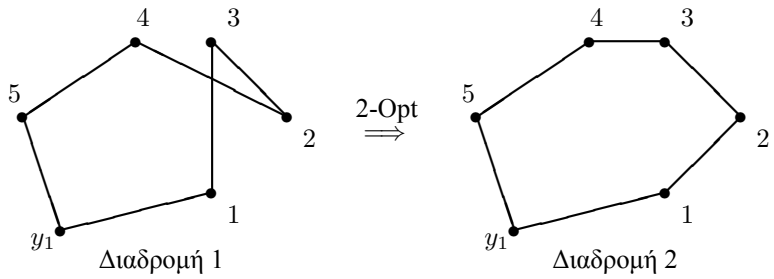
Η μέθοδος τοπικής αναζήτησης που χρησιμοποιούμε στον αλγόριθμο μας είναι η 2-Opt. Πρόκειται για ένα απλό αλγόριθμο τοπικής αναζήτησης που προτάθηκε πρώτη φορά από τον Croes το 1958 για την επίλυση του προβλήματος του πλανόδιου πωλητή (TSP)[1]. Το VRP που αποτελεί μέρος του LRP πρόκειται για μια επέκταση του προβλήματος του πλανόδιου πωλητή για τις περιπτώσεις που δεν είναι δυνατή η επίσκεψη όλων των πελατών από ένα μόνο όχημα. Η κεντρική ιδέα πίσω από αυτή τη μέθοδο είναι να βρεθεί μια διαδρομή που περνά πάνω από τον εαυτό της και να την αναδιανείμει με τέτοιο τρόπο ώστε αυτό να μην συμβαίνει.

Η διαδικασία που ακολουθείται με την χρήση αυτής της μεθόδου είναι η διαγραφή δύο ακμών σε μια διαδρομή και η επανένωσή τους με διαφορετικό τρόπο σε σχέση με πριν. Στην περίπτωση του 2-Opt αυτό μπορεί να γίνει με ένα μόνο τρόπο για τη δημιουργία μιας διαδρομής που περιλαμβάνει τους ίδιους κόμβους με πριν και αυτός είναι η σύνδεση των ελεύθερων άκρων με τον αντίθετο τρόπο που ήταν συνδεδεμένα. Αν η διαδρομή που προκύπτει δεν είναι καλύτερη σε σχέση με την αρχική απορρίπτουμε την αλλαγή και ψάχνουμε για ένα άλλο ζευγάρι ακμών που με τον τρόπο που μόλις περιγράψαμε θα μας δώσει μια καλύτερη διαδρομή.

Για την καλύτερη κατανόηση της μεθόδου ας δούμε ένα παράδειγμα για την επίλυση του προβλήματος TSP. Θεωρούμε μια αποθήκη y_1 και 5 πελάτες διασκορπισμένους στο χώρο, όπως φαίνονται στο σχήμα 2.1. Θεωρούμε σαν αρχική διαδρομή τη διαδρομή 1 με συνολικό κόστος λόγω μήκους 11.09 μονάδες και επιχειρούμε να βρούμε μια νέα καλύτερη με τη μέθοδο 2-Opt.

Για να βελτιωθεί η διαδρομή 1 που βλέπουμε στο σχήμα με τη χρήση του αλγορίθμου 2-Opt, θα "κόψουμε" τις ακμές 4-2 και 3-1. Με αυτό τον τρόπο έχουμε δύο ανοιχτές διαδρομές, την 1-y_1-5-4 και την 2-3. Αν τώρα ενώσουμε τον κόμβο 4 με αυτόν που ήταν πριν συνδεδεμένος ο κόμβος 1, δηλαδή τον κόμβο 3, και αντίστοιχα ενώσουμε τον κόμβο 1 με τον κόμβο 2 θα δημιουργηθεί μια νέα κλειστή διαδρομή. Το αποτέλεσμα θα είναι η διαδρομή 2 του σχήματος με συνολικό κόστος λόγω μήκους 9.26 μονάδες.

[1] Travelling Salesman Problem



Σχήμα 2.1: Παράδειγμα επίλυσης TSP με τη μέθοδο 2-Opt

Στο πρόβλημα που επιλύουμε η μέθοδος αυτή εφαρμόζεται στο διάνυσμα δρομολόγησης. Επιλέγονται δύο τυχαία σημεία του διανύσματος και το τμήμα που περιέχεται ανάμεσα σε αυτούς τους δύο πελάτες αντιστρέφεται. Έτσι ο επόμενος πελάτης του πρώτου σημείου τομής γίνεται ο προηγούμενος του δεύτερου σημείου τομής και οι πελάτες που βρίσκονται ανάμεσα τους θα εξυπηρετηθούν από τα οχήματα διανομής με την αντίθετη σειρά. Αν το νέο διάνυσμα έχει μικρότερο κόστος από το προηγούμενο τότε το αντικαθιστά. Στην αντίθετη περίπτωση το νέο διάνυσμα απορρίπτεται και συνεχίζεται η εκτέλεση των υπολοίπων διαδικασιών του γενετικού αλγορίθμου.

Εφαρμόζοντας παρόμοια κωδικοποίηση στο παραπάνω παράδειγμα με αυτή του διανύσματος δρομολόγησης τότε θα έχουμε:

$$\text{Διάνυσμα 1: } [y_1 13245 y_1]$$

Κόβοντας το διάνυσμα στους κόμβους 4 και 1 έχουμε τα εξής τρία διανύσματα:

$$[y_1 1][32][45y_1]$$

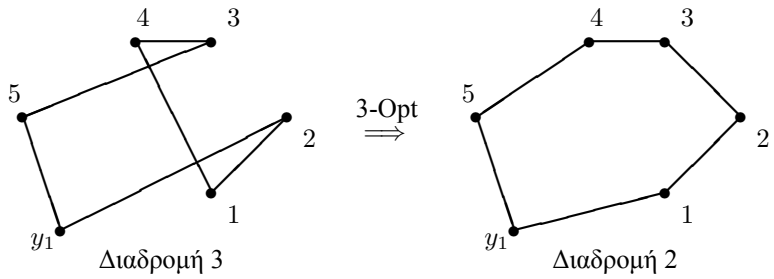
Αν αντιστρέψουμε το εσωτερικό διάνυσμα και τα ενώσουμε πάλι σε ένα, τότε έχουμε σαν αποτέλεσμα το διάνυσμα που περιγράφει τη διαδρομή 2:

$$\text{Διάνυσμα 2: } [y_1 12345 y_1]$$

Αυτή η διαδικασία μπορεί να γίνει πολύ απαιτητική σε υπολογιστικούς πόρους αν χρησιμοποιηθεί επαναληπτικά μέχρι τη εύρεση μιας καλύτερης λύσης ερευνώντας το σύνολο των διαθέσιμων κινήσεων και χαλάει την "τυχαιότητα" που εισάγει ο γενετικός αλγόριθμος στον πληθυσμό. Για αυτό το λόγο, στην υλοποίηση που έχουμε κάνει εδώ, γίνεται μόνο μια τυχαία κίνηση για κάθε μέλος του πληθυσμού και αν οδηγήσει σε βελτίωση γίνεται αποδεκτή.

2.3.2 Η μέθοδος τοπικής αναζήτησης 3-Opt

Αντίστοιχα με την μέθοδο τοπικής αναζήτησης 2-Opt που είδαμε προηγουμένως υπάρχει και η μέθοδος 3-Opt. Η διαφορά τους είναι ότι σε αυτή την περίπτωση κόβουμε τρεις ακμές αντί για δύο και ύστερα ενώνουμε τα ελεύθερα άκρα πάλι με διαφορετικό τρόπο



Διαδρομή 3 Διαδρομή 2

Σχήμα 2.2: Παράδειγμα επίλυσης TSP με τη μέθοδο 3-Opt

σε σχέση με τον αρχικό. Και σε αυτή την περίπτωση η νέα διαδρομή που προκύπτει γίνεται αποδεκτή μόνο αν έχει μικρότερο κόστος σε σχέση με την αρχική.

Έχοντας σαν αρχική διαδρομή την διαδρομή 3 του σχήματος 2.2 εφαρμόζουμε τη μέθοδο 3-Opt με σκοπό την εύρεση μιας συντομότερης διαδρομής. Το διάνυσμα που περιγράφει τη διαδρομή 3 είναι το εξής:

$$\text{Διάνυσμα 3: } [y_1 21435 y_1]$$

Για να εφαρμόσουμε τη μέθοδο θα πρέπει να επιλέξουμε τρεις ακμές που θα απαλειφθούν έτσι ώστε να μπορέσουμε να συνδέσουμε τους ελεύθερους κόμβους με διαφορετικό τρόπο. Επιλέγουμε τις ακμές y_1-2, 1-4 και 3-5. Αυτό έχει σαν αποτέλεσμα το διάνυσμα 3 να χωριστεί σε 4 μικρότερα διανύσματα:

$$[y_1][21][43][5 y_1]$$

Αντιστρέφοντας τα δύο μεσαία διανύσματα έτσι ώστε όλοι οι ελεύθεροι κόμβοι να συνδεθούν με άλλους ελεύθερους που δεν ήταν πριν συνδεδεμένοι και έτσι προκύπτει το διάνυσμα που περιγράφει τη διαδρομή 2, που αποτελεί και τη βέλτιστη λύση του συγκεκριμένου συστήματος.

$$\text{Διάνυσμα 2: } [y_1 12345 y_1]$$

Παρατηρούμε ότι στην περίπτωση του 3-Opt ο τρόπος με τον οποίο μπορούμε να επανασυνδέσουμε τους ελεύθερους κόμβους δεν είναι μοναδικός. Συγκεκριμένα άλλες πιθανές εκδοχές είναι:

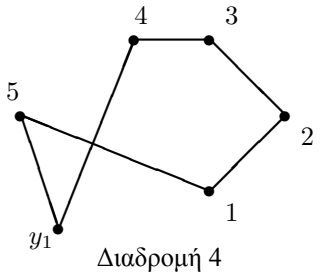
$$\text{Διάνυσμα 4: } [y_1 43215 y_1] \text{ (βλ. Σχήμα 2.3)}$$

$$\text{Διάνυσμα 5: } [y_1 34125 y_1] \text{ (βλ. Σχήμα 2.4)}$$

$$\text{Διάνυσμα 6: } [y_1 43125 y_1] \text{ (βλ. Σχήμα 2.5)}$$

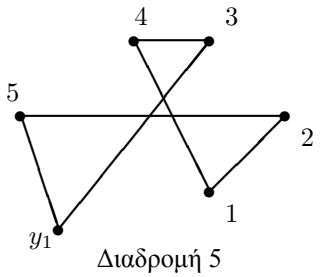
$$\text{Διάνυσμα 7: } [y_1 34215 y_1] \text{ (βλ. Σχήμα 2.6)}$$

Όπως και η 2-Opt, η μέθοδος 3-Opt μπορεί να γίνει πολύ απαιτητική σε υπολογιστικούς πόρους αν υλοποιηθεί επαναληπτικά για την έρευνα όλων των διαθέσιμων κινήσεων. Η



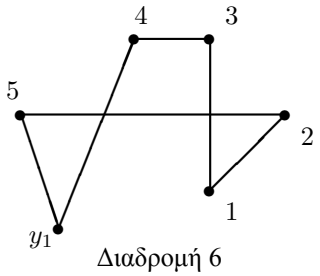
Διαδρομή 4

Σχήμα 2.3: Γραφική απεικόνιση διανύσματος 4



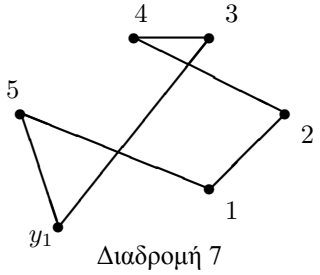
Διαδρομή 5

Σχήμα 2.4: Γραφική απεικόνιση διανύσματος 5



Διαδρομή 6

Σχήμα 2.5: Γραφική απεικόνιση διανύσματος 6



Διαδρομή 7

Σχήμα 2.6: Γραφική απεικόνιση διανύσματος 7

υλοποίηση που κάναμε στον αλγόριθμό μας είναι παρόμοια με αυτή του 2-Opt. Δηλαδή ελέγχεται αν το παραγόμενο διάνυσμα έχει μικρότερο κόστος από το αρχικό και αν επαληθεύεται αυτή η συνθήκη, τότε αντικαθιστούμε το αρχικό διάνυσμα με το καινούριο. Όσον αφορά την έρευνα μεταξύ των διαθέσιμων τρόπων επανασύνδεσης η υλοποίηση που έχει γίνει είναι αυτή που περιγράφεται στο παράδειγμα για την παραγωγή του διανύσματος 2. Δεδομένου ότι η μέθοδος αυτή χρησιμοποιείται σαν βοηθητική των γενετικών αλγορίθμων επιθυμούμε την εξοικονόμηση πόρων στο σύστημα για να επιτυγχάνουμε γρηγορότερη εκτέλεση του μιμητικού αλγορίθμου. Αν διερευνούσαμε και αυτές τις περιπτώσεις θα χάναμε πολύτιμο χρόνο για την εκτέλεση των υπόλοιπων διαδικασιών του αλγορίθμου χωρίς να κερδίζουμε με σημαντική βελτίωση στο κόστος της τρέχουσας λύσης που διερευνάται.

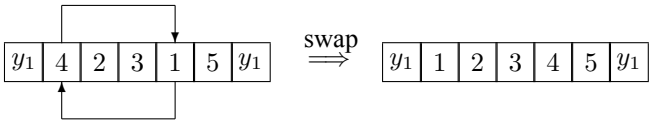
2.3.3 Η μέθοδος τοπικής αναζήτησης αντικατάστασης (Swap)

Από την πληθώρα μεθόδων τοπικής αναζήτησης, η μέθοδος της αντικατάστασης (swap) είναι μια από τις πιο απλές και εύκολες στην υλοποίηση. Η εφαρμογή της γίνεται με την επιλογή δύο στοιχείων ενός διανύσματος μέσω κάποιας διαδικασίας ή με τυχαίο τρόπο, και ολοκληρώνεται με την αντικατάσταση του ενός με το άλλο. Μπορεί λόγω της απλότητας της να μην προκαλεί ριζικές αλλαγές στο διάνυσμα και η χρήση της να μην είναι τόσο αποδοτική όσο άλλες μέθοδοι τοπικής αναζήτησης, αλλά η εφαρμογή της σε συνδυασμό με άλλες μεθόδους συντελεί σημαντικά την βελτίωση των αποτελεσμάτων.

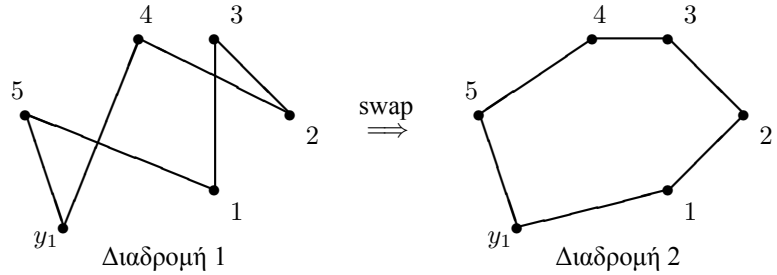
Στην περίπτωση του μιμητικού αλγορίθμου που έχουμε κατασκευάσει, αυτή η μέθοδος εφαρμόζεται στο διάνυσμα δρομολόγησης. Δύο πελάτες επιλέγονται με τυχαίο τρόπο και γίνεται η αντικατάσταση του ενός με τον άλλο, όπως προστάζει η μέθοδος. Αυτοί οι πελάτες δεν είναι απαραίτητο ότι θα ανήκουν στην ίδια διαδρομή ή θα εξυπηρετούνται από την ίδια αποθήκη, έτσι με τον τρόπο αυτό δημιουργείται ένα διάνυσμα με χαρακτηριστικά που είναι πιθανό να μην υπάρχουν εκείνη τη στιγμή στον πληθυσμό.

Το κόστος της νέας λύσης που παράγεται δεν είναι απαραίτητο ότι θα είναι μικρότερο από την αρχική. Μάλιστα, στην περίπτωση που το πρόβλημα που επιλύεται έχει πρόβλημα με την χωρητικότητα των αποθηκών και την ζήτηση των πελατών που εξυπηρετούνται από αυτή, τότε η λύση που προκύπτει μπορεί να μην είναι καν εφικτή. Για αυτό το λόγο στην περίπτωση που υπάρχει πρόβλημα με την συνολική ζήτηση που μπορεί να ικανοποιήσει η αποθήκη, γίνεται έλεγχος για την εφικτότητα της λύσης και κατά πόσο ικανοποιεί όλους τους περιορισμούς. Εάν η νέα λύση δεν δημιουργεί πρόβλημα με τον τρόπο που έχει αντιστοιχήσει τους πελάτες στις ανοιχτές αποθήκες και το κόστος της είναι μικρότερο σε σχέση με την λύση πριν την εφαρμογή της μεθόδου, τότε γίνεται δεκτή αντικαθιστώντας την αρχική. Στην αντίθετη περίπτωση η λύση που προέκυψε με τη μέθοδο της αντικατάστασης απορρίπτεται και διατηρείται η παλιά.

Για να γίνει πιο κατανοητή η εφαρμογή της μεθόδου, θα δούμε πως μπορεί να εφαρμοστεί σε ένα διάνυσμα που περιγράφει την λύση του προβλήματος περιπλανώμενου πωλητή. Στην αριστερή πλευρά του σχήματος 2.7 έχουμε ένα διάνυσμα το οποίο περιγράφει μια λύση του TSP. Στην αριστερή πλευρά του σχήματος 2.8 μπορούμε να δούμε την γραφική απεικόνιση της λύσης που περιγράφεται στο παραπάνω διάνυσμα. Αν γίνει αντικατάσταση του δεύτερου στοιχείου με το πέμπτο όπως φαίνεται στο σχήμα 2.7, τότε θα προκύψει το διάνυσμα στην δεξιά πλευρά. Η γραφική απεικόνιση του νέου διανύσματος είναι αυτή που βρίσκεται στη δεξιά πλευρά του σχήματος 2.8. Παρατηρούμε



Σχήμα 2.7: Εφαρμογή της μεθόδου αντικατάστασης σε διάνυσμα.



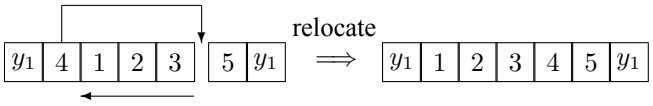
Σχήμα 2.8: Παράδειγμα επίλυσης TSP με τη μέθοδο αντικατάστασης.

ότι η νέα λύση είναι καλύτερη από την προηγούμενη, αφού δεν υπάρχουν ακμές που διασταυρώνονται, επομένως η συνολική απόσταση που διανύεται είναι λιγότερη.

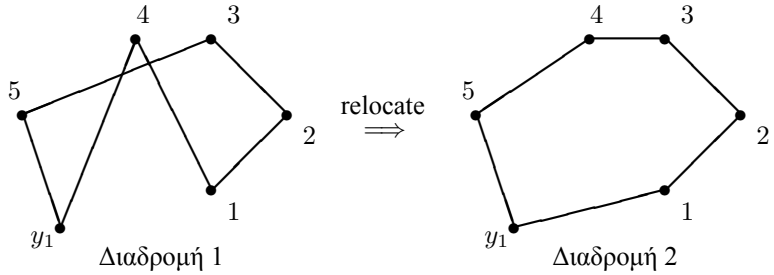
2.3.4 Η μέθοδος τοπικής αναζήτησης επανατοποθέτησης (Relocate)

Άλλη μία απλή μέθοδος τοπικής αναζήτησης είναι η μέθοδος επανατοποθέτησης. Όπως και η μέθοδος της αντικατάστασης έτσι και αυτή είναι πολύ εύκολη στην εφαρμογή και κατανόησή της. Για την εφαρμογή της μεθόδου χρειάζεται να επιλεγεί κάποιο στοιχείο ενός διανύσματος είτε με τη χρήση κάποιου κριτηρίου είτε τυχαία. Ύστερα αυτό το στοιχείο μετακινείται σε μια νέα θέση στο διάνυσμα, η οποία βρίσκεται δεξιά ή αριστερά της αρχικής. Σε ομοιότητα πάλι με την μέθοδο αντικατάστασης, η επανατοποθέτηση μπορεί να βοηθήσει σημαντικά στην επίτευξη καλών λύσεων σε συνεργασία με άλλες μεθόδους έρευνας του χώρου λύσεων.

Η εφαρμογή της στον μιμητικό αλγόριθμο γίνεται στο διάνυσμα δρομολόγησης. Ένας πελάτης και η θέση στην οποία θα καταλήξει επιλέγονται με τυχαίο τρόπο και το στοιχείο αυτό μετακινείται στην νέα θέση. Όλα τα στοιχεία που βρίσκονται ανάμεσα σε αυτά τα δύο σημεία του διανύσματος μετακινούνται με τέτοιο τρόπο προς τη διπλανή τους θέση, έτσι ώστε να καλυφθεί το κενό από τη αρχική θέση του επιλεγμένου στοιχείου. Το αποτέλεσμα αυτής της κίνησης είναι η μετακίνηση ενός στοιχείου από μια διαδρομή σε μία άλλη ή και να αντιστοιχηθεί σε κάποια άλλη αποθήκη. Αν ο πελάτης αυτός ήταν μακρυά από την αποθήκη που τον εξυπηρετούσε είναι πιθανό να καταλήξει σε μια άλλη που βρίσκεται πιο κοντά του, με αποτέλεσμα το κόστος που απαιτείται για την εξυπηρέτηση του να είναι μικρότερο. Αντίστοιχα μπορεί αυτή η μετακίνηση να γίνει από μια διαδρομή σε μια άλλη πιο κατάλληλη και έτσι να μειωθεί η απόσταση που ακολουθούν τα οχήματα. Όπως και στην μέθοδο αντικατάστασης, η νέα λύση μπορεί να μην ικανοποιεί όλους τους περιορισμούς. Αν είναι εφικτή και το κόστος μετά την με-



Σχήμα 2.9: Εφαρμογή της μεθόδου επανατοποθέτησης σε διάνυσμα.



Σχήμα 2.10: Παράδειγμα επίλυσης TSP με τη μέθοδο επανατοποθέτησης.

τακίνηση του στοιχείου είναι μικρότερο, τότε η νέα λύση γίνεται δεκτή. Σε διαφορετική περίπτωση απορρίπτεται και χρησιμοποιείται η αρχική.

Μπορούμε να δούμε ένα παράδειγμα εφαρμογής της μεθόδου στο πρόβλημα του περιπλανόμενου πωλητή. Αρχικά έχουμε το διάνυσμα που βλέπουμε στην αριστερή πλευρά του σχήματος 2.9, όπου μπορούμε να δούμε τη γραφική του απεικόνιση στην αριστερή πλευρά του σχήματος 2.10. Επιλέγουμε να μετακινήσουμε τον πελάτη που βρίσκεται στο δεύτερο στοιχείο και να τον επανατοποθετήσουμε στην πέμπτη θέση όπως φαίνεται στο σχήμα. Τα στοιχεία που βρίσκονται στην δεύτερη, τρίτη και τέταρτη θέση μετακινούνται κατά μία θέση αριστερά, έτσι ώστε να μην υπάρχει το κενό που δημιουργείται από τη μετακίνηση του πελάτη 4 στη δεύτερη θέση του διανύσματος. Το διάνυσμα που προκύπτει μετά την εφαρμογή της μεθόδου φαίνεται στην δεξιά πλευρά του σχήματος 2.9 και από την γραφική του απεικόνιση στο δεξί μέρος του σχήματος 2.10. Παρατηρούμε ότι τώρα η λύση έχει βελτιωθεί, αφού το συνολικό μήκος που διανύεται είναι λιγότερο.

2.3.5 Η αρχή του βέλτιστου

Η *Αρχή του βέλτιστου* είναι μια ιδιότητα πάνω στην οποία βασίζεται ο *Δυναμικός Προγραμματισμός*, που αποτελεί τη γενική ονομασία του *Προβλήματος Βέλτιστου Ελέγχου (ΠΒΕ)* και άλλων προβλημάτων βελτιστοποίησης. Η μέθοδος του Δυναμικού Προγραμματισμού αναπτύχθηκε στις αρχές της δεκαετίας του 1950 από τον μαθηματικό R. E. Bellman. Του έδωσε αυτή την ονομασία για να κρύψει το γεγονός ότι ο Αμερικάνος μαθηματικός έκανε μαθηματική έρευνα κάτω από την RAND, μια εταιρία που δούλευε εκείνη την περίοδο για την πολεμική αεροπορία. Ο υπουργός άμυνας των ΗΠΑ τότε ήταν ένας άνθρωπος που μισούσε τη λέξη "έρευνα" και "μαθηματικά". Ο Bellman προκειμένου να καμουφλάρει την έρευνά του για την επίλυση στοχαστικών διαδικασιών

αποφάσεων πολλαπλών βαθμίδων.

Οι εφαρμογές του Δυναμικού Προγραμματισμού είναι πολλές και εκτείνονται σε διάφορες περιοχές (Επιχειρησιακή Έρευνα, Τεχνικές Επιστήμες, Οικονομικές Επιστήμες) και σε διάφορους τύπους προβλημάτων (οργάνωση, σχεδιασμός, αυτόματος έλεγχος). Η *Αρχή του Βέλτιστου* βρίσκει άμεση εφαρμογή στην επίλυση *προβλημάτων αποφάσεων πολλαπλών βαθμίδων*, που αποτελούν κατηγορία των συνδυαστικών προβλημάτων βελτιστοποίησης. Σε αυτά τα προβλήματα η μετάβαση από ένα σημείο μιας βαθμίδας στην επόμενη βαθμίδα είναι δυνατή μέσω ενός πεπερασμένου αριθμού εναλλακτικών αποφάσεων. Κάθε απόφαση, και μαζί της η αντίστοιχη μετάβαση στην επόμενη βαθμίδα, συνδέεται με ένα αντίστοιχο κόστος. Το πρόβλημα έγκειται στην ελαχιστοποίηση του συνολικού κόστους κατά τη μετάβαση από μια αρχική βαθμίδα 0 στην τελική βαθμίδα K.

Βασική προϋπόθεση για την εφαρμογή του Δυναμικού Προγραμματισμού είναι η διαμόρφωση του συνδυαστικού προβλήματος σε *πρόβλημα αποφάσεων πολλαπλών βαθμίδων*. Η διαμόρφωση αυτή είναι άμεση σε προβλήματα βραχύτατων μονοπατιών οργανωμένων σε βαθμίδες, λόγω της φύσης των συγκεκριμένων προβλημάτων. Αντίθετα, η διαμόρφωση άλλων συνδυαστικών προβλημάτων σε αντίστοιχα προβλήματα αποφάσεων πολλαπλών βαθμίδων μπορεί να μην είναι προφανής χωρίς κατάλληλη επεξεργασία (ή και να μην είναι δυνατή).

Όπως είδαμε και προηγουμένως η κωδικοποίηση του διανύσματος δρομολόγησης μοιάζει πολύ με την αντίστοιχη του *προβλήματος του περιπλανώμενου πωλητή (TSP)*, ένα πρόβλημα που μπορεί να παρουσιασθεί σαν πρόβλημα πολλαπλών βαθμίδων μετά από κατάλληλη μορφοποίηση, και να επιλυθεί με εφαρμογή του Δυναμικού Προγραμματισμού.

Στην βασική εκδοχή του TSP έχουμε ένα δίκτυο που αποτελείται από κόμβους, όπου το κόστος μετάβασης από τον κόμβο i στον κόμβο j είναι d_{ij}. Προφανώς $d_{ii} = 0$ και είναι δυνατό να μην ισχύει η συνθήκη $d_{ij} = d_{ji}$. Ένας πωλητής ξεκινά από τον πρώτο κόμβο και πρέπει να επισκεφτεί κάθε ένα από τους υπόλοιπους $N-1$ κόμβους ακριβώς μια φορά. Η λύση του προβλήματος είναι η διαδρομή που ελαχιστοποιεί το συνολικό κόστος (συνολικό μήκος, συνολική χρονική διάρκεια ταξιδιού).

Για τη μορφοποίηση του προβλήματος αυτού σε πρόβλημα αποφάσεων πολλαπλών βαθμίδων θεωρούμε ότι ο αριθμός των βαθμίδων K ισούται με τον συνολικό αριθμό των κόμβων, δηλαδή $K = N$. Στις βαθμίδες 0 και K υπάρχει μόνο ένας κόμβος που αντιστοιχεί στην αφετηρία και τελικού προορισμό, δηλαδή στον κόμβο 1 και κόμβο K. Στις βαθμίδες $k = 1, \ldots, K-1$ κάθε κόμβος απόφασης χαρακτηρίζεται από ένα ζεύγος S, j όπου j είναι ο κόμβος προορισμού για τον κόμβο αυτό και S είναι ένα σύνολο ενδιάμεσων κόμβων από τον κόμβο 1 στον κόμβο j, άρα $1 \notin S$ και $j \notin S$. Ο αριθμός $|S|$ των ενδιάμεσων κόμβων για κάθε κόμβο της βαθμίδας k ορίζεται ίσος με $k-1$ κ.ο.κ., μέχρι τη βαθμίδα $K-1$ όπου το σύνολο για κάθε κόμβο $S = \emptyset$, στην βαθμίδα 2 έχουμε $|S| = 1$ κ.ο.κ., μέχρι την βαθμίδα $K-1$ όπου το σύνολο των ενδιάμεσων κόμβων κάθε κόμβου έχει $N-2$ στοιχεία, δηλαδή $|S| = N-2$. Έτσι, ο αριθμός κόμβων της βαθμίδας k καθορίζεται από όλους τους δυνατούς συνδυασμούς ζευγών (S, j) για $j = 2, \ldots, N$, και για όλα τα δυνατά μη διατεταγμένα σύνολα S που ικανοποιούν τις ως άνω προδιαγραφές $|S| = k-1, 1 \notin S, j \notin S$.

Σε κάθε κόμβο απόφασης S, j μιας βαθμίδας $k = 1, \ldots, K-1$ υπάρχουν τόσες εναλλακτικές αποφάσεις για την μετάβαση στην επόμενη βαθμίδα $k+1$, όσος και ο αριθμός των κόμβων που μπορεί να επισκεφθεί ο πωλητής μετά τον κόμβο j δεδομένου ότι έχει

ήδη επισκεφθεί τους κόμβους του συνόλου S (και απαγορεύεται να τις ξαναεπισκεφθεί). Άρα ο αριθμός μεταβάσεων κάθε κόμβου της βαθμίδας k είναι $N-1-|S|-1$ (δηλαδή $N-1$ κόμβοι προς μετάβαση μείον αυτές του συνόλου S μείον ο κόμβος j), και αφού $|S|=k-1$, ο αριθμός μεταβάσεων είναι $N-k-1$. Το κόστος μετάβασης από τον κόμβο (S,j) της βαθμίδας k σε ένα ένα κόμβο (S',i) της βαθμίδας (k+1) είναι d_{ij}, δηλαδή όσο και το κόστος μετακίνησης από την πόλη j στην πόλη i, και ισχύει $S'=S\cup\{j\}$.

Ο απαιτούμενος αριθμός κόμβων κάθε βαθμίδας k για την επίλυση του προβλήματος TSP είναι:

$$(N-1)\left(\begin{array}{c} N-2 \\ k-1 \end{array}\right)$$

Άρα το σύνολο του αριθμού των κόμβων είναι:

$$(N-1)(N-2)!\sum_{k=0}^{N}[(k-1)!(N-k-1)!]^{-1}$$

Και το σύνολο των δυνατών εναλλακτικών διαδρομών του TSP είναι:

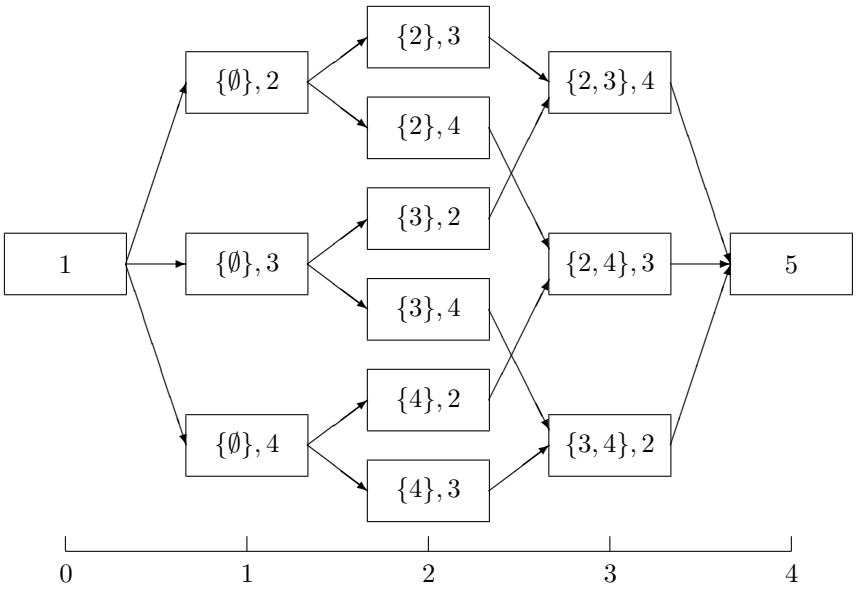
$$(N-1)!$$

Επειδή το διάνυσμα αυτό αποτελεί τμήμα ενός μεγαλύτερου διανύσματος, ο περιορισμός που επιβάλει την επιστροφή στον αρχικό κόμβο μετά από την επίσκεψη όλων των υπολοίπων κόμβων θα δημιουργούσε προβλήματα στην περίπτωση του μιμητικού αλγορίθμου. Έτσι δεν τον λαμβάνουμε υπ' όψιν. Επειδή ο πωλητής δεν επιστρέφει στην αφετηρία, αλλά καταλήγει στον τελικό κόμβο, στις παραπάνω σχέσεις χρησιμοποιούμε N', όπου $N=N'+1$.

Βλέπουμε ότι η πολυπλοκότητα του TSP σαν πρόβλημα απόφασης πολλαπλών βαθμίδων αυξάνεται εκθετικά με την προσθήκη νέων κόμβων. Για το λόγο αυτό επιλέξαμε να επιλέγεται με τυχαίο τρόπο ένα μικρό τμήμα του διανύσματος δρομολόγησης, πλήθους 5 στοιχείων. Για ένα διάνυσμα 6 στοιχείων θα είχαμε συνολικά 34 κόμβους και 24 πιθανές διαδρομές, ενώ για ένα διάνυσμα 7 στοιχείων θα είχαμε συνολικά 82 κόμβους και 120 πιθανές διαδρομές. Για τα 5 στοιχεία που χρησιμοποιούμε, το πρόβλημα έχει 10 κόμβους στο σύνολο και 6 πιθανές διαδρομές, με αποτέλεσμα να λύνεται αρκετά γρήγορα και εξοικονομώντας υπολογιστικό φόρτο, ενώ διορθώνονται τμήματα του διανύσματος δρομολόγησης με βέλτιστο τρόπο εξασφαλίζοντας καλύτερα γονίδια στο σύνολο του πληθυσμού.

Για να βρούμε τον αρχικό και τελικό κόμβο δημιουργούμε τον άνω τριγωνικό πίνακα D με διαστάσεις 5×5, όπου για i έχουμε τους κόμβους αφετηρίας και για j τους κόμβους προορισμού. Ο λόγος που υπολογίζουμε μόνο το άνω τριγωνικό τμήμα του πίνακα αντί για όλα τα στοιχεία του, είναι επειδή οι αποστάσεις μεταξύ των πελατών είναι ευκλείδειες και ισχύει $d_{ij}=d_{ji}$.

$$D=\left[\begin{array}{ccccc} 0 & d_{12} & d_{13} & d_{14} & d_{15} \\ & 0 & d_{23} & d_{24} & d_{25} \\ & & 0 & d_{34} & d_{35} \\ & & & 0 & d_{45} \\ & & & & 0 \end{array}\right]$$



Σχήμα 2.11: Γραφικό ανάπτυγμα του TSP για 5 κόμβους χωρίς επιστροφή στην αφετηρία.

Για το i του στοιχείου με τη μεγαλύτερη τιμή έχουμε τον κόμβο αφετηρίας και για το αντίστοιχο j έχουμε τον τελικό κόμβο, αφού πρόκειται για τους κόμβους με τη μεγαλύτερη απόσταση μεταξύ τους. Ο κόμβος i μετακινείται στην αρχή του τμήματος του διανύσματος δρομολόγησης και ο κόμβος j στο τέλος του. Ζητούμε το μονοπάτι που περνά από όλους τους υπόλοιπους κόμβους με αφετηρία και τέλος τους κόμβους που μόλις βρήκαμε με το μεγαλύτερο μήκος μεταξύ τους.

Το πρόβλημα θα είναι το ίδιο κάθε φορά όσον αφορά τους περιορισμούς και τους κόμβους από τους οποίους επιτρέπεται μετάβαση και αυτούς που δεν επιτρέπεται. Η διαφορά βρίσκεται στα κόστη μετάβασης d_{ij} από τον ένα κόμβο στον άλλο, αφού κάθε φορά το τμήμα του διανύσματος που επιλέγεται μπορεί να περιέχει άλλους πελάτες. Το ανάπτυγμα του προβλήματος μπορούμε να το δούμε στο σχήμα 2.11, όπου μπορούμε να δούμε τον τρόπο με τον οποίο είναι δυνατή η μετάβαση από τον ένα κόμβο-πελάτη στον επόμενο. Οι αριθμοί των κόμβων δεν αναφέρονται σε συγκεκριμένους πελάτες, αλλά στη θέση με την οποία τους συναντούμε στο τμήμα του διανύσματος δρομολόγησης που έχει επιλεχθεί.

Για την επίλυση του προβλήματος εφαρμόζουμε έμπροσθεν δυναμικό προγραμματισμό, ξεκινώντας από την βαθμίδα 0 και υπολογίζοντας το ελάχιστο κόστος από τον τρέχοντα κόμβο της βαθμίδας k σε αυτούς που είναι δυνατή η μετάβαση στη βαθμίδα $k + 1$. Για τους πελάτες στους οποίους μπορεί να γίνει μετάβαση από παραπάνω από έναν κόμβους σημειώνουμε την ακμή για την οποία το κόστος είναι το ελάχιστο, έτσι ώστε να μπορούμε να φτιάξουμε το βέλτιστο μονοπάτι που θα επισκέπτεται όλους τους πελάτες με το ελάχιστο μήκος συνολικά στο τέλος του αλγορίθμου.

Η αρχή του βέλτιστου δίνει τη βέλτιστη λύση στο προς επίλυση σύστημα, όμως δεν είναι απαραίτητο ότι το νέο διάνυσμα δρομολόγησης θα έχει μικρότερο κόστος από πριν. Κατά την αντικατάσταση του παλιού τμήματος με το νέο που φέρει τη λύση που μόλις

υπολογίστηκε, είναι πιθανό η νέα σειρά με την οποία διατάχθηκαν οι πέντε πελάτες να δίνει μεγαλύτερο κόστος σε σχέση με πριν. Δεδομένου ότι δεν λήφθηκαν υπ' όψιν οι γειτονικοί πελάτες και αν το σημείο του διανύσματος δρομολόγησης που επιλέχθηκε βρίσκεται σε σημείο που ξεκινά μια νέα διαδρομή, το νέο διάνυσμα γίνεται δεκτό μόνο στην περίπτωση που αποτελεί μέρος λύσης με μικρότερο κόστος σε σχέση με πριν.

2.4 Άλλες μέθοδοι

2.4.1 Λίστα περιορισμένης αναζήτησης (Tabu Search)

Όπως είδαμε οι μεθευρετικοί αλγόριθμοι είναι πολύ αποδοτικοί στην έρευνα του χώρου λύσεων συνδυαστικών προβλημάτων μεγάλης πολυπλοκότητας. Πέρα από τους γενετικούς και μιμητικούς αλγορίθμους που είναι το αντικείμενο αυτής της πτυχιακής εργασίας, ένας άλλος πολύ διαδεδομένος αλγόριθμος είναι αυτός της *Περιορισμένης Αναζήτησης*, ή αλλιώς Tabu Search. Ο αλγόριθμος της Περιορισμένης Αναζήτησης έχει χρησιμοποιηθεί με μεγάλη επιτυχία για την επίλυση τόσο προβλημάτων που αφορούν την χωροθέτηση εγκαταστάσεων, όσο και σε διάφορες μορφές του VRP. Έχοντας λοιπόν σοβαρές ενδείξεις για την επιτυχία της μεθόδου στο LRP, ο Tabu Search έχει προταθεί ως μέθοδος επίλυσης αυτού του προβλήματος από διάφορους ερευνητές [8] [9].

Ο Tabu Search ως μεθευρετικός αλγόριθμος χρησιμοποιεί κάποιο μηχανισμό για την αποφυγή του εγκλωβισμού σε τοπικό ελάχιστο και την αποδοτική αναζήτηση στο χώρο λύσεων. Η διαφορά με τους υπόλοιπους μεθευρετικούς αλγορίθμους είναι ότι αυτός ο μηχανισμός δεν χρησιμοποιεί την τύχη όπως είδαμε στους γενετικούς και τους μιμητικούς, αλλά την απομνημόνευση των κινήσεων που έχουν γίνει στο παρελθόν. Κατά την εκτέλεση του TS γίνεται μετακίνηση από μια λύση στην καλύτερη γειτονική, ακόμα και αν αυτό σημαίνει χειροτέρευση σε σχέση με την καλύτερη τρέχουσα. Αυτή η στρατηγική επιτρέπει στην αναζήτηση να ξεφύγει από το τρέχον τοπικό βέλτιστο και να ερευνήσει άλλες περιοχές του χώρου λύσεων. Για να αποφευχθεί η μετάβαση σε κάποια λύση που έχει ήδη ελεγχθεί, όλες οι πρόσφατες κινήσεις που έχουν γίνει για την μετάβαση από μια λύση σε μια άλλη καταγράφονται σε μια λίστα περιορισμένου μεγέθους. Αυτό έχει σαν αποτέλεσμα την αποφυγή επανάληψης των ίδιων κινήσεων ξανά και ξανά, εξαλείφοντας τον κίνδυνο προσκόλλησης σε συγκεκριμένη περιοχή του χώρου λύσεων.

Πιο αναλυτικά, ο αλγόριθμος TS χρησιμοποιεί δύο στρατηγικές για την αποδοτική αναζήτηση του χώρου λύσεων. Αυτές οι στρατηγικές είναι η *εντατικοποίηση (intensification)* της αναζήτησης γύρω από ένα τοπικό βέλτιστο και την *διάχυση (diversification)* της έρευνας σε νέες περιοχές του εφικτού συνόλου. Ο συγκεκριμένος αλγόριθμος είναι μια τεχνική καθοδήγησης (guiding technique) της τοπικής αναζήτησης βασισμένη σε προσαρμοστική μνήμη, δίνοντας στην περιορισμένη αναζήτηση της ιδιότητες μιας μεθευρετικής μεθόδου. Είναι μέθοδος προσδιοριστικού (deterministic) χαρακτήρα που προσπαθεί να επιτύχει ό,τι θεωρητικά υπόσχεται η προσομοιωμένη ανόπτηση (simulated annealing) που είναι μέθοδος τυχαιοποίησης (στοχαστικού χαρακτήρα).

Έτσι τα βασικά χαρακτηριστικά της μεθόδου μπορούν να συνοψιστούν στα εξής τέσσερα σημεία:

- **Μνήμη**, ℌ, βασισμένη στην ιστορία της τοπικής αναζήτησης σε αντίθεση προς την τοπική αναζήτηση που δεν διαθέτει μνήμη.

- **Περιορισμοί Tabu**, που βασιζόμενοι στην μνήμη, απαγορεύουν συγκεκριμένες κινήσεις (κυρίως στοιχειώδεις πράξεις σε λύσεις για μετάβαση σε άλλες): $N(\mathfrak{H}, x^{(k)}) \not\equiv N(x^{(k)})$, π.χ. $N(\mathfrak{H}, x^{(k)}) \subset N(x^{(k)})$.
- **Κριτήρια φιλοδοξίας** (aspiration), που επιτρέπουν υπέρβαση των περιορισμών της μεθόδου, π.χ. εάν η χρήση κάποιων απαγορευμένων κινήσεων παρήγαγαν λύση καλύτερη από όλες που έχουν παραχθεί μέχρι εκείνη τη στιγμή, π.χ. $f(\mathfrak{H}, x) \neq f(x)$.
- **Κριτήρια διάχυσης** (diversification), που επιτρέπουν την επιβολή ιδιοτήτων λύσεων που ιστορικά παρήγαγαν καλά αποτελέσματα και την διάχυση της αναζήτησης σε νέες περιοχές εφικτών λύσεων: Η $N(\mathfrak{H}, x^{(k)})$ μπορεί να περιέχει λύσεις εκτός $N(x^{(k)})$.

Το κυριότερο χαρακτηριστικό που ενσωματώθηκε στον αλγόριθμο που κατασκευάσαμε από την μέθοδο περιορισμένης αναζήτησης, ήταν αυτό της μνήμης με τη μορφή μιας λίστας περιορισμένης αναζήτησης (Tabu List) και των tabu περιορισμών. Η χρήση αυτού του χαρακτηριστικού έγινε με τη δημιουργία μιας δομής δεδομένων τύπου ουράς, όπου κάθε στοιχείο της περιέχει τα τρία διανύσματα με τα οποία κωδικοποιούμε την κάθε λύση. Η λίστα αρχικά είναι άδεια και κάθε φορά που ο γενετικός αλγόριθμος πραγματοποιεί την διαδικασία διασταύρωσης δύο λύσεων (ενός ή δύο σημείων) ελέγχει εάν κάποια από τις λύσεις που προέκυψαν, υπάρχει στη λίστα περιορισμένης αναζήτησης. Στην περίπτωση που συμβαίνει αυτό, οι δύο νέες λύσεις-απόγονοι απορρίπτονται και επιλέγονται από την αρχή δύο νέοι γονείς προς διασταύρωση και γίνεται επανάληψη της διαδικασίας. Στην αντίθετη περίπτωση οι δύο νέες λύσεις προστίθενται στο τέλος της λίστας. Εάν η λίστα είναι γεμάτη τότε οι πιο παλιές λύσεις της λίστας διαγράφονται. Με αυτό τον τρόπο η λίστα διατηρεί ένα σταθερό μέγεθος κατά την εκτέλεση του αλγορίθμου, δηλαδή ο ορίζοντας των κινήσεων που "βλέπει" είναι περιορισμένος. Το μέγεθος της λίστας προσαρμόζεται σύμφωνα με το μέγεθος του προβλήματος και ο μέγιστος αριθμός των στοιχείων που μπορεί να περιέχει ορίζεται στο τετραπλάσιο πλήθος στοιχείων από αυτό των πελατών.

Ο έλεγχος αυτός γίνεται μόνο κατά τη διαδικασία διασταύρωσης του διανύσματος δρομολόγησης. Αυτό συμβαίνει επειδή πάρα πολλές καλές λύσεις μοιράζονται τα διανύσματα αποθηκών και δείκτη, ενώ μια νέα λύση δεν είναι ολοκληρωμένη πριν από τη δημιουργία του νέου διανύσματος δρομολόγησης. Έτσι η εισαγωγή των διανυσμάτων σε ξεχωριστές λίστες δεν έχει νόημα, αφού μόνο με τη χρήση και των τριών μπορεί να περιγραφεί μια λύση ολοκληρωμένα, ενώ πριν ολοκληρωθεί η διαδικασία της διασταύρωσης των διανυσμάτων που περιέχουν τη σειρά επίσκεψης των πελατών δεν υπάρχουν σχηματισμένες νέες λύσεις και στις δύο μεθόδους επίλυσης (δύο φάσεις ταυτόχρονα ή ξεχωριστά).

Ο τρόπος με τον οποίο απορρίπτονται ή γίνονται αποδεκτές οι λύσεις που προκύπτουν από τις μεθόδους τοπικής αναζήτησης μοιάζει αρκετά με τον τρόπο που λειτουργεί το κριτήριο φιλοδοξίας στον Tabu Search. Οι λύσεις που προκύπτουν από αυτές τις μεθόδους γίνονται αποδεκτές μόνο εάν δίνουν καλύτερο κόστος σε σχέση με τις αρχικές, οπότε δεν έχει νόημα ο έλεγχος στην Tabu List, ενώ στην περίπτωση που δίνουν χειρότερο κόστος σε σχέση με πριν, απορρίπτονται ούτως ή άλλως με τις αρχικές να υπάρχουν ήδη στη λίστα.

Το κριτήριο διάχυσης το αναλαμβάνει ο τρόπος με τον οποίο δουλεύει ο γενετικός αλγόριθμος. Όπως είδαμε πιο πριν, οι καλύτερες λύσεις έχουν περισσότερες πιθανότητες να αναπαραχθούν, άρα και η πιθανότητα ύπαρξης των γονιδίων τους στις νέες γενιές

του πληθυσμού είναι μεγαλύτερη. Οι διαδικασίες της μετάλλαξης και ο στοχαστικός χαρακτήρας των εξελικτικών στρατηγικών που χρησιμοποιούνται εξασφαλίζουν την αναζήτηση όλου του χώρου λύσεων είτε γίνεται χρήση της λίστας περιορισμένης αναζήτησης είτε όχι.

Η υλοποίηση της λίστας περιορισμένης αναζήτησης έγινε με τη χρήση δύο βοηθητικών συναρτήσεων και ενός πίνακα (array) σταθερού μεγέθους με μια μεταβλητή δείκτη, η οποία μας δείχνει το τρέχον στοιχείο - κεφαλή της λίστας. Αρχικά η λίστα είναι κενή. Με τη βοήθεια μιας συνάρτησης εισαγωγής στοιχείου, εκχωρείται το στοιχείο που δίνουμε στο σημείο που μας δείχνει η μεταβλητή και η τιμή της αυξάνεται κατά μια μονάδα έτσι ώστε να δείχνει την επόμενη θέση στη λίστα. Όταν αυτή η μεταβλητή πάρει τη μέγιστη τιμή της (μέγεθος λίστας), τότε της δίνεται η κατάλληλη τιμή ώστε να δείχνει στο πρώτο στοιχείο του πίνακα. Με αυτό τον τρόπο η μεταβλητή δείχνει πάντα το στοιχείο που βρίσκεται τον περισσότερο χρόνο στη λίστα, δημιουργώντας μια δομή δεδομένων τύπου ουράς, όπου επικρατεί η συνθήκη first in first out. Μπορούμε να δούμε πως λειτουργεί με μορφή ψευδοκώδικα:

```
method tabulistinsert
input tabulist, tbsize, tbstart, item

tabulist(tbstart) ⟵ item
if tbstart = tbsize, then
   tbstart ⟵ 1
else
   tbstart ⟵ tbstart + 1
end if

return tabulist, tbstart
```

Η αναζήτηση στη λίστα γίνεται επίσης με τη χρήση μιας βοηθητικής συνάρτησης. Στη συνάρτηση δίνεται το στοιχείο προς αναζήτηση και στη συνέχεια γίνεται αναζήτηση στη λίστα για την ύπαρξη αυτού του στοιχείου. Αν η αναζήτηση ήταν επιτυχής και βρέθηκε το ζητούμενο στοιχείο η συνάρτηση επιστρέφει την τιμή 1, ενώ σε διαφορετική περίπτωση την τιμή 0. Εδώ παρουσιάζεται η συνάρτηση αναζήτησης σε μορφή ψευδοκώδικα:

```
method tabulistsearch
input tabulist, tbsize, searchitem

value ⟵ 0
for i: 1→tbsize
   if tabulist(i) = searchitem, then
      returnvalue ⟵ 1
      exit for loop
   end if
end for

return value //returns value 1 if search item in list, 0 otherwise.
```

Στη συνέχεια παρουσιάζεται σε μορφή ψευδοκώδικα ο τρόπος με τον οποίο συνδυάζεται η λίστα περιορισμένης αναζήτησης στον αλγόριθμο κατά τη διαδικασία διασταύρωσης λύσεων.

```
method Tabu List implementation
input oldgeneration, tabulist, tbstart, populationsize

i ⟵ 1
do
   parent1 ⟵ roulette(oldgeneration)
   parent2 ⟵ roulette(oldgeneration)
   offspring1, offspring2 ⟵ crossover(parent1, parent2)
   if tabulistsearch(offspring1) = 0, then //Not in list
      if tabulistsearch(offspring2) = 0, then //Not in list
         tabulistinsert(offspring1)
         newgeneration(i) ⟵ offspring1
         tabulistinsert(offspring2)
         newgeneration(i+1) ⟵ offspring2
         i ⟵ i+2
      end if
   end if
while (i ≤ populationsize)

return newgeneration
```

Για προβλήματα με μεγάλο πλήθος πελατών η Tabu List μπορεί να πάρει πολύ μεγάλο μέγεθος. Ακόμη, τα προβλήματα αυτά χρησιμοποιούν μεγαλύτερα διανύσματα λόγω του μεγάλου αριθμού των πελατών που πρέπει να εξυπηρετήσουν. Έτσι είναι επόμενο η αναζήτηση στη λίστα περιορισμένης αναζήτησης να καθυστερεί τον αλγόριθμο σε βάρος της παραγωγής νέων γενιών και έτσι δεν είναι εξασφαλισμένη η επίτευξη καλύτερων αποτελεσμάτων σε σχέση με την περίπτωση που δεν γίνεται χρήση της. Όπως θα δούμε παρακάτω η επιτυχία ή όχι του συνδυαστικού αλγορίθμου εξαρτάται από τη φύση του προβλήματος.

Κεφάλαιο 3

Πειραματικά αποτελέσματα

3.1 Γενικά

Αρχικά η πτυχιακή αυτή εργασία ξεκίνησε με στόχο την επίλυση του προβλήματος με τη χρήση γενετικών αλγορίθμων. Στη διαθέσιμη βιβλιογραφία μπορούμε να δούμε ότι και οι δύο μέθοδοι επίλυσης (ταυτόχρονη ή ξεχωριστή επίλυση των δύο φάσεων) προτιμούνται εξ ίσου από τους ερευνητές. Στη δικιά μας περίπτωση επιλέχθηκε αρχικά η υλοποίηση του προγράμματος με τη χρήση της μεθόδου επίλυσης των δύο φάσεων ταυτόχρονα. Όμως, παρατηρήθηκε ότι η μετάβαση σε νέες βέλτιστες λύσεις δεν γινόταν αρκετά γρήγορα. Έτσι χρησιμοποιήσαμε και τοπική αναζήτηση στις διαδρομές προκειμένου να διερευνούνται μερικώς ή πλήρως οι διαθέσιμες λύσεις. Έτσι πλέον η επίλυση του προβλήματος δεν γινόταν με γενετικούς αλγορίθμους, αλλά με μιμητικούς. Με αυτό τον τρόπο μπορέσαμε να πετύχουμε πολύ πιο χαμηλές λύσεις από άποψη κόστους. Όμως αρκετά συχνά απαιτούνταν πολλές γενιές για την επίτευξη αυτών των λύσεων, ειδικά όσο χαμηλότερα βρισκόταν η τρέχουσα βέλτιστη λύση. Έτσι υλοποιήθηκε και η επίλυση του προβλήματος σε δύο φάσεις, όπου το εκάστοτε υπό-πρόβλημα επιλυόταν ξεχωριστά. Όπως θα δούμε και παρακάτω δεν υπάρχει κάποια κατάλληλη μέθοδος που να οδηγεί πάντα σε καλύτερα αποτελέσματα. Αναλόγως το πρόβλημα που αντιμετωπίζεται, η κάθε μέθοδος επίλυσης μπορεί να φέρει διαφορετικά αποτελέσματα και κάποια από τις δύο να υστερεί σε σχέση με την άλλη.

Οι πιθανότητες με τις οποίες εκτελούνταν οι διαδικασίες των γενετικών αλγορίθμων έχουν οριστεί εμπειρικά. Συγκεκριμένα η πιθανότητα να συμβεί crossover έχει οριστεί στο 90%, ενώ τα ενδεχόμενα να γίνει crossover ενός η δύο σημείων έχουν την ίδια πιθανότητα, δηλαδή 50%.

Ωστόσο, η επιλογή του συνδυασμού των πιθανοτήτων που θα οδηγεί σε μετάλλαξη των διανυσμάτων ήταν κρίσιμη. Αν γινόταν μετάλλαξη πιο συχνά απ'ότι έπρεπε τότε με τη συχνή εφαρμογή της διαδικασίας καλές λύσεις θα γίνονταν χειρότερες και συνολικά η διαδικασία αυτή θα δημιουργούσε προβλήματα στην εκτέλεση του προγράμματος αντί να βοηθά στην επίλυσή του. Αντίθετα, αν η πιθανότητες με τις οποίες γινόταν μετάλλαξη ήταν πιο μικρές από το βέλτιστο, τότε το πρόγραμμα θα κινδύνευε να κολλήσει σε κάποιο τοπικό ελάχιστο καθώς δε θα έμπαιναν νέα γονίδια αρκετά συχνά στον πληθυσμό. Επίσης, επειδή υπήρχε το ενδεχόμενο η μια πιθανότητα να επηρεάζει την άλλη, δοκιμάστηκε ο κάθε συνδυασμός πιθανοτήτων. Το πρόβλημα που επιλέχθηκε

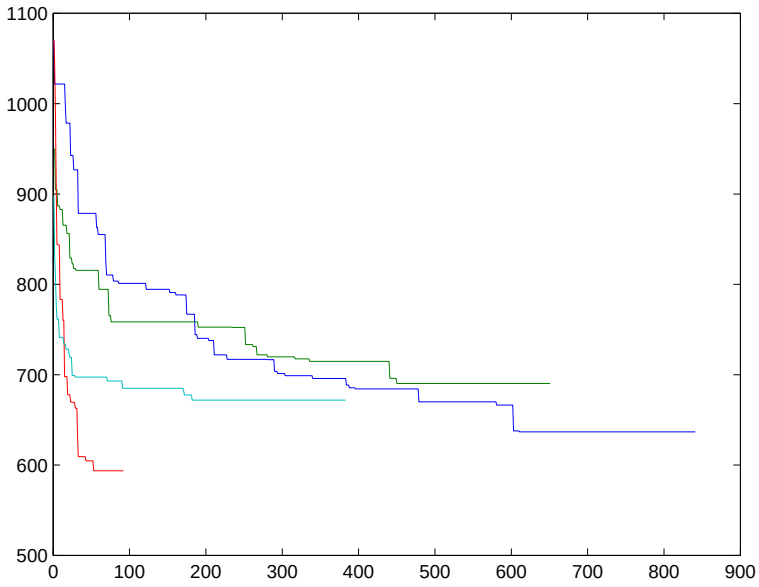
να δοκιμαστεί ο εκάστοτε συνδυασμός είναι το βιβλιογραφικό πρόβλημα αναφοράς *Gaskell67-22x5*. Το συγκεκριμένο πρόβλημα επιλέχθηκε επειδή είναι μικρό σε μέγεθος και εύρεση λύσης με κόστος μικρότερο από το upper bound γινόταν αρκετά γρήγορα. Η πιθανότητα να εκτελεστεί κάθε διαδικασία μετάλλαξης (μετάλλαξη διανύσματος αποθήκης, μετάλλαξη διανύσματος δείκτη και μετάλλαξη διανύσματος δρομολόγησης) δοκιμάστηκε για τις τιμές 1%-20%. Για κάθε πιθανό συνδυασμό αυτών των πιθανοτήτων επιλύθηκε το πρόβλημα 50 φορές. Για τη μείωση του υπολογιστικού κόστους και του χρόνου εκτέλεσης, η τιμή της κάθε μεταβλητής που αντιστοιχούσε σε πιθανότητα μετάλλαξης αυξανόταν με βήμα 2. Η εκτέλεση του προγράμματος σταματούσε αν η τρέχουσα βέλτιστη λύση ήταν μικρότερη του upper bound ή είχαν περάσει 800 επαναλήψεις χωρίς βελτίωση. Οι δύο φάσεις του προβλήματος επιλύθηκαν ταυτόχρονα, αφού οι τιμές των μεταβλητών αυτών δεν φαίνεται να επηρεάζουν την εκτέλεση με τη χρήση της εκάστοτε μεθόδου από τη στιγμή που χρησιμοποιούνται οι ίδιες διαδικασίες και στις δύο. Έτσι το πρόβλημα επιλύθηκε συνολικά 50000 φορές και επιλέχθηκε ο συνδυασμός των μεταβλητών που είχε το χαμηλότερο μέσο όρο κόστους και συνόλου επαναλήψεων (γενιών).

Για τις επιμέρους διαδικασίες μετάλλαξης οι πιθανότητες ορίστηκαν εμπειρικά. Και με βάση αυτές τις τιμές έγινε η παραπάνω διερεύνηση. Έτσι για για τη μετάλλαξη με αντιστροφή του διανύσματος δρομολόγησης έχουμε πιθανότητα 30%, ενώ για να συμβεί απλή αντιμετάθεση δύο τυχαίων σημείων έχουμε προφανώς πιθανότητα 70%.

3.2 Σύγκριση μεθόδων επίλυσης

Για τη σύγκριση των δύο μεθόδων με και χωρίς τοπική αναζήτηση τρέξαμε το πρόγραμμα που κατασκευάσαμε για τις τέσσερις αυτές περιπτώσεις. Το βιβλιογραφικό πρόβλημα που επιλύθηκε είναι το *Gaskell67-22x5* του οποίου το upper bound ορίζεται στην τιμή 591.1. Στον κατακόρυφο άξονα απεικονίζεται το καλύτερο fitness κάθε γενιάς, ενώ στον οριζόντιο άξονα ο αύξων αριθμός των γενιών. Η περίπτωση που το πρόβλημα επιλύεται χωρίς τη χρήσης τοπικής αναζήτησης απεικονίζεται με μπλε χρώμα. Η επίλυση του προβλήματος με τη δεύτερη μέθοδο χωρίς να γίνεται πάλι χρήση της τοπικής αναζήτησης απεικονίζεται με πράσινο χρώμα. Για την περίπτωση που η λύση γίνεται με τη μέθοδο 1 με τον μιμητικό αλγόριθμο η γραμμή στα σχήματα 3.1 και 3.2 έχει κόκκινο χρώμα. Με γαλάζιο χρώμα απεικονίζεται η λύση του προβλήματος με χρήση τοπικής αναζήτησης σε συνδυασμό με τη μέθοδο 2. Η διαφορά μεταξύ των σχημάτων 3.1 και 3.2 είναι ότι στο δεύτερο απεικονίζονται λιγότερες γενιές για να φαίνεται με μεγαλύτερη λεπτομέρεια το αρχικό κόστος κατά την εκκίνηση του αλγορίθμου και ο ρυθμός με τον οποίο γίνεται η μετάβαση σε χαμηλότερες λύσεις.

Για τις δύο περιπτώσεις όπου το πρόβλημα λύνεται σε δύο φάσεις ο αριθμός των γενιών μετρά από την έναρξη της επίλυσης του υπό-προβλήματος της δρομολόγησης. Αυτό έγινε γιατί ο υπολογισμός του fitness αυτών των μεθόδων ξεκίνησε μόνο αφού επιλύθηκε το υπό-πρόβλημα χωροθέτησης εγκαταστάσεων. Πριν από αυτό η τιμή του fitness αντιπροσώπευε την απόσταση από και προς τον εκάστοτε πελάτη που ανήκε σε κάθε ανοιχτή αποθήκη και δεν μπορεί να συγκριθεί με την τιμή fitness μιας λύσης ολόκληρου του προβλήματος LRP. Για αυτό το λόγο η γραφική παράσταση αυτών των μεθόδων θα βρισκόταν μετατοπισμένη ως προς την έναρξη του οριζόντιου άξονα και δε θα ήταν εύκολη η σύγκριση των γραφημάτων. Επίσης για την υλοποίηση των γραφημάτων χρησιμοποιούνται οι τιμές fitness για τις οποίες υπήρξε βελτίωση σε σχέση με τις



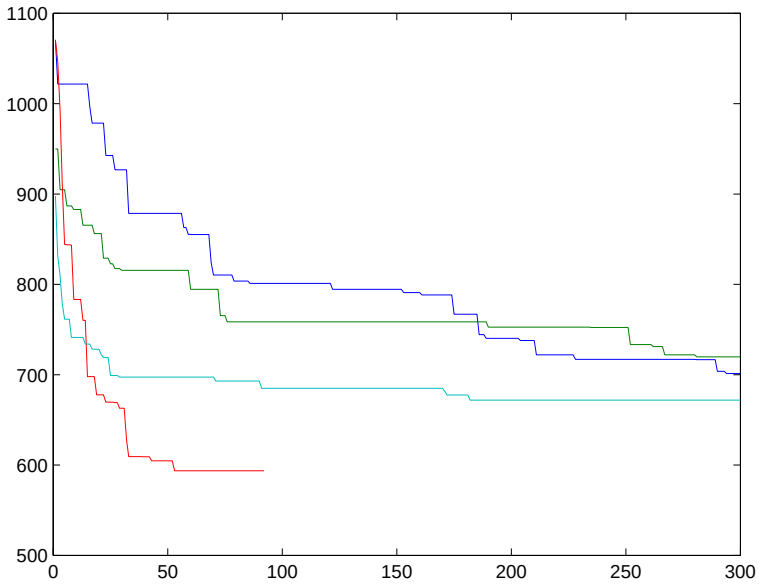
Σχήμα 3.1: Απεικόνιση γενιών 1-841

τιμές των προηγούμενων γενιών. Έτσι μπορούμε να κρίνουμε πόσο γρήγορα κλίνουμε προς νέες καλύτερες λύσεις. Αξίζει να σημειωθεί ότι για τη μέθοδο 2 με χρήση τοπικής αναζήτησης χρειάστηκαν 207 γενιές για την επίλυση του προβλήματος χωροθέτησης, ενώ για την περίπτωση που το πρόβλημα λύθηκε χωρίς τοπική αναζήτηση χρειάστηκαν 188 γενιές. Η συνθήκη εξόδου για την επίλυση του FLP ήταν 150 επαναλήψεις χωρίς βελτίωση. Η τοπική αναζήτηση εφαρμόζεται μόνο κατά τη φάση της δρομολόγησης οπότε ο αριθμός γενιών που χρειάστηκαν δεν επηρεάστηκε από αυτό τον παράγοντα.

Βλέπουμε στο σχήμα 3.1 ότι με τη χρήση τοπικής αναζήτησης μειώνεται σημαντικά το πλήθος των απαιτούμενων γενιών για τη βελτίωση της καλύτερης λύσης σε σύγκριση με τη χρήση μόνο γενετικών αλγόριθμων. Αυτό όμως δεν σημαίνει και μείωση του συνολικού υπολογιστικού φόρτου, αφού η τοπική αναζήτηση σε κάθε λύση κάθε γενιάς αποτελεί μια ιδιαίτερα απαιτητική διαδικασία. Παρ'όλα αυτά με τη χρήση της μπορούν να βρεθούν λύσεις που δε θα μπορούσαν να βρεθούν αλλιώς δεδομένου των βαθμών ελευθερίας του προβλήματος.

Στο σχήμα 3.2 παρατηρούμε ότι με τη χρήση της δεύτερης μεθόδου, το κόστος που έχει η καλύτερη λύση του αρχικού πληθυσμού είναι χαμηλότερη από αυτή της πρώτης μεθόδου και στις δύο περιπτώσεις (με ή χωρίς Local Search). Αυτό είναι αναμενόμενο αφού έχοντας λύσει το πρόβλημα της χωροθέτησης οι πελάτες έχουν αντιστοιχηθεί σε πιο κοντινές τους αποθήκες. Επομένως, τα τυχαία διανύσματα δρομολόγησης που χρησιμοποιούνται κατά την εκκίνηση του αλγορίθμου είναι πιο πιθανό να φέρουν μικρότερα κόστη σε σχέση με τα αντίστοιχα της πρώτης μεθόδου, όπου εκεί η χωροθέτηση και η αντιστοίχηση γίνεται με τυχαίο τρόπο.

Σημαντική παρατήρηση μπορεί να γίνει επίσης όσον αφορά την κλίση των γραφικών παραστάσεων. Βλέπουμε και στα δύο σχήματα ότι η μετάβαση σε χαμηλότερα κόστη



Σχήμα 3.2: Απεικόνιση γενιών 1-300

γίνεται πολύ πιο γρήγορα με τη χρήση τοπικής αναζήτησης. Ο γενετικός αλγόριθμος εκτελείται πιο γρήγορα από τον μιμητικό, αλλά αυτή η κατανομή πόρων στις επιπλέον διαδικασίες του δεύτερου μας ανταμείβει με λύσεις χαμηλότερου κόστους και σε λιγότερες γενιές. Άρα ενώ ο ρυθμός των παραγόμενων γενιών στον μιμητικό είναι πιο μικρός, οι λύσεις που επιτυγχάνονται είναι καλύτερες και από άποψη κόστους και από άποψη χρόνου.

Το συγκεκριμένο πρόβλημα που επιλέχθηκε προς επίλυση αποτελεί μια πολύ ενδιαφέρουσα περίπτωση. Όπως θα δούμε παρακάτω με το να επιλύουμε ξεχωριστά τις δύο φάσεις του LRP καταλήγουμε συχνότερα σε χαμηλές λύσεις, όμως πάντα έχουμε τον κίνδυνο η λύση στην οποία θα καταλήξουμε να είναι υποβέλτιστη με αδυναμία περαιτέρω βελτίωσης. Έτσι σε αυτό το πρόβλημα αντιμετωπίζουμε μια τέτοια περίπτωση. Αν ανατρέξουμε στο παράρτημα στην γραφική παράσταση των καλύτερων λύσεων που έχουν επιτευχθεί για κάθε μέθοδο για το εκάστοτε πρόβλημα, στην περίπτωση μας το πρόβλημα 7, θα δούμε ότι με το να λύσουμε ταυτόχρονα τις δύο φάσεις του LRP (βλ. Σχήμα 5.13), όλοι οι πελάτες εξυπηρετούνται από μόνο μια ανοιχτή αποθήκη, σε αντίθεση με την επίλυση των δύο φάσεων ξεχωριστά (βλ. Σχήμα 5.14) όπου εξυπηρετούνται από δύο. Έτσι με τη χρήση της πρώτης μεθόδου μπορούμε και βρίσκουμε λύση ίση με το lower bound που έχει οριστεί για αυτό το πρόβλημα (585.1 μονάδες). Με τη δεύτερη μέθοδο δεν καταφέρνουμε να ξεπεράσουμε ούτε το upper bound αφού η καλύτερη λύση που επιτεύχθηκε είχε κόστος 629.07 μονάδων.

3.3 Προβλήματα βιβλιογραφίας

Όπως είδαμε προηγουμένως μπορούμε να πετύχουμε καλύτερα αποτελέσματα με τη χρήση του μιμητικού αλγορίθμου. Έτσι με τη χρήση αυτού του αλγορίθμου επιλύσαμε ένα σετ από βιβλιογραφικά προβλήματα αναφοράς [14] επιλύοντας τις δύο φάσεις ταυτόχρονα (μέθοδος 1) και ξεχωριστά (μέθοδος 2). Τα προβλήματα αυτά λύθηκαν με τη χρήση του αλγορίθμου που περιγράψαμε στα προηγούμενα κεφάλαια. Για την επίτευξή τους έγινε χρήση ενός υπολογιστή PC με επεξεργαστή AMD Athlon 64 3200+ του 2004. Το πρόγραμμα κατασκευάστηκε στο προγραμματιστικό περιβάλλον Matlab r2009b. Μπορούμε να δούμε τα καλύτερα αποτελέσματα που επιτεύχθηκαν με κάθε μέθοδο στο εκάστοτε πρόβλημα στους πίνακες 3.1,3.2. Τα αποτελέσματα που είναι σημειωμένα με έντονα γράμματα έχουν επιτευχθεί με τη χρήση της λίστας περιορισμένης αναζήτησης.

Στον πίνακα 3.3 μπορούμε να συγκρίνουμε τις καλύτερες τιμές που επιτεύχθηκαν από τον αλγόριθμο που κατασκευάσαμε σε σχέση με τις καλύτερες τιμές που έχουν επιτευχθεί στη βιβλιογραφία [14][15][16].

Συγκρίνοντας τις δύο μεθόδους με τη χρήση των πινάκων 3.1 και 3.2 παρατηρούμε ότι καταλήξαμε περισσότερες φορές σε χαμηλή λύση με τη χρήση της μεθόδου 2. Όμως σε ορισμένα από τα προβλήματα αναφοράς συναντήσαμε το πρόβλημα το οποίο περιγράφεται στη βιβλιογραφία, όπου η καλύτερη λύση που μπορεί να επιτευχθεί με αυτή τη μέθοδο είναι υπό-βέλτιστη. Χαρακτηριστικά μπορούμε να δούμε με τη χρήση των παραπάνω πινάκων και της γραφικής απεικόνισής τους στο παράρτημα πως στα προβλήματα 7,9 και 10 καταλήξαμε σε τοπικό ελάχιστο με αδυναμία περαιτέρω βελτίωσης της λύσης λόγω της συγκεκριμένης επιλογής των ανοιχτών εγκαταστάσεων. Ενώ η λύση του προβλήματος χωροθέτησης ήταν βέλτιστη, λόγω της κατανομής των πελατών και των πρόσθετων περιορισμών του LRP καταλήξαμε σε υπό-βέλτιστη λύση. Αντίθετα με τη χρήση της μεθόδου 1 δεν υφίσταται τέτοιος κίνδυνος. Καθώς όμως πλησιάζουμε σε περιοχή με λύσεις χαμηλότερου κόστους οι περισσότεροι βαθμοί ελευθερίας εμποδίζουν την εξέλιξη προς νέες λύσεις χαμηλότερου κόστους, αφού οι πιθανότητες δημιουργίας μιας νέας λύσης με μεγαλύτερο κόστος είναι περισσότερες. Για αυτό το λόγο έχουν αναπτυχθεί διάφορες τεχνικές ώστε να προσαρμόζεται ο αλγόριθμος σε αυτές τις συνθήκες όπως για παράδειγμα η δυναμική μεταβολή της πιθανότητας μετάλλαξης των κωδικοποιημένων λύσεων (βλ. [4]). Όμως κάτι τέτοιο δεν ήταν στα πλαίσια αυτής της πτυχιακής εργασίας και αποτελεί ένα σημείο για μελλοντική έρευνα.

S/N	Problem	Vehicle Capacity	Lower Bound	Upper Bound	Method 1	Time (sec)	Deviation
1	Christofides69-50x5	160	549.4	582.7	595.28	107.089	2.16%
2	Christofides69-75x10	140	744.7	886.3	902.58	1402.83	1.84%
3	Christofides69-100x10	200	788.6	889.4	910.43	1817.14	2.36%
4	Daskin95-88x8	9000000	356.4	384.9	365.28	405.67	-5.10%
5	Daskin95-150x10	8000000	43406	46642.7	**54537.14**	2128.67	16.93%
6	Gaskell67-21x5	6000	424.9	435.9	424.9	157.16	-2.52%
7	Gaskell67-22x5	4500	585.1	591.5	**585.11**	4.522	-1.08%
8	Gaskell67-29x5	4500	512.1	512.1	**532.93**	76.62	4.07%
9	Gaskell67-32x5	8000	556.5	571.7	**562.22**	27.86	-1.66%
10	Gaskell67-32x5	11000	504.3	511.4	504.33	11.75	-1.38%
11	Gaskell67-36x5	250	460.4	470.7	**470.42**	48.27	-0.06%
12	Min92-27x5	2500	3062	3062	**3062.02**	65.61	0.00%
13	Min92-134x8	850		6238	**7168.86**	735.91	14.92%
14	Perl83-12x2	140	204	204	204	0.701	0.00%
15	Perl83-55x15	120	1074.8	1136.2	1148.31	167.27	1.07%
16	Perl83-85x7	160	1568.1	1656.9	1698.7	135 .86	2.52%
17	Or76-117x14	150	12048.4	12474.2	13690.56	655.58	9.75%

Πίνακας 3.1: Results of benchmark instances using method 1.

S/N	Problem	Vehicle Capacity	Lower Bound	Upper Bound	Method 2	Time (sec)	Deviation
1	Christofides69-50x5	160	549.4	582.7	586.03	98.44	0.57%
2	Christofides69-75x10	140	744.7	886.3	905.28	196.39	2.14%
3	Christofides69-100x10	200	788.6	889.4	899.19	1031.31	1.10%
4	Daskin95-88x8	9000000	356.4	384.9	370.08	255.08	-3.85%
5	Daskin95-150x10	8000000	43406	46642.7	47256.88	437.97	1.32%
6	Gaskell67-21x5	6000	424.9	435.9	424.9	16.939	-2.52%
7	Gaskell67-22x5	4500	585.1	591.5	615.9	54.82	4.13%
8	Gaskell67-29x5	4500	512.1	512.1	**512.1**	23.937	0.00%
9	Gaskell67-32x5	8000	556.5	571.7	585.18	94.7	2.36%
10	Gaskell67-32x5	11000	504.3	511.4	535.58	59.54	4.73%
11	Gaskell67-36x5	250	460.4	470.7	472.07	25.78	0.29%
12	Min92-27x5	2500	3062	3062	3062	6.723	0.00%
13	Min92-134x8	850		6238	**6011.2**	1032.59	-3.64%
14	Perl83-12x2	140	204	204	204	1.552	0.00%
15	Perl83-55x15	120	1074.8	1136.2	1117.5	110.79	-1.65%
16	Perl83-85x7	160	1568.1	1656.9	1668.15	335.73	0.68%
17	Or76-117x14	150	12048.4	12474.2	**12909.66**	771.01	3.49%

Πίνακας 3.2: Results of benchmark instances using method 2.

S/N	Problem	Vehicle capacity	BKS	Mimetic algorithm	Deviation
1	Christofides69-50x5	160	565.6	586.03	3.61%
2	Christofides69-75x10	140	850.8	902.58	6.09%
3	Christofides69-100x10	200	833.4	899.19	7.89%
4	Daskin95-88x8	9000000	355.8	365.28	2.66%
5	Daskin95-150x10	8000000	43963.6	47256.88	7.49%
6	Gaskell67-21x5	6000	424.9	424.9	0.00%
7	Gaskell67-22x5	4500	585.1	585.11	0.00%
8	Gaskell67-29x5	4500	512.1	512.1	0.00%
9	Gaskell67-32x5	8000	562.2	562.22	0.00%
10	Gaskell67-32x5	11000	504.3	504.33	0.01%
11	Gaskell67-36x5	250	460.4	470.42	2.18%
12	Min92-27x5	2500	3062	3062.02	0.00%
13	Min92-134x8	850	5719.3	6011.2	5.10%
14	Perl83-12x2	140	204	204	0.00%
15	Perl83-55x15	120	1127.1	1117.5	-0.85%
16	Perl83-85x7	160	1655.2	1656.9	0.78%
17	Or76-117x14	150	12474.2	12909.66	3.49%

Πίνακας 3.3: Results of benchmark instances using best results achieved from both methods, compared to Best Known Solutions from bibliography.

Κεφάλαιο 4

Συμπεράσματα και επίλογος

4.1 Συμπεράσματα και επίλογος

Αυτή η πτυχιακή εργασία ερευνά τη χρήση γενετικών και μιμητικών αλγορίθμων για την επίλυση του προβλήματος χωροθέτησης εγκαταστάσεων και δρομολόγησης οχημάτων (LRP). Παρόλο που έχουν γίνει πολλές προσεγγίσεις επίλυσης του LRP με τη χρήση μεθευρετικών αλγορίθμων, δεν φαίνεται να υπάρχει αρκετή βιβλιογραφία για την επίλυση του με γενετικές στρατηγικές εξέλιξης και ειδικότερα με γενετικούς αλγορίθμους και μιμητικούς αλγορίθμους. Το ίδιο δεν ισχύει για τα υπό-προβλήματα που αποτελούν το LRP, όπου οι αλγόριθμοι αυτοί έχουν αποδειχθεί ότι μπορούν να πετύχουν πολύ ικανοποιητικά αποτελέσματα.

Εξετάζεται η επίλυση του προσαρμοσμένου προβλήματος LRP, όπως προτάθηκε από τους *J. Perl και M. S. Daskin*[2] με τη χρήση γενετικών αλγορίθμων και υβριδικών γενετικών αλγορίθμων (μιμητικών αλγορίθμων). Το πρόβλημα διαχωρίζεται σε δύο φάσεις και εξετάζεται η επίλυση των δύο φάσεων ταυτόχρονα καθώς και ξεχωριστά. Αφού γίνει σύγκριση μεταξύ των γενετικών και μιμητικών αλγορίθμων, μπορούμε να δούμε ότι οι μιμητικοί αλγόριθμοι είναι πιο αποδοτικοί ανεξάρτητα από την προσέγγιση που επιλέγεται για την επίλυση των επί μέρους φάσεων.

Με τη χρήση των μιμητικών αλγορίθμων πλέον λύνεται ένα σύνολο από βιβλιογραφικά προβλήματα αναφοράς, όπου μπορούμε να δούμε τη συμπεριφορά των δύο μεθόδων επίλυσης των φάσεων του προβλήματος, καθώς και τη γραφική απεικόνιση των καλύτερων λύσεων που έχουν επιτευχθεί με κάθε μέθοδο για το εκάστοτε πρόβλημα. Έτσι καταλήγουμε στο συμπέρασμα ότι η προσέγγιση για την επίλυση των φάσεων εξαρτάται σημαντικά από τη φύση του προβλήματος. Είδαμε ότι με την επίλυση των δύο υπό-προβλημάτων ξεχωριστά καταλήξαμε περισσότερες φορές σε χαμηλότερες λύσεις συγκριτικά με την επίλυση τους ταυτόχρονα. Ωστόσο, είδαμε επίσης ότι υπάρχουν προβλήματα όπου αυτή η προσέγγιση μπορεί να οδηγήσει σε υπό-βέλτιστες λύσεις με αδυναμία περαιτέρω βελτίωσης, ενώ η ταυτόχρονη επίλυση των δύο φάσεων σε αυτά τα προβλήματα οδηγούσε σε πολύ καλύτερες λύσεις.

Οι μελλοντικές κατευθύνσεις που θα μπορούσε να πάρει αυτή η έρευνα είναι η βελτιστοποίηση της προσαρμοστικότητας του προγράμματος με τη χρήση δυναμικής διαμόρφωσης των πιθανοτήτων μετάλλαξης, καθώς και ο συνδυασμός του με διαδικασίες

που συναντούμε σε άλλες μεθευρετικές και ευρετικές μεθόδους. Επίσης, η υλοποίηση του προγράμματος θα μπορούσε να γίνει σε μια αντικειμενοστραφή γλώσσα προγραμματισμού όπως η C++, όπου η απόδοση του από άποψη χρόνου θα γίνει καλύτερη, ενώ θα είναι σχετικά εύκολη η υλοποίησή του. Τέλος, αφού βελτιωνόταν στην επίλυση του προσαρμοσμένου LRP, θα μπορούσε να γίνει κατάλληλη υλοποίηση για να λύνει το πρόβλημα LRP στη γενική μορφή του.

Κεφάλαιο 5

Παράρτημα

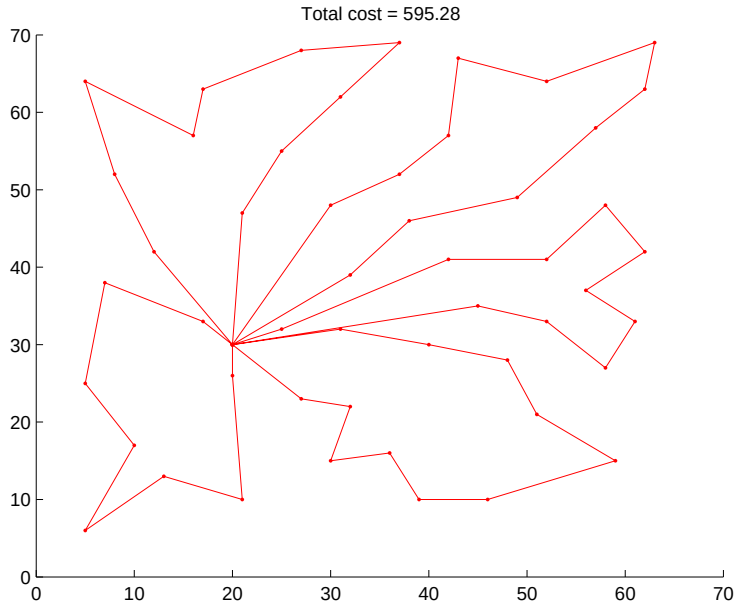
5.1 Γραφική απεικόνιση λύσεων

Στο εξής παράρτημα παρουσιάζονται γραφικά οι καλύτερες λύσεις που έχουν βρεθεί στα προβλήματα αναφοράς της βιβλιογραφίας (Πίνακας 3.1, Πίνακας 3.2). Για την γραφική τους αναπαράσταση, όπως και για την υλοποίηση του προγράμματος επίλυσης, χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον Matlab.

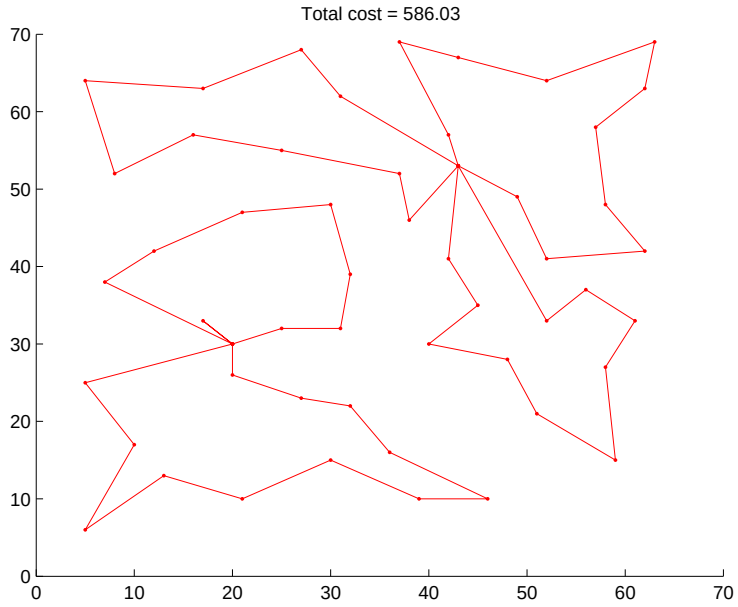
Για την ανάγκη της γραφικής απεικόνισης, οι κωδικοποιημένες λύσεις μετατρέπονται σε μια άλλη μορφή, παρόμοια με αυτή που περιγράφεται στο μαθηματικό μοντέλο των Perl και Daskin [2], η οποία χρησιμοποιείται συνήθως στην υλοποίηση του LRP. Ο εκάστοτε πελάτης συμβολίζεται με τον αύξοντα αριθμό του αυξημένο κατά το πλήθος των διαθέσιμων εγκαταστάσεων m. Έτσι ο νέος αριθμός που συμβολίζει κάθε πελάτη είναι $i + m$. Η κάθε λύση κωδικοποιείται σε ένα νέο διάνυσμα όπου κάθε στοιχείο του αποτελεί τον επόμενο κόμβο που θα πρέπει να επισκεφθεί το όχημα που εκτελεί τις διανομές, είτε αυτό είναι πελάτης είτε αποθήκη.

Ας δούμε για παράδειγμα πως μετασχηματίζεται η λύση 1:

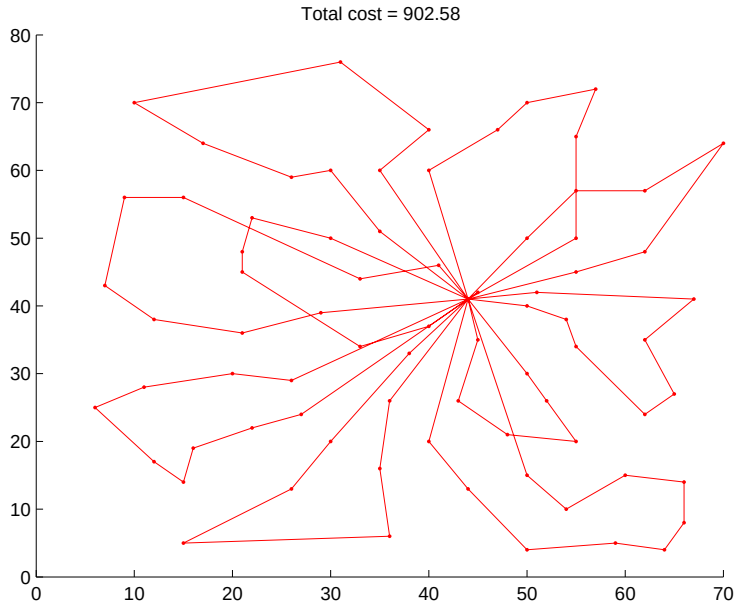
$$\left.\begin{matrix}[52341]\\ [0101]\\ [0104]\end{matrix}\right\} \Longrightarrow [296724854]$$



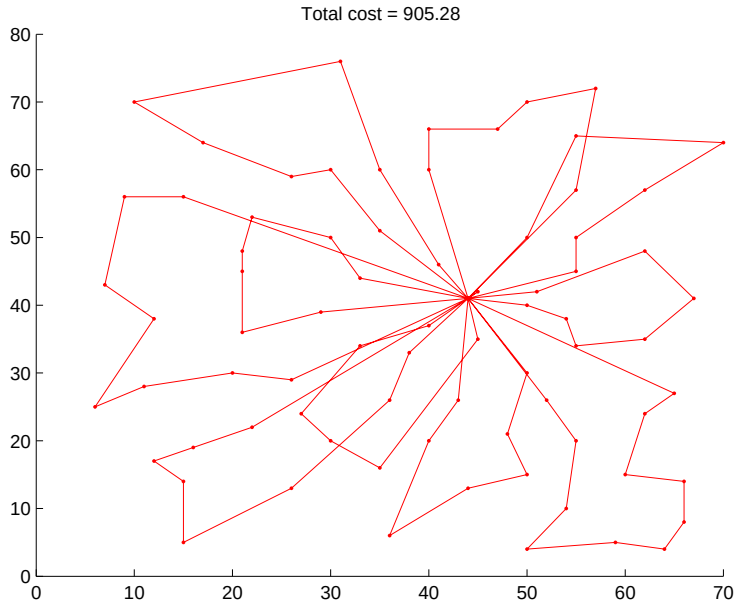
Σχήμα 5.1: Πρόβλημα 1, μέθοδος 1



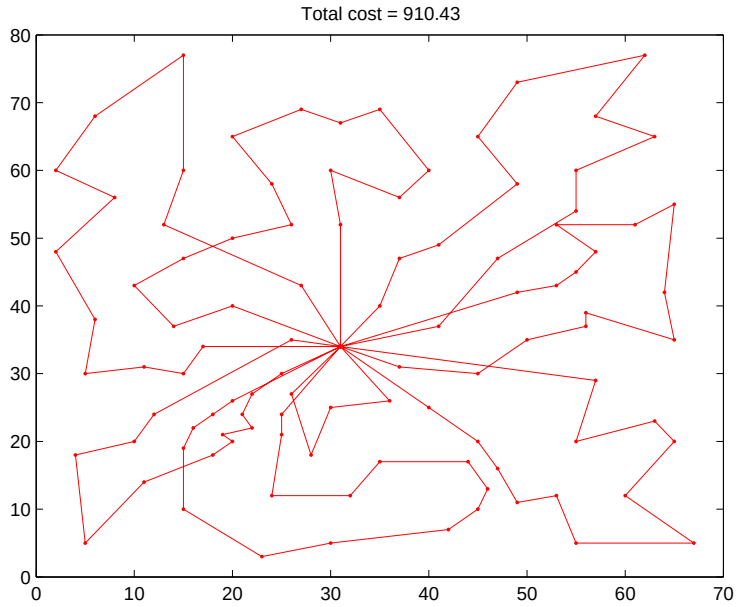
Σχήμα 5.2: Πρόβλημα 1, μέθοδος 2



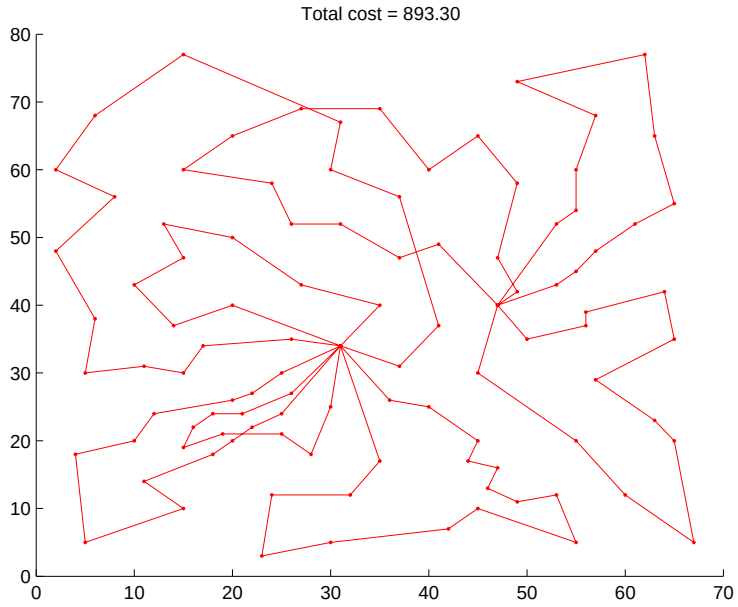
Σχήμα 5.3: Πρόβλημα 2, μέθοδος 1



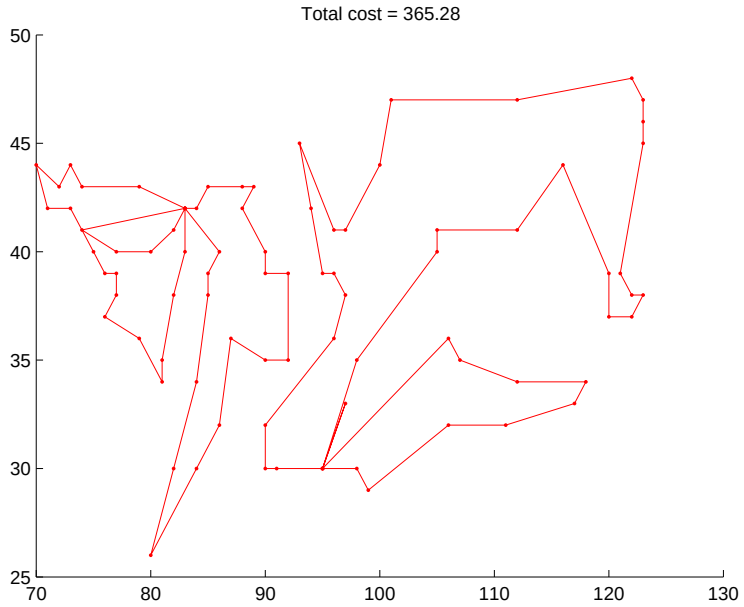
Σχήμα 5.4: Πρόβλημα 2, μέθοδος 2



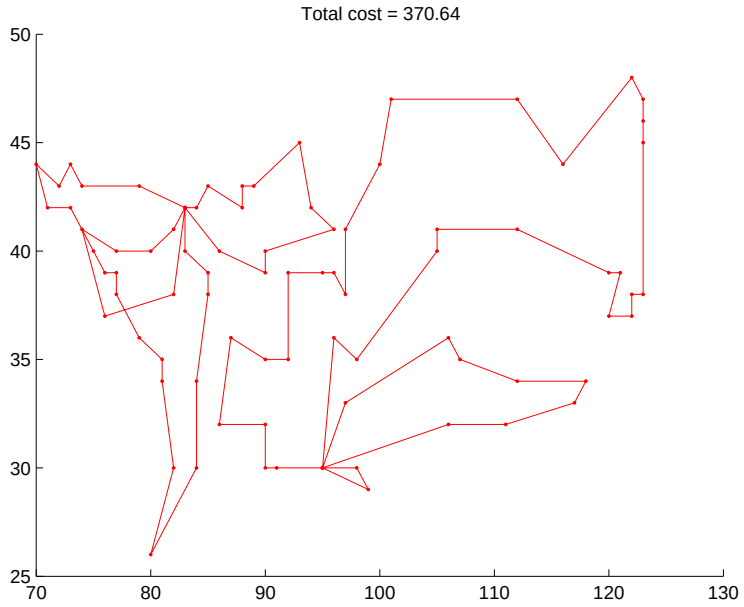
Σχήμα 5.5: Πρόβλημα 3, μέθοδος 1



Σχήμα 5.6: Πρόβλημα 3, μέθοδος 2

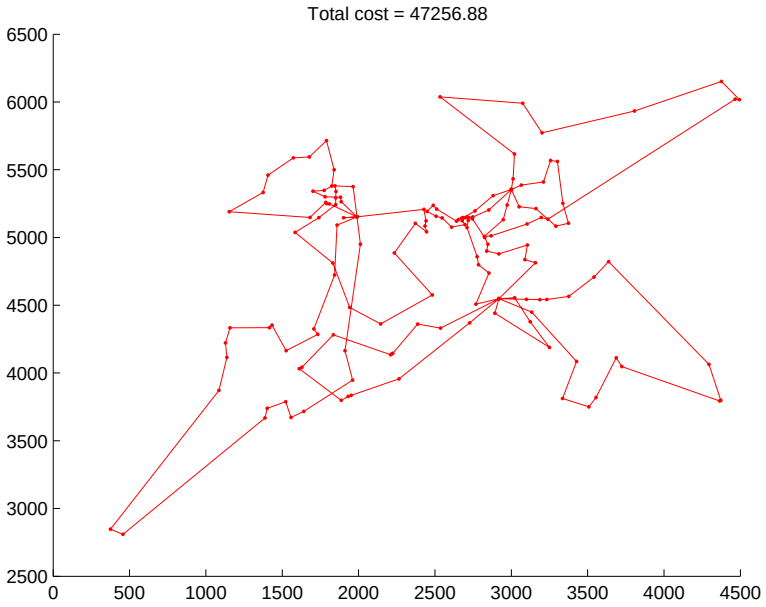


Σχήμα 5.7: Πρόβλημα 4, μέθοδος 1

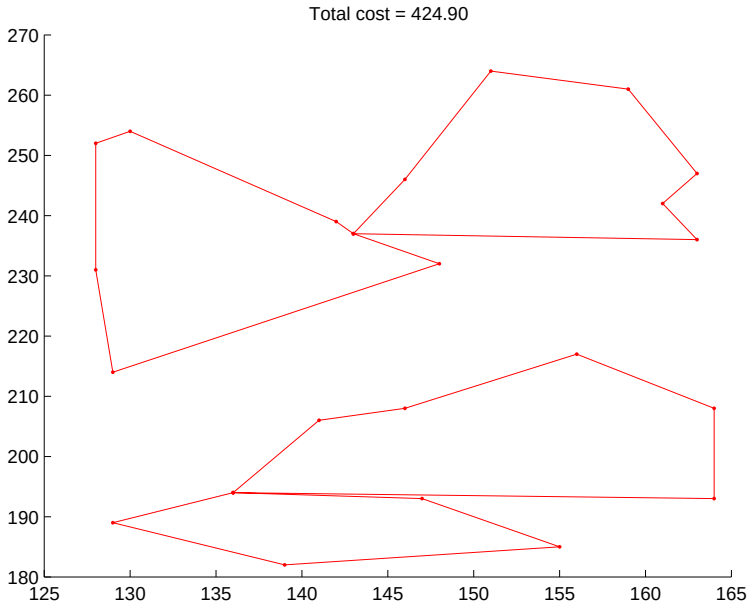


Σχήμα 5.8: Πρόβλημα 4, μέθοδος 2

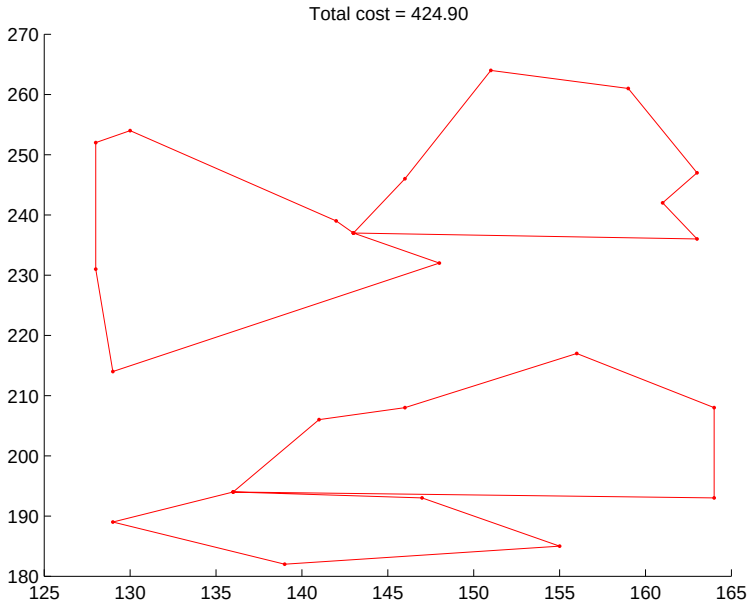
Σχήμα 5.9: Πρόβλημα 5, μέθοδος 1



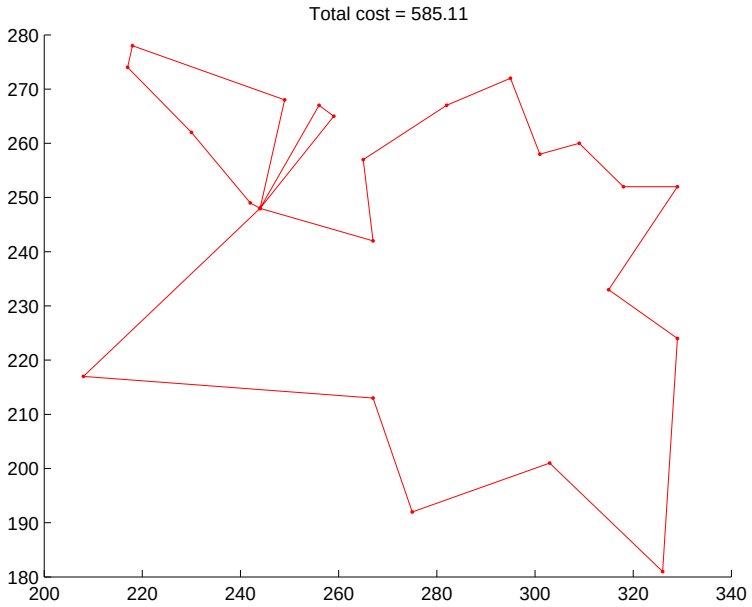
Σχήμα 5.10: Πρόβλημα 5, μέθοδος 2



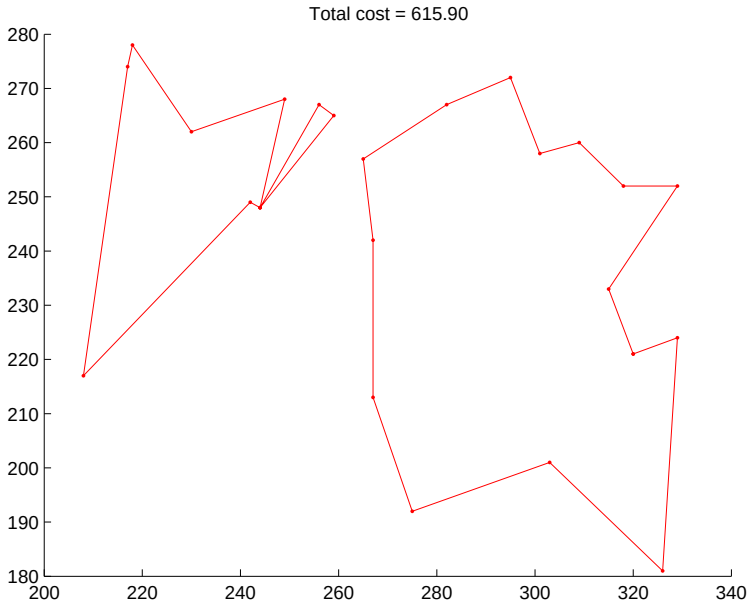
Σχήμα 5.11: Πρόβλημα 6, μέθοδος 1



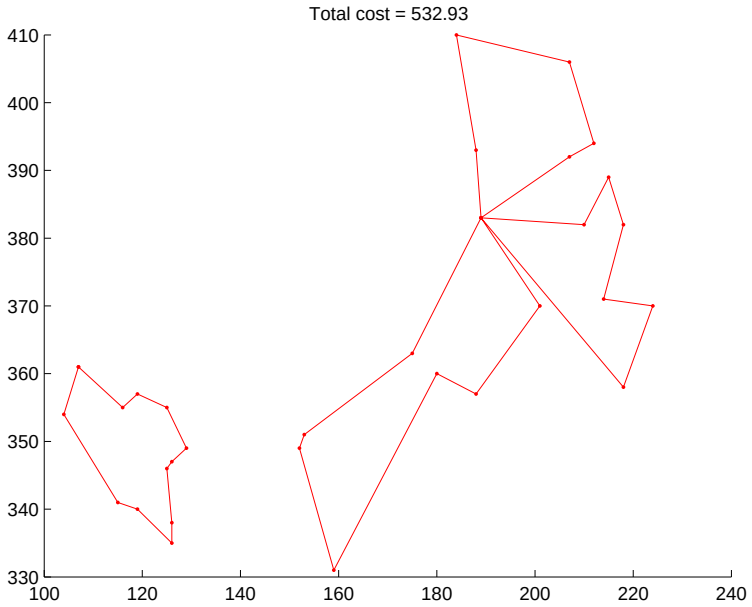
Σχήμα 5.12: Πρόβλημα 6, μέθοδος 2



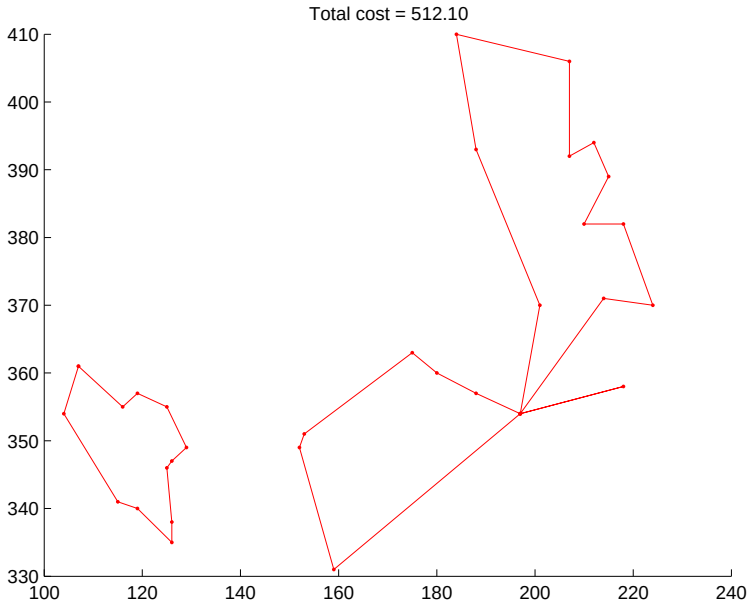
Σχήμα 5.13: Πρόβλημα 7, μέθοδος 1



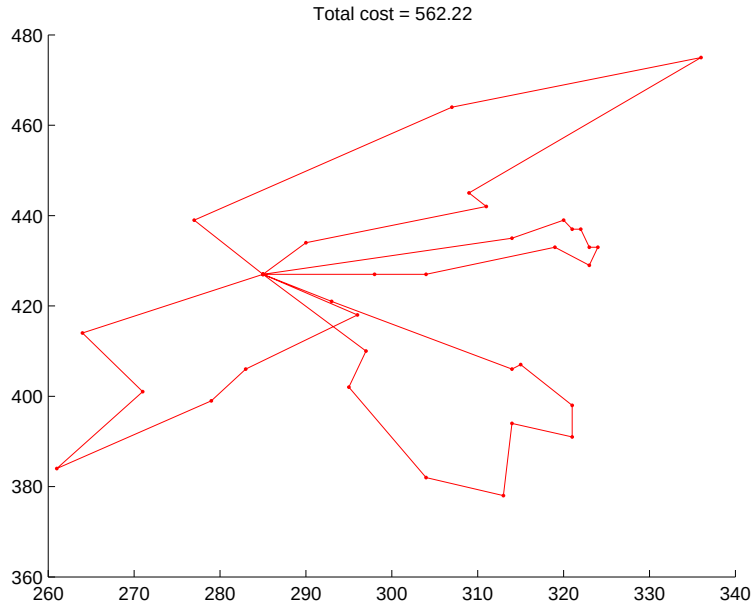
Σχήμα 5.14: Πρόβλημα 7, μέθοδος 2



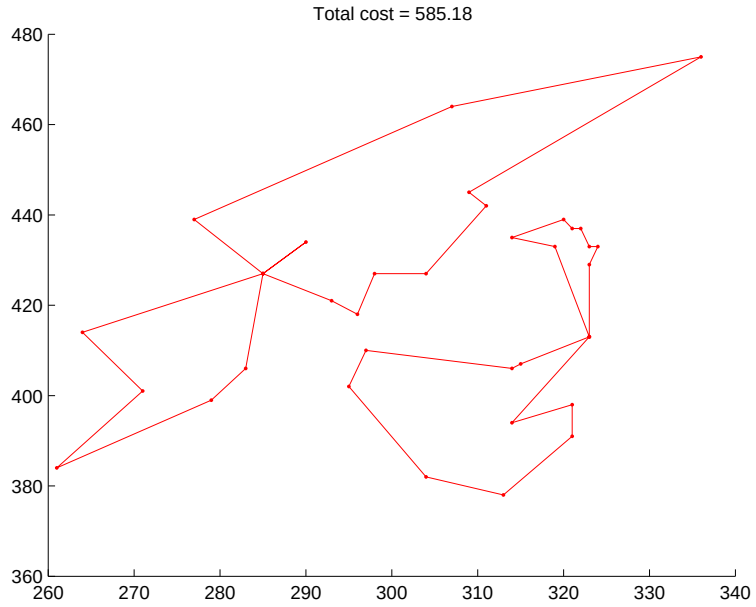
Σχήμα 5.15: Πρόβλημα 8, μέθοδος 1



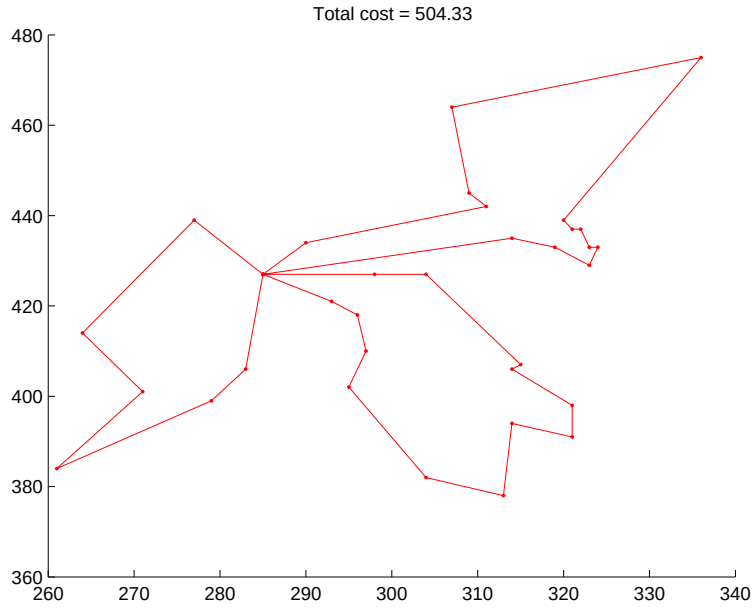
Σχήμα 5.16: Πρόβλημα 8, μέθοδος 2



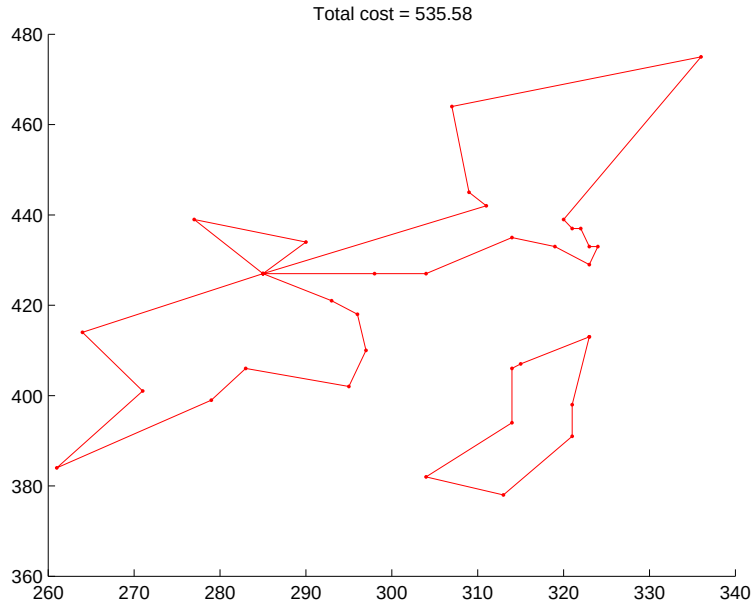
Σχήμα 5.17: Πρόβλημα 9, μέθοδος 1



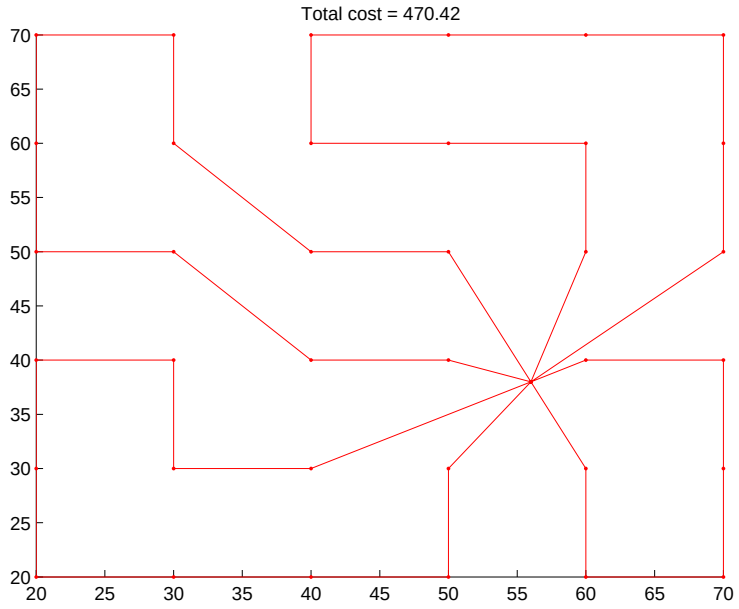
Σχήμα 5.18: Πρόβλημα 9, μέθοδος 2



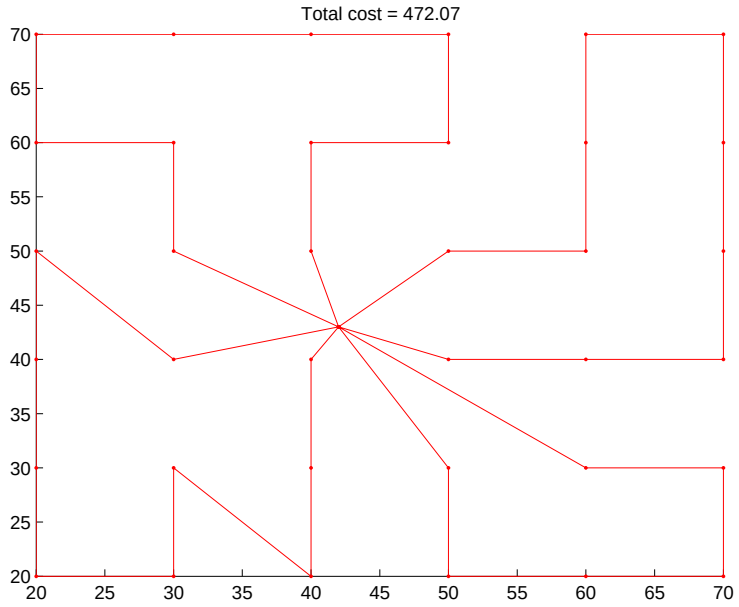
Σχήμα 5.19: Πρόβλημα 10, μέθοδος 1



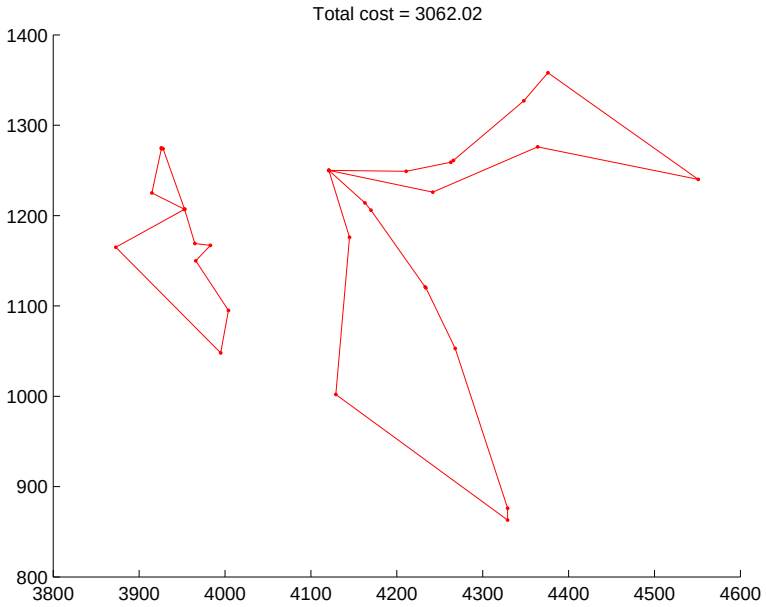
Σχήμα 5.20: Πρόβλημα 10, μέθοδος 2



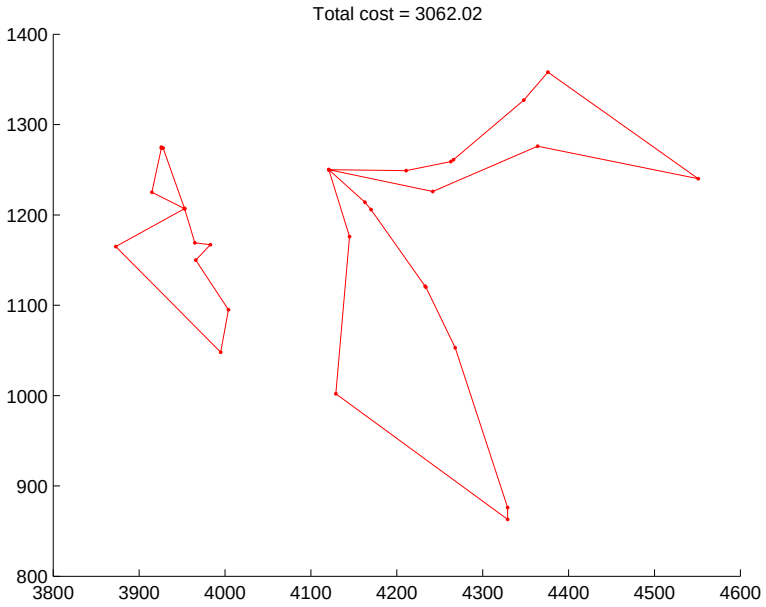
Σχήμα 5.21: Πρόβλημα 11, μέθοδος 1



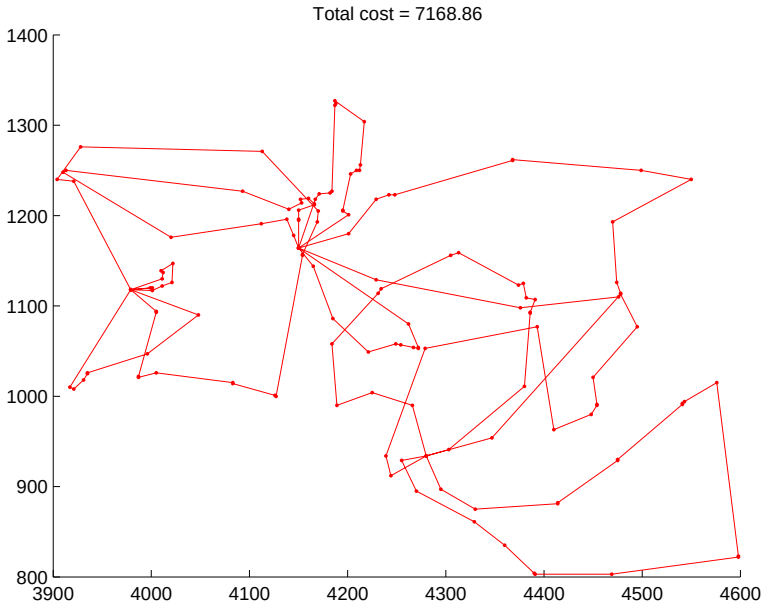
Σχήμα 5.22: Πρόβλημα 11, μέθοδος 2



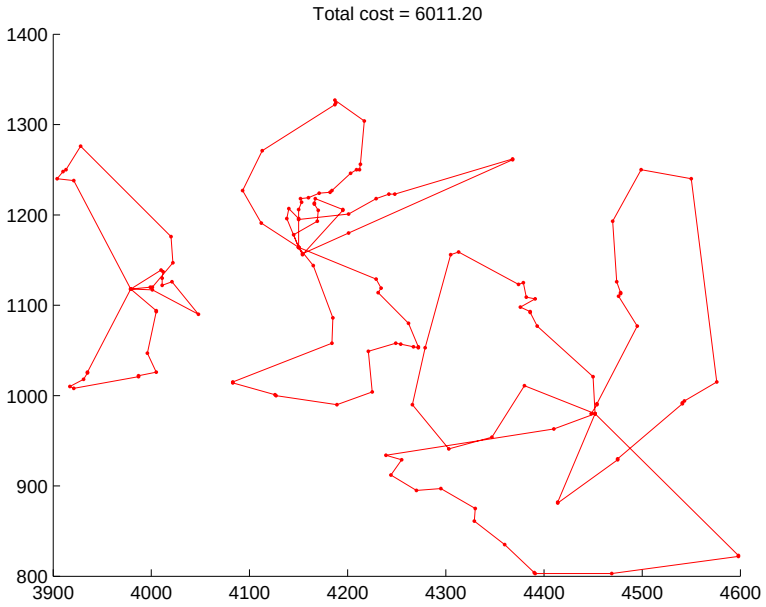
Σχήμα 5.23: Πρόβλημα 12, μέθοδος 1



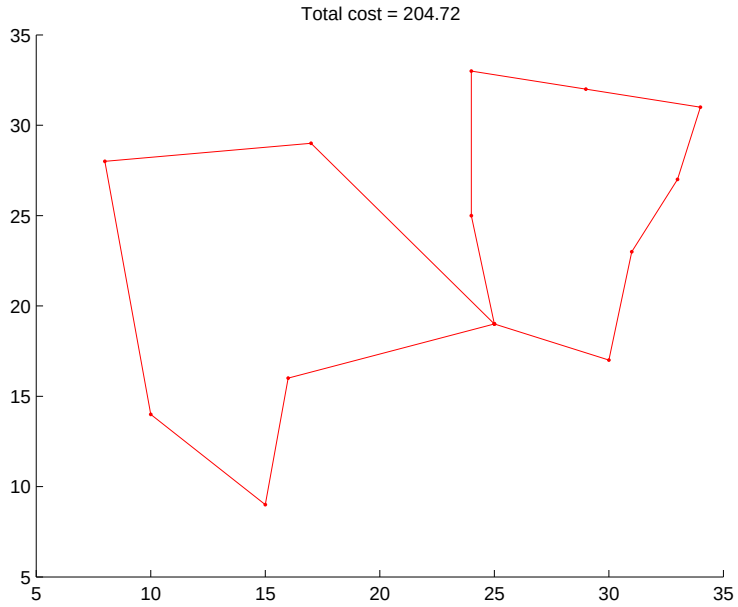
Σχήμα 5.24: Πρόβλημα 12, μέθοδος 2



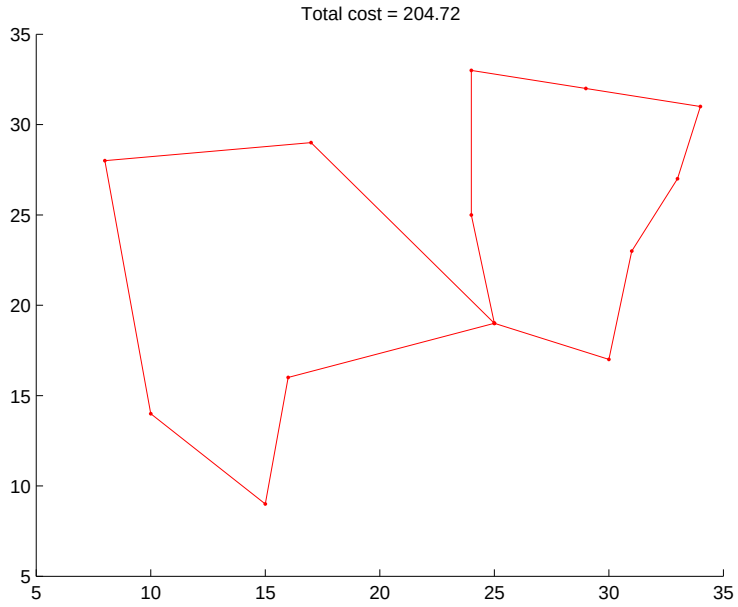
Σχήμα 5.25: Πρόβλημα 13, μέθοδος 1



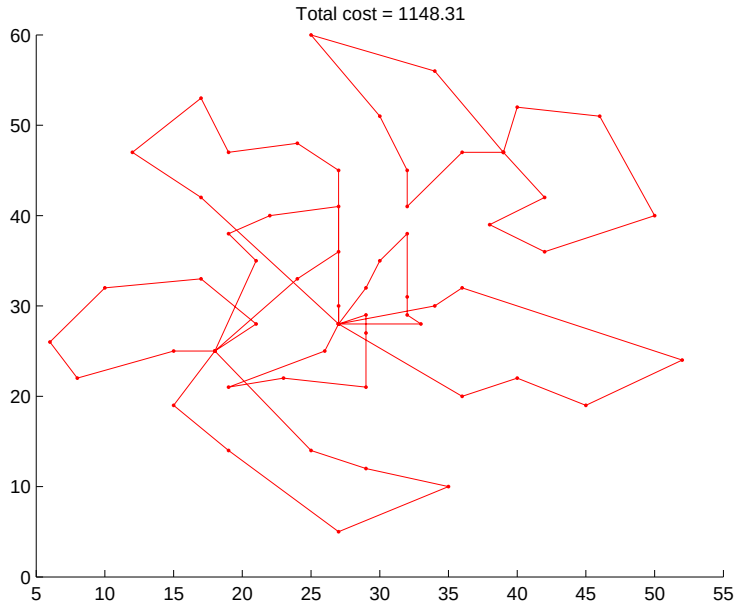
Σχήμα 5.26: Πρόβλημα 13, μέθοδος 2



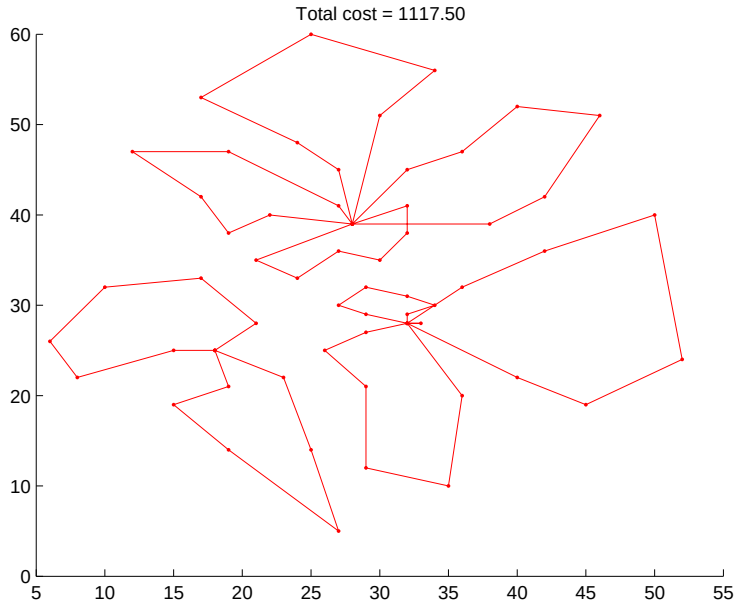
Σχήμα 5.27: Πρόβλημα 14, μέθοδος 1



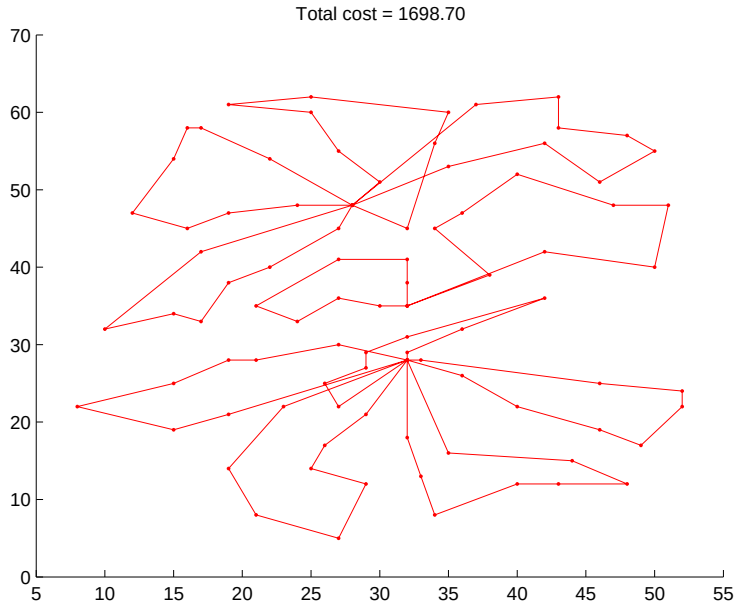
Σχήμα 5.28: Πρόβλημα 14, μέθοδος 2



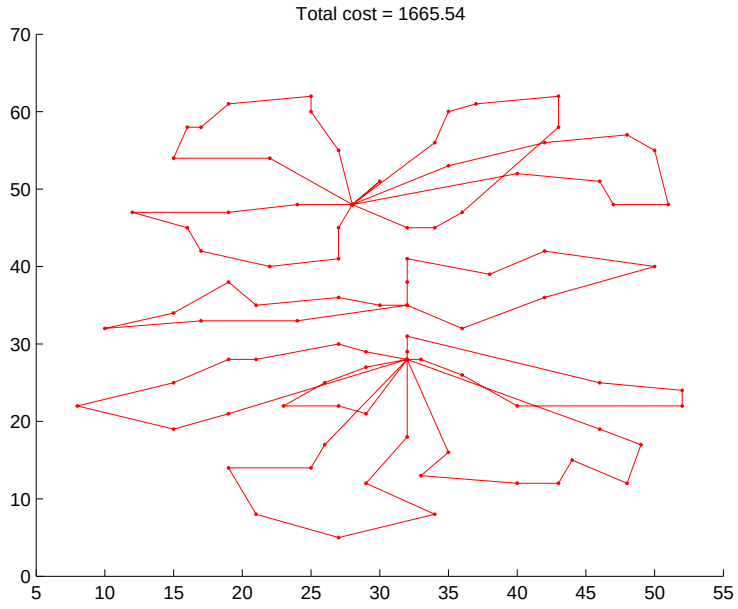
Σχήμα 5.29: Πρόβλημα 15, μέθοδος 1



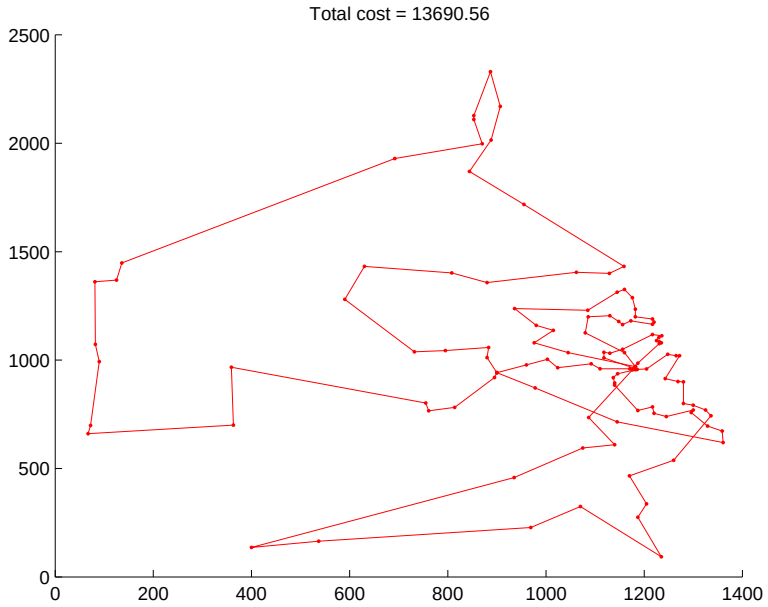
Σχήμα 5.30: Πρόβλημα 15, μέθοδος 2



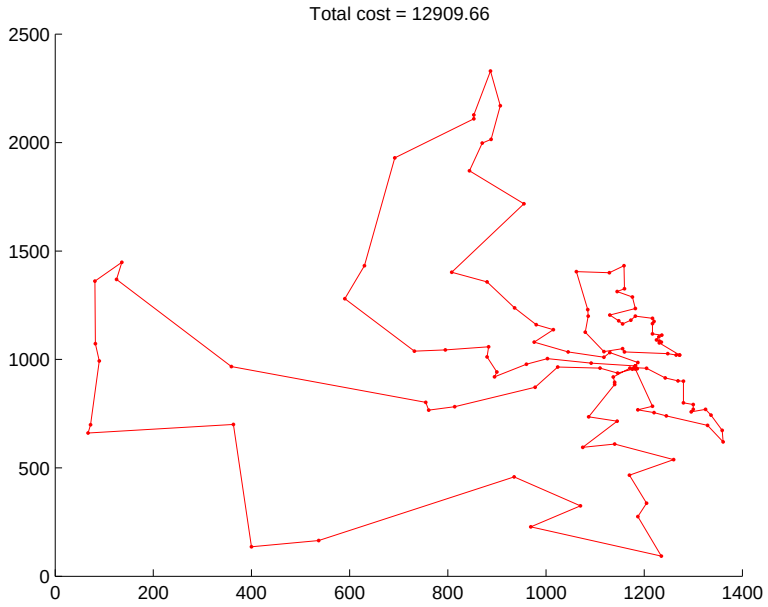
Σχήμα 5.31: Πρόβλημα 16, μέθοδος 1



Σχήμα 5.32: Πρόβλημα 16, μέθοδος 2



Σχήμα 5.33: Πρόβλημα 17, μέθοδος 1



Σχήμα 5.34: Πρόβλημα 17, μέθοδος 2

Βιβλιογραφία

[1] Ι. Μαρινάκης και Α. Μυγδαλάς, *Σχεδιασμός και βελτιστοποίηση της εφοδιαστικής αλυσίδας*. Εκδόσεις "σοφία" α.ε., Θεσσαλονίκη, 2008.

[2] J. Perl and M.S. Daskin, *A Warehouse Location-Routing Problem*. Transpn. Res. -B Vol. 19B, No. 5, p. 381-396, 1985.

[3] G. Laporte, Y. Nobert and S. Taillefer, *Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems*. Transportation Science Vol. 22, No. 3, August 1988.

[4] H. Yildiz, M. P. Johnson and S. Roehrig, *A Genetic Algorithm for the Home-Delivered Meals Location-Routing Problem*

[5] S. K. Jacobsen and O. B. G. Madsen, *A comparative study of heuristics for a two-level routing-location problem*. European Journal of Operational Research 5 (1980), p. 378-387, North-Holland Publishing Company

[6] Y. Marinakis and M. Marinaki, *A Bilevel Genetic Algorithm for a real life location routing problem*. International Journal of Logistics Research and Applications, 11:1, p. 49-65, 2007.

[7] S. C. Liu and S. B. Lee, *A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration*. Springer-Verlag London Limited, 2003.

[8] D. Tuzun and L. I. Burke, *A two-phase tabu search approach to the location routing problem*. European Journal of Operational Reasearch 116, p. 87-99, 1999.

[9] R. Caballero, M. González, F. M. Guerrero, J. Molina and C. Paralera, *Solving a multiobjective location routing problem with a metaheuristic based on a tabu search. Application to a real case in Andalusia*. European Journal of Operational Reasearch 177, p. 1751-1763, 2007.

[10] R. Srivastava and W. C. Benton, *The Location-Routing Problem: Considerations In Physical Distribution System Design*. Computers Opns Res. Vol. 17, No. 5, p. 427-435, 1990.

[11] C. Prins, C. Prodon and R. W. Calvo, *A Memetic Algorithm with Population Management (MA|PM) for the Capacitated Location-Routing Problem*. J. Gottlieb and G.R. Raidl (Eds.): EvoCOP 2006, LNCS 3906, pp. 183–194, Springer-Verlag Berlin Heidelberg, 2006.

[12] Μ. Παπαγεωργίου, *Δυναμικός Προγραμματισμός*. Πολυτεχνείο Κρήτης, Τμήμα Μηχανικών Παραγωγής και Διοίκησης, Χανιά, 2008.

[13] Ι. Μαρινάκης, Α. Μυγδαλάς, *Σημειώσεις Συνδυαστικής Βελτιστοποίησης*, Χανιά, 19 Δεκεμβρίου 2009.

[14] S. Barreto, *Location Routing Problem Instances*, http://sweet.ua.pt/iscf143/private/SergioBarretoHomePage.htm , 2007.

[15] Y. Marinakis, M. Marinaki and N. Matsatsinis, *Honey Bees Mating Optimization for the Location Routing Problem*.

[16] Duhamel C. et al, *A GRASP×ELS approach for the capacitated location-routing problem*. Computers & Operations Research (2009), doi: 10.1016/j.cor.2009.07.004