Εθνικό Μετσόβιο Πολυτεχνείο

Τμήμα Μηχανολόγων Μηχανικών

Τομέας Βιομηχανικής Διοίκησης και Επιχειρησιακής Έρευνας

Σχεδιασμός και υλοποίηση υβριδικού μεταευρετικού αλγορίθμου βελτιστοποίησης για το πράσινο πρόβλημα δρομολόγησης στόλου ετερογενών οχημάτων σε αστικό περιβάλλον

Διπλωματική Εργασία

του

ΕΥΑΓΓΕΛΟΥ-ΙΑΣΟΝΑ Γ. ΜΟΣΧΟΥ

`Επιβλέπων: Πόνης Σταύρος
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον υπεύθυνο καθηγητή μου κ. Σταύρο Πόνη τόσο για την άριστη συνεργασία μας, όσο και για τις πολύτιμες συμβουλές και τη διαρκή του καθοδήγηση όλους αυτούς τους μήνες. Ταυτόχρονα, θα ήθελα να ευχαριστήσω την δρ. Έλενα Ρόκου για την αμέριστη βοήθεια της, την υπομονή της και την διαρκώς χαρούμενη διάθεση της, χωρίς την οποία δεν θα ήταν δυνατή η περαίωση της παρούσας διπλωματικής. Τέλος, θέλω να ευχαριστήσω τους γονείς, τα αδέλφια και τη σύντροφο μου, για τη στήριξη και τη ψυχική δύναμη που μου παρείχαν όλους αυτούς τους μήνες, αλλά και καθ'όλη τη διάρκεια των σπουδών μου.

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Αλγορίθμων

Κατάλογος Εξισώσεων

Κατάλογος Πινάκων

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την υλοποίηση ενός νέου υβριδικού αλγορίθμου για την επίλυση ενός «πράσινου» προβλήματος δρομολόγησης στόλου ετερογενών οχημάτων σε αστικό περιβάλλον. Ο αλγόριθμος που παρουσιάζεται κάνει χρήση δύο εμφωλευμένων γενετικών αλγορίθμων(Genetic Algorithm-GA), ενός εξωτερικού και ενός εσωτερικού αλγορίθμου. Οι δύο αλγόριθμοι αλληλεπιδρούν μεταξύ τους για την εύρεση του τελικού αποτελέσματος.

Το πρόβλημα αποτελείται από ένα σύνολο πελατών προς εξυπηρέτηση, μια κεντρική αποθήκη και ένα σύνολο ετερογενών μεταξύ τους οχημάτων με συγκεκριμένη χωρητικότητα και χαρακτηριστικά το καθένα. Κάθε πελάτης έχει συγκεκριμένη ζήτηση και συντεταγμένες, βρίσκεται σε δεδομένη ζώνη κίνησης σε αστικό περιβάλλον και πρέπει να εξυπηρετηθεί από ένα και μόνο όχημα. Κάθε όχημα ξεκινά το δρομολόγιο του από την αποθήκη, εξυπηρετεί όλους τους πελάτες που του έχουν ανατεθεί με συγκεκριμένη σειρά και επιστρέφει πάλι στη κεντρική αποθήκη πριν τη λήξη της βάρδιας. Στόχος του προβλήματος είναι η ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και η ελαχιστοποίηση της διανυόμενης απόστασης, ενώ ο συνολικός όγκος εκπομπών όλων των οχημάτων πρέπει να βρίσκεται κάτω από ένα συγκεκριμένο όριο. Το τελευταίο ακριβώς χαρακτηριστικό το εντάσσει στην κατηγορία των «πράσινων» προβλημάτων ή αλλιώς προβλημάτων που λαμβάνουν υπόψην τους παράγοντες που επιβαρύνουν το περιβάλλον.

Η μέθοδος που προτείνεται για την επίλυση περιλαμβάνει έναν εξωτερικό γενετικό αλγόριθμο που αντιστοιχίζει τους πελάτες σε σύνολα, τα οποία εξυπηρετούνται από συγκεκριμένη κατηγορία οχημάτων το καθένα, και έναν εσωτερικό γενετικό αλγόριθμο, ο οποίος επιλύει το πρόβλημα δρομολόγησης οχημάτων για κάθε κατηγορία ξεχωριστά. Συγκεκριμένα, κάθε λύση στον εσωτερικό γενετικό κωδικοποιείται ως ένα χρωμόσωμα. Ο αλγόριθμος, σε κάθε επανάληψη, με χρήση κατάλληλων τεχνικών, προσπαθεί να δημιουργήσει καλύτερες λύσεις συνδυάζοντας μεταξύ τους τα χρωμοσώματα. Ο εσωτερικός γενετικός αλγόριθμος χρησιμοποιείται τόσες φορές, όσες και οι κατηγορίες των οχημάτων που περιλαμβάνονται στο πρόβλημα. Όταν φτάσει στο τέλος των επαναλήψεων που έχουν ορισθεί, η τελική λύση τροφοδοτείται στον εξωτερικό γενετικό αλγόριθμο. Ο εξωτερικός γενετικός, ελέγχει τη συνολικά διανυόμενη απόσταση για όλες τις κατηγορίες οχημάτων μαζί και προσπαθεί να την ελαχιστοποιήσει συνδυάζοντας τους πελάτες σε διαφορετικά σύνολα. Με άλλα λόγια, ο εξωτερικός γενετικός αλγόριθμος, προσπαθεί να βρει εκείνο το συνδυασμό πελατών που πρέπει να εξυπηρετηθεί από κάθε κατηγορία οχημάτων ώστε η συνολικά διανυόμενη απόσταση να είναι η ελάχιστη, οι συνολικές εκπομπές να είναι κάτω από το δεδομένο όριο και το πλήθος των απαιτούμενων οχημάτων να είναι μικρότερο από το πλήθος των διαθέσιμων οχημάτων, ενώ ο εσωτερικός γενετικός αλγόριθμος, προσπαθεί για συγκεκριμένο σύνολο πελατών να ελαχιστοποιήσει τη διανυόμενη απόσταση και τα απαιτούμενα οχήματα της εκάστοτε κατηγορίας.

Ο αλγόριθμος αυτός εξετάσθηκε ως προς την αποδοτικότητα και την αποτελεσματικότητα, τόσο σε πρότυπα προβλήματα από τη βιβλιογραφία, όσο και σε ένα ενδεικτικό πρόβλημα που κατασκευάσθηκε με βάση πραγματικά στοιχεία.

Abstract

The current thesis focuses on developing a new hybrid algorithm for solving the green capacitated vehicle routing problem, with a heterogeneous fleet of vehicles in an urban environment. The proposed algorithm is a combination of two genetic algorithms (GA), one internal and one external, which interact with each other.

The problem consists of a set of customers, which need to be serviced, a central depot and a set of heterogeneous vehicles, each one with different capacities and characteristics. Each customer has a predetermined demand, coordinates, and belongs in a traffic zone in an urban environment and has to be serviced by exactly one vehicle. Every vehicle begins its tour from the central depot, services all the customers assigned to it, in a particular order and returns to the depot before the end of duty. The objective of the problem is the minimization of the used vehicles, as well as the minimization of the total travelled distance, while the total volume of emissions is below the specified limit.

The proposed method for solving the problem consists of an exterior genetic algorithm that assigns customers to different sets, which are to be serviced by a specific category of vehicles, and an interior genetic algorithm that solves the vehicle routing problem for each category independently. Specifically, every solution in the internal genetic algorithm is represented as a chromosome. At each iteration, the algorithm tries to create better solutions by combining chromosomes, using specific methods. The internal genetic algorithm is used as many times, as the number of categories of vehicles in the problem. The final solution after the end of iterations is being fed to the external genetic algorithm. The external genetic, on the other hand, calculates the total travelled distance for every category of vehicles combined, and tries to minimize it by assigning different customers to different sets of vehicles. In other words, the external genetic algorithm is trying to find the combination of customers to be serviced by each vehicle category, so that the total travelled distance is minimum, the total volume of emissions is below the specified limit and the number of used vehicles is below the number of available vehicles for each category, while the internal genetic algorithm is trying to minimize the travelled distance and the required amount of vehicles for a given set of customers.

The efficiency and effectiveness of the proposed algorithm has been thoroughly tested using both test cases from the literature and an illustrative case study.

1 Εισαγωγή

1.1 Η σημασία της εφοδιαστικής αλυσίδας

Λόγω της τάσης διεθνοποίησης τις τελευταίες δεκαετίες, η σημασία της εφοδιαστικής(logistics) γίνεται ολοένα και μεγαλύτερη. Το σημείο-κλειδί μιας εφοδιαστικής αλυσίδας αποτελούν οι μεταφορές, οι οποίες συμπληρώνουν τις επιμέρους διαδικασίες και αναλογούν στο ένα τρίτο του συνολικού κόστους logistics για τα περισσότερα προϊόντα. Στόχος κάθε εφοδιαστικής αλυσίδας είναι η μεγιστοποίηση της συνολικής της αξίας, η οποία ορίζεται ως η διαφορά της αξίας του τελικού προϊόντος για τον πελάτη από την προσπάθεια που καταβάλει η εφοδιαστική αλυσίδα για την εξυπηρέτηση του πελάτη. Σημαντική συνιστώσα της προσπάθειας αυτής αποτελεί η διαδικασία διανομής προϊόντων(Chopra and Meindl 2007).

Τις τελευταίες δεκαετίες, πληθώρα ερευνητών έχουν επικεντρωθεί στο πρόβλημα δρομολόγησης οχημάτων λόγω του αυξημένου ενδιαφέροντος σε γεωγραφικές λύσεις και τεχνολογίες, αλλά και λόγω της χρήσης του στο κομμάτι των logistics και των μεταφορών, για τη διανομή προϊόντων. Έτσι, ολοένα και περισσότερες εταιρείες logistics προσπαθούν να οργανώσουν τις παραδόσεις τους χρησιμοποιώντας νέες σύγχρονες τεχνολογίες. Το σημαντικότερο τμήμα της μείωσης του κόστους των μεταφορών σχετίζεται με την καλύτερη οργάνωση των διαδρομών, δηλαδή την επίλυση του προβλήματος δρομολόγησης οχημάτων(VRP). Έτσι, με τον προσεκτικό σχεδιασμό των διαδρομών στις περιοχές παράδοσης, μπορούν να μειωθούν το κόστος, ο χρόνος, και τα καύσιμα που χρησιμοποιούνται, αλλά και να δοθεί μια περιβαλλοντική διάσταση στο πρόβλημα.

Η παρούσα διπλωματική ασχολείται με τη μοντελοποίηση και επίλυση ενός τέτοιου προβλήματος δρομολόγησης οχημάτων, το οποίο να προσεγγίζει ικανοποιητικά τις συνθήκες που επικρατούν στην πράξη, ώστε να δίνεται η δυνατότητα αξιοποίησης του για την επίλυση πραγματικών προβλημάτων δρομολόγησης. Έτσι, εκτός από τις κλασικές παραδοχές που υιοθετούνται σε προβλήματα αυτής κατηγορίας, όπως αυτά συναντώνται στη βιβλιογραφία, χρησιμοποιείται επιπλέον η έννοια του αστικού περιβάλλοντος η οποία περιλαμβάνει πολλούς πελάτες σε κοντινές αποστάσεις μεταξύ τους, ώρες αιχμής και ζώνες κίνησης. Ταυτόχρονα, γίνεται προσπάθεια απόδοσης περιβαλλοντικής διάστασης στο πρόβλημα, υιοθετώντας ένα συγκεκριμένο ανώτατο όριο εκπομπών διοξειδίου του άνθρακα.

Ο τρόπος επίλυσης χρησιμοποιεί σύγχρονες τεχνικές, με τη χρήση δύο εμφωλευμένων γενετικών αλγορίθμων καθώς και μια μέθοδο τοπικής αναζήτησης για βελτίωση της παραγόμενης λύσης.

1.2 Δομή της διπλωματικής εργασίας

Η παρούσα διπλωματική επικεντρώνεται στη μελέτη, σχεδιασμό και υλοποίηση ενός αλγορίθμου επίλυσης για μια νέα εκδοχή του προβλήματος δρομολόγησης οχημάτων.

Κεφάλαιο 1

Περιγράφεται η σημασία των μεταφορών στην εφοδιαστική αλυσίδα.

Κεφάλαιο 2

Παρουσιάζεται μια σύντομη βιβλιογραφική επισκόπηση του προβλήματος του περιοδεύοντος πωλητή, πάνω στο οποίο στηρίζεται το πρόβλημα δρομολόγησης οχημάτων, όσον αφορά την ιστορία, το μαθηματικό υπόβαθρο αλλά και τις εφαρμογές στις οποίες συναντάται.

Κεφάλαιο 3

Στο κεφάλαιο αυτό παρουσιάζεται το πρόβλημα δρομολόγησης οχημάτων, οι παραλλαγές του, οι εφαρμογές καθώς και οι τρόποι επίλυσης του.

Κεφάλαιο 4

Παρουσιάζεται εκτενώς η δομή του γενετικού αλγορίθμου και οι τρόποι προσαρμογής του στο πρόβλημα δρομολόγησης οχημάτων.

Κεφάλαιο 5

Στο κεφάλαιο αυτό πραγματοποιείται η αναλυτική περιγράφη του προβλήματος που πρόκειται να επιλυθεί.

Κεφάλαιο 6

Παρουσιάζεται η μαθηματική μοντελοποίηση του εξεταζόμενου προβλήματος, και επεξηγούνται αναλυτικά οι περιορισμοί που έχουν τεθεί.

Κεφάλαιο 7

Στο κεφάλαιο αυτό, αναλύεται η προτεινόμενη μέθοδος επίλυσης του αναφερθέντος προβλήματος σε μορφή διαγραμμάτων ροής και αναλύονται τα δομικά στοιχεία του αλγορίθμου.

Κεφάλαιο 8

Παρουσιάζεται ο αλγόριθμος που αναπτύχθηκε και δίνονται επεξηγήσεις στις διάφορες μεθόδους.

Κεφάλαιο 9

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα χρήσης του αλγορίθμου σε προβλήματα της βιβλιογραφίας, η παραμετροποίηση του, και στατιστική ανάλυση των αποτελεσμάτων.

Κεφάλαιο 10

Παρουσιάζονται τα δεδομένα από πραγματική εταιρεία, τα δεδομένα από σχετική έρευνα που πραγματοποιήθηκε, η εισαγωγή τους στον αλγόριθμο και η γραφική απεικόνιση της παραγόμενης λύσης.

Κεφάλαιο 11

Στο κεφάλαιο αυτό παρουσιάζεται μια σύνοψη της παρούσας διπλωματικής με συμπεράσματα και κατευθύνσεις για έρευνα στο μέλλον.

2 Το πρόβλημα του περιοδεύοντος πωλητή(TSP)

Το VRP βασίζεται σε ένα άλλο πρόβλημα βελτιστοποίησης, αυτό του Περιοδεύοντος Πωλητή. Το πρόβλημα του περιοδεύοντος πωλητή (Travelling Salesman Problem) είναι ένα από τα πιο μελετημένα συνδυαστικά προβλήματα βελτιστοποίησης. Το πρόβλημα μπορεί να περιγραφεί ώς εξής:

Με δεδομένα ένα γράφο G=(V,A), όπου V είναι ένα σύνολο από κόμβους και A ένα σύνολο από ακμές, C=(c_{ij}) είναι ένας πίνακας αποστάσεων (ή κόστους) που συνδέεται με το A. Το TSP προσπαθεί να βρει την ελάχιστη απόσταση της διαδρομής που ξεκινά από μια αφετηρία, περνάει από κάθε κόμβο, μια και μόνο φορά, και καταλήγει πάλι στην αφετηρία. Μια τέτοια διαδρομή ονομάζεται Χαμιλτονιανή(Hamiltonian)(Laporte 1992).

2.1 Σύντομη ιστορική αναδρομή του TSP

Η πιο συνήθης ερμηνεία του TSP αφορά έναν πωλητή που επισκέπτεται μα σειρά από **n** πελάτες ή πόλεις, αναζητώντας τη συντομότερη απόσταση. Το βασικό αυτό πρόβλημα χρησιμοποιείται σε πληθώρα εφαρμογών σχετικών με τη δρομολόγηση οχημάτων, αλλά εμφανίζει μια σειρά από περιορισμούς στη δρομολόγηση τους. Παρ'όλα αυτά, αρκετά προβλήματα βελτιστοποίησης, τα οποία δεν συνδέονται άμεσα με τη δρομολόγηση, μπορούν να αναχθούν στο TSP(Laporte 1992).

Μαθηματικά προβλήματα σχετιζόμενο με το TSP μελετήθηκαν γύρω στο 1800, από τον Ιρλανδό μαθηματικό Sir William Rowan Hamilton και από το Βρετανό μαθηματικό Thomas Penyngton Kirkman. Το TSP μελετήθηκε πρώτη φορά το 1930 από το μαθηματικό και οικονομολόγο Karl Menger στη Βιέννη και το Harvard. Αργότερο, το πρόβλημα, διερευνήθηκε από τους Hassler Whitney και Merrill Flood στο Princeton. Το 1940, το TSP μελετήθηκε από τους στατιστικολόγους Mahalanobis, Jessen, Gosh, Marks σε σχέση με την αγροτική εφαρμογή του και διαδόθηκε από το μαθηματικό Merrill Flood στους συναδέλφους του, στην Εταιρεία RAND. Μέθοδοι επίλυσης του TSP άρχισαν να εμφανίζονται σε δημοσιεύσεις στα μέσα του 1950, οι οποίες περιείχαν μικρές παραλλαγές του όρου TSP. Αν και το πρόβλημα είναι εύκολα αντιληπτό, είναι δύσκολο να λυθεί, και όπως απέδειξε ο Richard M.Karp το 1972, το Χαμιλτονιανό Κυκλικό Πρόβλημα είναι NP-complete (μη ντετερμινιστικό πολυωνυμικού χρόνου), το οποίο υποδεικνύει ότι και το TSP είναι NP-hard. Τα NP-hard προβλήματα είναι μια ιδιαίτερη κατηγορία προβλημάτων, τα πιο δύσκολα NP, το οποίο σημαίνει ότι δεν υπάρχει αλγόριθμος πολυωνομικού χρόνου για να τα επιλύσει, γεγονός που αποδυκνύει την προφανή υπολογιστική δυσκολία εύρεσης βέλτιστων διαδρομών. (Maredia 2010).

Το 1960 περιγράφηκε από τον R.Bellmann πρώτη λύση με δυναμικό προγραμματισμό, ο οποίος μπόρεσε να λύσει το πρόβλημα για 17 πόλεις. Ο M.F.Dacey κατάφερε να βρει έναν ευρετικό αλγόριθμο, ο οποίος έλυνε το πρόβλημα και η λύση ήταν 4.8% χειρότερη από τη βέλτιστη. Αργότερα χρησιμοποιήθηκαν ευρετικοί αλγόριθμοι με 60 και 10 πόλεις για να λύσουν το πρόβλημα. Σε όλα αυτά τα χρόνια, δημοσιεύσεις για branch and bound αλγορίθμους εφαρμόστηκαν πάνω στο TSP (Little, Murty et al. 1963, Maredia 2010).

2.2 Εφαρμογές και πρακτική χρήση του TSP

Το TSP αφορά πολλούς διαφορετικούς τομείς και συνδέεται με προβλήματα , όπως:

I. **Διάτρηση τυπωμένων κυκλωμάτων** (Drilling of printed circuit boards) : Για τη σύνδεση ενός αγωγού σε μια επιφάνεια, με ένα άλλο αγωγό σε μια διαφορετική επιφάνεια, είναι απαραίτητη η διάτρηση του πίνακα του κυκλώματος. Οι τρύπες ενδέχεται να είναι διαφορετικού μεγέθους. Για την διάτρηση τους, εαν έχουν διαφορετική διάμετρο, το κοπτικό εργαλείο πρέπει να πάει στη βάση και να αλλάξει κεφαλή. Επειδή η διαδικασία αυτή είναι χρονοβόρα, επιλέγεται μια διάμετρος κοπτικού για όλες τις τρύπες, και μετά διαδοχικά διαφορετικές κεφαλές για μεγαλύτερες διαμέτρους. Έτσι το πρόβλημα μπορεί να παρουσιαστεί ως μια σειρά TSP προβλημάτων, ένα για κάθε διάμετρο τρύπας και στόχος είναι η ελαχιστοποίηση του χρόνου του κοπτικού εργαλείου(Lenstra and Kan 1975, Grötschel, Jünger et al. 1991, Matai, Mittal et al. 2010).

II. **Αναμόρφωση κινητήρων αερίου σε στροβιλομηχανές** (overhauling gas turbine engines) : Για να εγγυηθεί μια ομοιόμορφη ροή αερίου διαμέσου των στροβίλων υπάρχει ένα συγκρότημα πτερυγίων με ακροφύσιο σε κάθε βαθμίδα του στροβίλου. Κάθε τέτοιο συγκρότημα αποτελείται από ένα αριθμό οδηγών πτερυγίων γύρω από την περιφέρεια του. Τα πτερύγια αυτά έχουν ατομικά χαρακτηριστικά και η σωστή τοποθέτηση τους παρέχει σημαντικά οφέλη (όπως μείωση κραδασμών, διευκόλυνσης της ομαλής ροής κλπ). Το πρόβλημα τοποθέτησης των πτερυγίων μπορεί να μοντελοποιηθεί ως TSP με συγκεκριμένη αντικειμενική συνάρτηση(Matai, Mittal et al. 2010).

III. **Κρυσταλλογραφία με ακτίνες –Χ** (X ray Crystallography) : Η ανάλυση της δομής των κρυστάλλων είναι μια σημαντική εφαρμογή του TSP. Ένα περιθλασίμετρο (diffractometer) χρησιμοποιείται για να συλλέξει πληροφορίες για τη δομή του κρυσταλλικού υλικού. Για το σκοπό αυτό ένας ανιχνευτής μετρά την ένταση των αντανακλάσεων των ακτίνων Χ σε διάφορες θέσεις. Η μέτρηση αυτή καθ'αυτή πραγματοποιείται πολύ γρήγορα, αλλά δαπανάται πολύ μεγάλος χρόνος στη διαδικασία τοποθέτησης, η οποία μπορεί να χρειαστεί να γίνει εκατοντάδες χιλιάδες φορές. Στα δύο παραδείγματα που αναφέρεται η μελέτη, η διαδικασία τοποθέτησης αφορά τέσσερις κινητήρες. Ο χρόνος μετακίνησης από μια θέση σε μια άλλη μπορεί να υπολογιστεί ακριβώς. Το αποτέλεσμα του πειράματος δεν εξαρτάται από τη σειρά με την οποία γίνονται οι μετρήσεις σε κάθε θέση, αλλά ο συνολικός χρόνος εκτέλεσης του πειράματος εξαρτάται. Έτσι, το πρόβλημα μπορεί να θεωρηθεί ως TSP με σκοπό την ελαχιστοποίηση του συνολικού χρόνου τοποθέτησης(Bland and Shallcross 1989).

IV. **Καλωδίωση υπολογιστών** (Computer wiring) : Μια άλλη περίπτωση είναι η σύνδεση εξαρτημάτων σε ένα πίνακα υπολογιστή. Συγκεκριμένα, ενότητες(modules) βρίσκονται σε ένα πίνακα υπολογιστή και ένα σύνολο καρφιτσών(pins) πρέπει να συνδεθούν . Σε αντίθεση με τη συνηθισμένη περίπτωση, όπου απαιτείται μια σύνδεση δέντρου τύπου Steiner, σε αυτό το πρόβλημα απαιτείται κάθε καρφίτσα συνδέεται με το πολύ δύο καλώδια. Έτσι, το πρόβλημα

είναι η εύρεση Χαμιλτονιανής διαδρομής, χωρίς συγκεκριμένη αφετηρία-τερματισμό. Αντίστοιχη κατάσταση παρατηρείται και κατά την σύνδεση του test-bus(Graham, Lawler et al. 1979, Matai, Mittal et al. 2010)

V. **Order picking στις Αποθήκες** (order-picking in warehouses) : Το πρόβλημα αυτό συνδέεται με τον χειρισμό υλικών σε μια αποθήκη(Ratliff and Rosenthal 1983). Με δεδομένη μια συγκεκριμένη παραγγελία, ένα όχημα πρέπει να συλλέξει όλα τα αντικείμενα της παραγγελίας για να σταλούν στον πελάτη. Οι τοποθεσίες των προϊόντων μέσα στην αποθήκη είναι οι κόμβοι του προβλήματος και η απόσταση μεταξύ των κόμβων δίνεται από το χρόνο που χρειάζεται το όχημα από την τοποθεσία του ενός αντικειμένου μέχρι την τοποθεσία του επόμενου. Έτσι, το πρόβλημα εκφράζεται ως TSP με σκοπό την ελαχιστοποίηση του χρόνου picking(Dal 1992).

VI. **Δρομολόγηση οχημάτων**(Vehicle routing) : Με δεδομένο ένα αριθμό προορισμών - πελατών, σκοπός του προβλήματος είναι να βρει τον ελάχιστο αριθμό οχημάτων που χρειάζονται για την εξυπηρέτηση τους, ούτως ώστε η χωρητικότητα του κάθε φορτηγού να μην παραβιάζεται και η συνολική διανυόμενη απόσταση να ελαχιστοποιείται. Έτσι, το πρόβλημα αυτό μπορεί να εκφραστεί ως ένα αριθμό m TSP προβλημάτων (Maredia 2010).

2.3 Υπολογισμός κάτω ορίων ενός TSP προβλήματος

Πριν την επίλυση οποιουδήποτε προβλήματος TSP, πρέπει να μπορεί να υπολογιστεί πόσο καλή είναι η παραγόμενη λύση. Για αυτό το λόγο, δύναται να χρησιμοποιηθεί μια προσέγγιση της συντομότερης διαδρομής ή αλλιώς το κατώτατο όριο που θα μπορούσε θεωρητικά να λάβει το μήκος της διαδρομής χωρίς όμως να είναι απαραίτητο η τιμή αυτή να αντιστοιχεί σε διαδρομή που τηρεί τους περιορισμούς.

Με δεδομένο ένα πρόβλημα περιοδεύοντος πωλητή, και ένα αριθμό από n προορισμούς που πρέπει να επισκεφθεί (n>>1). Ένα απλό κάτω όριο για την απόσταση μπορεί να οριστεί ως $\frac{1}{2} \cdot \sqrt{n}$. Σε αυτή την περίπτωση, οι προορισμοί είναι κατανεμημένοι σε μια τετραγωνική μονάδα και κάθε προορισμός συνδέεται με το γειτονικό του με μέση απόσταση $\frac{1}{2\sqrt{n}}$. Ένα άλλο όριο που μπορεί να ορισθεί είναι το $(\frac{1}{2} + \frac{3}{4}) \cdot \frac{\sqrt{n}}{2}$, όπου κάθε προορισμός j συνδέεται όπως και πριν με τον πιο κοντινό γειτονικό του με μέση απόσταση $\frac{1}{2\sqrt{n}}$ και με το δεύτερο πιο κοντινό του με απόσταση $\frac{3}{4\sqrt{n}}$. Ο David S. Johnson υπολόγισε ένα κάτω όριο πειραματικά: $0.708 \cdot \sqrt{n}$ + 0.522 (Johnson 1996). Τέλος, οι Christine L. Valenzuela και Antonia J. Jones υπολόγισαν ένα καλύτερο κάτω όριο ως: $0.708 \cdot \sqrt{n}$ + 0.551 (Valenzuela and Jones 1997).

2.4 Επίλυση του TSP

Υπάρχουν αρκετοί τρόποι επίλυσης ενός TSP προβλήματος, μερικοί από τους οποίους θα παρουσιαστούν συνοπτικά στη συνέχεια, αλλά η συγκεκριμένη διπλωματική δεν θα ασχοληθεί ενδελεχώς με αυτούς, εφόσον επικεντρώνεται στην επίλυση ενός VRP προβλήματος με χρήση ενός Υβριδικού Γενετικού αλγορίθμου.

Brute force

Ο ευκολότερος τρόπος επίλυσης είναι η εύρεση όλων των δυνατών εναλλακτικών στο διάστημα των δυνατών λύσεων και η επιλογή της καλύτερης. Παρ'όλα αυτά, ο διαρκώς αυξανόμενος αριθμός δυνατών λύσεων, απαγορεύει αυτό τον τρόπο επίλυσης ακόμα και με σύγχρονους υπολογιστές(Maredia 2010).

Ευρετικές μέθοδοι

Για την επίλυση του TSP, ένας καλός τρόπος προσέγγισης είναι οι ευρετικοί αλγόριθμοι. Αυτοί οι αλγόριθμοι δίνουν μια αρκετά καλή λύση, με σφάλμα 2-3%. Το κυριότερο πλεονέκτημα αυτών των αλγορίθμων είναι ότι μπορούν να λύσουν προβλήματα TSP με χώρο δυνατών λύσεων πολύ μεγαλύτερο από τους υπόλοιπους αλγόριθμους, σε λογικό χρόνο και με λογική απαιτούμενη υπολογιστική ισχύ.

Μεταευρετικές μέθοδοι

Για την επίλυση του TSP, εκτός από ευρετικές μεθόδους, μεγάλη απήχηση βρίσκουν τα τελευταία χρόνια μεταευρετικοί αλγόριθμοι, μερικοί από τους οποίους είναι Γενετικοί αλγόριθμοι(Genetic Algorithm), Προσομειωμένη Ανόπτηση (Simulated Annealing), Αναζήτηση Tabu (Tabu Search), Κβαντική Ανόπτηση (Quantum Annealing), Βελτιστοποίηση σμήνους σωματιδίων(Particle Swarm Optimization), Αναζήτηση Αρμονίας(Harmony Search)(Grefenstette, Gopal et al. 1985, Geem, Kim et al. 2001, Lourenço, Martin et al. 2003).

3 Το πρόβλημα δρομολόγησης Οχημάτων (VRP)

Το πρόβλημα δρομολόγησης οχημάτων (VRP) είναι ένα γενικό όνομα που δίνεται σε μια σειρά προβλημάτων, στα οποία ένα σύνολο οχημάτων εξυπηρετεί ένα σύνολο πελατών. Ο ορισμός αυτός δόθηκε το 1959 από τους Dantzig και Ramser(Dantzig and Ramser 1959). Το VRP είναι μια γενίκευση του προβλήματος του περιοδεύοντος πωλητή που περιγράφηκε στην προηγούμενη ενότητα, όπου μόνο ένας πωλητής περιλαμβάνεται και η χωρητικότητα του υπερβαίνει πάντα την συνολική ζήτηση (Maffioli 2003, Eksioglu, Vural et al. 2009).

3.1 Ορισμός του VRP

Το πρόβλημα δρομολόγησης οχημάτων μπορεί να περιγραφεί ώς εξής:

Με δεδομένο ένα γράφο G=(N,A) και ενός στόλου ομοιογενών οχημάτων $V = (v_1, v_2 \ldots, v_t)$, όπου t είναι ο αριθμός των οχημάτων, N το σύνολο των κόμβων και A το σύνολο των ακμών . Κάθε όχημα που εξυπηρετεί πελάτες ξεκινά από μια κεντρική αποθήκη depot και μετά την εξυπηρέτηση όλων των πελατών που του έχουν ανατεθεί καταλήγει πάλι στην κεντρική αποθήκη. Στόχος του VRP είναι η εύρεση εκείνης της λύσης που ελαχιστοποιεί αρχικά τον αριθμό των χρησιμοποιούμενων οχημάτων και επιπρόσθετα το μήκος της συνολικής διανυόμενης απόστασης(Dantzig and Ramser 1959, Potvin and Bengio 1996, Tan, Lee et al. 2001, Jung and Moon 2002, Ombuki, Nakamura et al. 2002, Alvarenga, De Abreu Silva et al. 2005, Yeun, Ismail et al. 2008).

Για το σύνολο των ακμών A, ένας πίνακας κόστους D κατασκευάζεται, τέτοιος ώστε d_{ij} να είναι το κόστος της ακμής (n_i n_j) και d_{ii}=0. Συνήθως, στο VRP υπάρχει συμμετρικότητα, δηλαδή d_{ij}=d_{ji}. Πάραυτα στον πραγματικό κόσμο, ο πίνακας του κόστους δεν είναι συμμετρικός και πρέπει να υπολογίζεται ξεχωριστά με βάση τα γεωγραφικά δεδομένα. Επιπρόσθετα, το σύνολο των οχημάτων δεν είναι ομοιογενές, και υπάρχουν δρόμοι αποκλειστικά για μερικούς τύπους οχημάτων.

Μια σειρά περιορισμών μπορούν να προστεθούν στο VRP, για να προσομοιώσει καλύτερα καταστάσεις στον πραγματικό κόσμο. Συνεπώς, δημιουργήθηκαν παραλλαγές του VRP που περιλαμβάνουν ποικίλες ανάγκες και υπηρεσίες που ζητούν οι πελάτες για την εξυπηρέτησή τους, τα χαρακτηριστικά του στόλου οχημάτων, των αποθηκών και του δικτύου που τα συνδέει, την αβεβαιότητα των προκαθορισμένων δεδομένων και τον χρονικό ορίζοντα προγραμματισμού της δρομολόγησης. Στις περισσότερες περιπτώσεις, οι αντικειμενικοί στόχοι είναι πρώτον η ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και δεύτερον η ελαχιστοποίηση της συνολικά διανυόμενης απόστασης, όπως ακριβώς και στο απλό VRP(Labadie and Prins 2012). Οι βασικές παραλλαγές του VRP περιγράφονται αναλυτικά στη συνέχεια.

3.2 Παραλλαγές του VRP

3.2.1 VRP με πολλές αποθήκες (MDVRP)

Το VRP με πολλές αποθήκες (Multiple depots VRP-MDVRP) αποτελεί μια απλή παραλλαγή του VRP, όπου οι πελάτες χωρίζονται σε αποθήκες, από τις οποίες θα εξυπηρετηθούν. Κάθε αποθήκη έχει τα δικά της οχήματα, τα οποία ξεκινούν από αυτή, εξυπηρετούν το σύνολο των πελατών που τους ανατίθεται και επιστρέφουν στην κεντρική αποθήκη(Tillman and Cain 1972, Wren and Holliday 1972, Gillett and Johnson 1976). Στόχος του προβλήματος είναι η ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και της συνολικά διανυόμενης απόστασης ικανοποιώντας όμως τους ακόλουθους περιορισμούς(Baldacci and Mingozzi 2009, Kuo and Wang 2012, Escobar, Linfati et al. 2014):

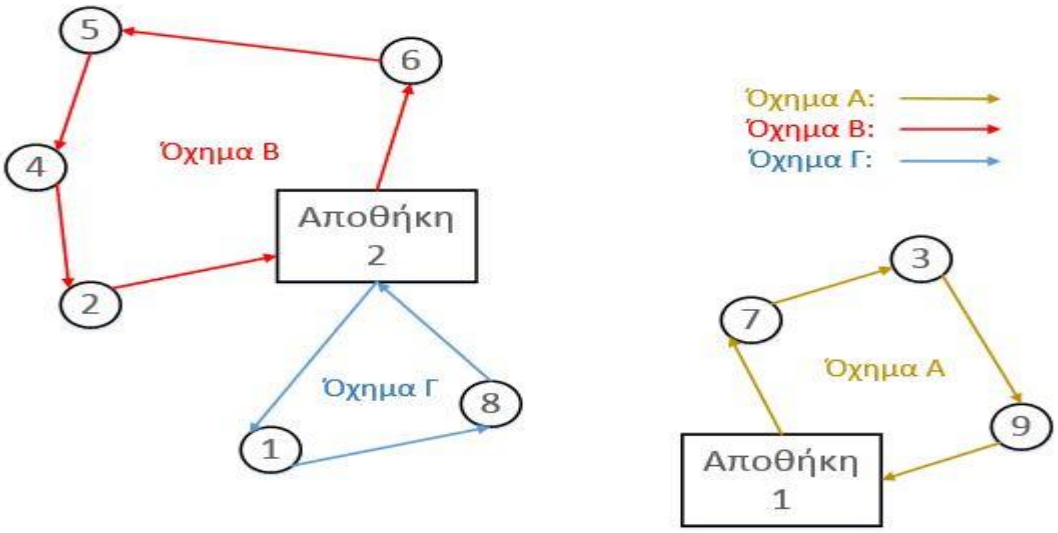
I. Τα οχήματα στο τέλος του δρομολογίου που εκτελούν για την εξυπηρέτηση των πελατών επιστρέφουν στην αποθήκη από την οποία ξεκίνησαν.

II. Η συνολική ζήτηση που ικανοποιεί το κάθε όχημα δεν πρέπει να ξεπερνά τη χωρητικότητα του οχήματος αυτού.

III. Κάθε πελάτης πρέπει να εξυπηρετηθεί από ακριβώς ένα όχημα.

IV. Το πλήθος των οχημάτων που ξεκινούν τη διαδρομή τους από την κάθε αποθήκη δεν πρέπει να ξεπερνούν μία προκαθορισμένη τιμή.

V. Η συνολική διάρκεια της διαδρομής του κάθε οχήματος δεν πρέπει να ξεπερνά μία προκαθορισμένη τιμή.

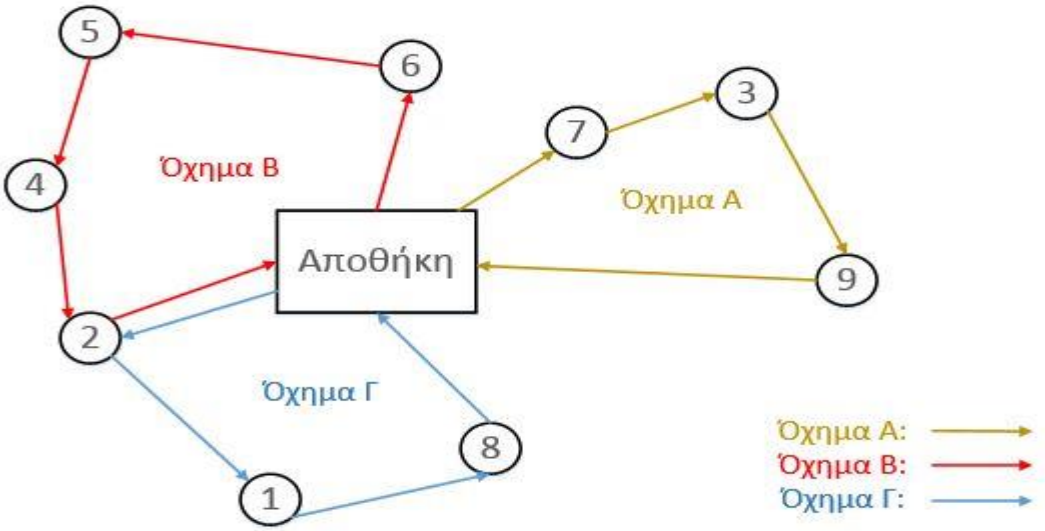


Σχήμα 1 Παράδειγμα MDVRP

3.2.2 VRP με πολλαπλές παραδόσεις (SDVRP)

Το πρόβλημα δρομολόγησης με πολλαπλές παραδόσεις (split deliveries VRP-SDVRP) είναι μια παραλλαγή του κλασικού VRP, στην οποία ένας πελάτης μπορεί να εξυπηρετηθεί από περισσότερα του ενός οχήματα, εαν αυτό μειώνει το συνολικό κόστος(Archetti, Speranza et al. 2006, Archetti and Speranza 2012). Επιπλέον, προτείνεται η εισαγωγή πελατών των οποίων η ζήτηση ξεπερνά τη χωρητικότητα ενός οχήματος και επομένως, πρέπει αναγκαστικά να εξυπηρετηθούν από περισσότερα οχήματα(Archetti, Savelsbergh et al. 2006). Με βάση την παραπάνω παραδοχή, μπορούν να ορισθούν μια σειρά παραλλαγών του VRP :

I. VRP: το κλασικό πρόβλημα βελτιστοποίησης, όπου η ζήτηση κάθε πελάτη είναι μικρότερη από τη χωρητικότητα του οχήματος και κάθε πελάτης εξυπηρετείται μόνο μια φορά.
II. SDVRP: το πρόβλημα αυτό αναιρεί την απαίτηση, ότι κάθε πελάτης θα εξυπηρετείται μόνο μια φορά, αλλά διατηρεί την απαίτηση, ότι η ζήτηση κάθε πελάτη θα είναι το πολύ ίση με τη χωρητικότητα του οχήματος.
III. VRP+: το πρόβλημα αυτό αναιρεί την απαίτηση, ότι η ζήτηση κάθε πελάτη θα είναι μικρότερη από τη χωρητικότητα, αλλά απαιτεί ο αριθμός επισκέψεων στον πελάτη να είναι ο ελάχιστος δυνατός, δηλαδή η συνολική ζήτηση του πελάτη ως προς τη χωρητικότητα του οχήματος στρογγυλοποιημένη προς τα πάνω στον κοντινότερο ακέραιο αριθμό.
IV. SDVRP+: η τελευταία παραλλαγή των split deliveries η οποία αναιρεί τόσο την απαίτηση, ότι η ζήτηση θα είναι μικρότερη της μέγιστης χωρητικότητας, όσο και την απαίτηση, ότι κάθε πελάτης θα εξυπηρετηθεί τις ελάχιστες φορές.



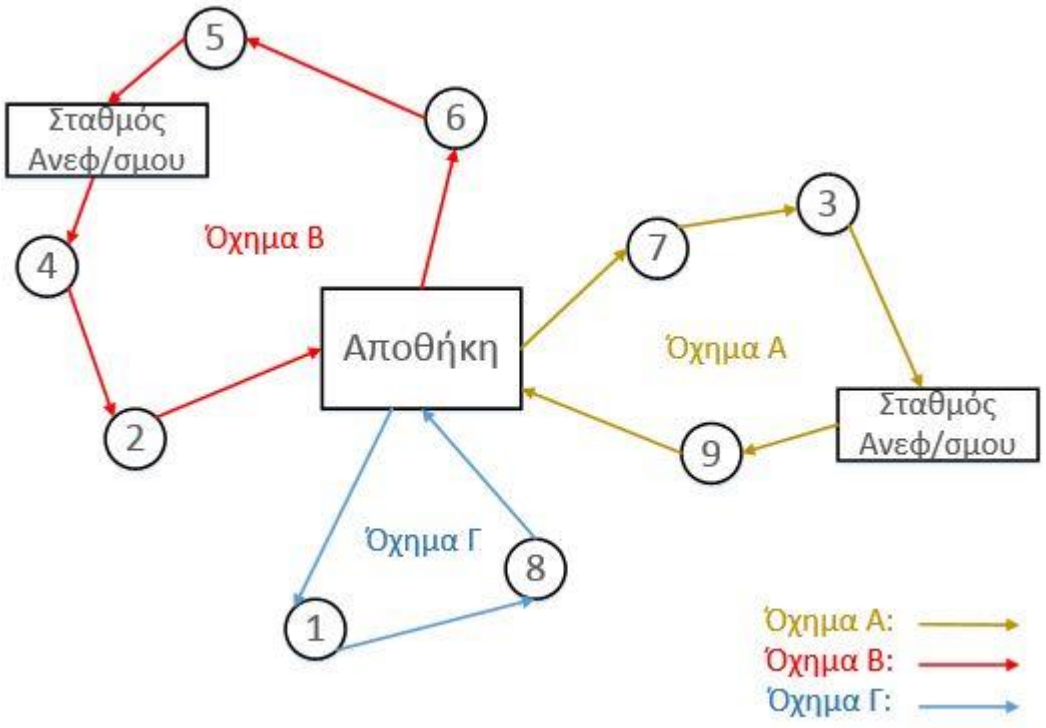
Σχήμα 2 Παράδειγμα SDVRP

3.2.3 VRP με εγκαταστάσεις ανεφοδιασμού (VRPSF)

Το πρόβλημα δρομολόγησης με εγκαταστάσεις ανεφοδιασμού(VRP with satellite facilities-VRPSF), είναι μια ακόμα παραλλαγή του κλασικού VRP, στην οποία οι οδηγοί μπορούν να επισκεφθούν εγκαταστάσεις, προκειμένου να αναπληρώσουν τα προϊόντα τους και να συνεχίσουν τις μεταφορές μέχρι το τέλος της βάρδιας τους, χωρίς όμως να χρειάζεται να επιστρέψουν στην κεντρική αποθήκη. Η κατάσταση αυτή συναντάται κυρίως στη διανομή καυσίμων και ορισμένων retail προϊόντων. Όταν η ζήτηση είναι τυχαία επιλέγοντας τα δρομολόγια εκ των προτέρων, αυτό μπορεί να οδηγήσει σε μεγάλα επιπλέον κόστη. Επομένως, τέτοιοι σταθμοί ανεφοδιασμού αποτελούν ένα τρόπο προστασίας ως προς την αβεβαιότητα της ζήτησης(Bard, Huang et al. 1998, Bard, Huang et al. 1998).

Σε ένα VRP 2 επιπέδων(2 echelon VRP,2E-VRP) ορίζονται δύο επίπεδα δρομολόγησης, ένα που αφορά την παράδοση από την κεντρική αποθήκη στους σταθμούς ανεφοδιασμού, και ένα δεύτερο που αφορά την παράδοση από τους σταθμούς στους πελάτες(Crainic, Perboli et al. 2010).

Για την επίλυση του συγκεκριμένου προβλήματος έχουν αναπτυχθεί αρκετοί ευρετικοί αλγόριθμοι, με κυριότερους τους Randomized Clarke-Wright, GRASP, modified Sweep(Bard, Huang et al. 1998).



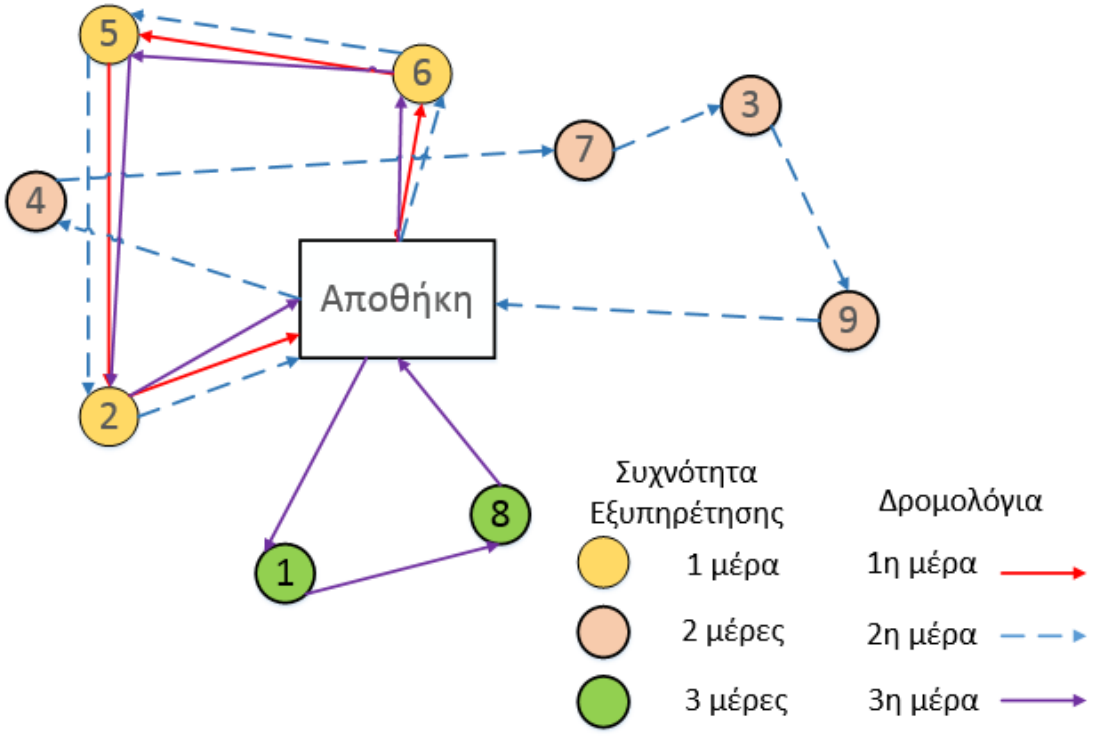
Σχήμα 3 Παράδειγμα VRPSF

3.2.4 Το VRP με πολλαπλές περιόδους (PVRP)

Το VRP με πολλαπλές περιόδους (periodic VRP-PVRP) διαφέρει αρκετά σε σχέση με την κλασική εκδοχή του VRP. Συγκεκριμένα, ενώ στην απλή εκδοχή του, τα οχήματα πρέπει να εξυπηρετήσουν όλους τους πελάτες σε μια ημέρα, στην παρούσα παραλλαγή του, στόχος είναι να σχεδιαστούν δρομολόγια για κάθε μέρα μιας περιόδου p ημερών. Κάθε πελάτης απαιτεί έναν αριθμό επισκέψεων k σε κάθε τέτοια περίοδο, οι οποίες μπορούν να γίνουν από οποιονδήποτε επιτρεπτό συνδυασμό k ημερών. Για παράδειγμα, όπως αναφέρουν οι Christofides και Beasly, αν κάποιος πελάτης απαιτεί 2 επισκέψεις σε μια χρονική περίοδο 5 ημερών, οι επιτρεπτοί συνδυασμοί μπορούν να είναι Δευτέρα-Παρασκευή ή Δευτέρα-Πέμπτη ή Τρίτη-Παρασκευή, αλλά όλοι οι υπόλοιποι συνδυασμοί να είναι απαγορευτικοί λόγω της φύσης του προβλήματος (Christofides and Beasley 1984, Baptista, Oliveira et al. 2002, Yu and Yang 2011).

Για την καλύτερη κατανόηση του προβλήματος PVRP παρουσιάζεται στο παρακάτω σχήμα ένα παράδειγμα δρομολόγησης με προγραμματισμό 3 ημερών. Κάθε πελάτης απαιτεί εξυπηρέτηση, συγκεκριμένο αριθμό ημερών σε κάθε περίοδο, όπως φαίνεται στο σχήμα.

Για το PVRP έχουν αναπτυχθεί αρκετοί ευρετικοί και μεταευρετικοί αλγόριθμοι, και έχει μελετηθεί εκτενώς τα τελευταία χρόνια, με αρκετές παραλλαγές (Amberg, Domschke et al. 2000, Francis, Smilowitz et al. 2008, Ombuki-Berman and Hanshar 2009).



Σχήμα 4 Παράδειγμα PVRP

3.2.5 VRP με παράθυρα χρόνου (VRPTW)

Το VRP με παράθυρα χρόνου(VRP with time windows-VRPTW) είναι μια από τις πιο μελετημένες παραλλαγές του VRP, κατά την οποία οι πελάτες θέλουν η παράδοση των προϊόντων να γίνεται μέσα σε συγκεκριμένο χρονικό πλαίσιο. Το χρονικό πλαίσιο αυτό έχει μια αρχή(e_i) και ένα τέλος (l_i) για κάθε πελάτη. Η παραλλαγή αυτή είναι εμπνευσμένη από τον πραγματικό κόσμο, στον οποίο τα καταστήματα, λόγω των πολλών προμηθευτών και των ωραρίων λειτουργίας τους απαιτούν η παράδοση να γίνεται σε ένα συγκεκριμένο και σχετικά μικρό χρονικό διάστημα (Solomon 1984, Thangiah 1993, Cordeau and décisions 2000).

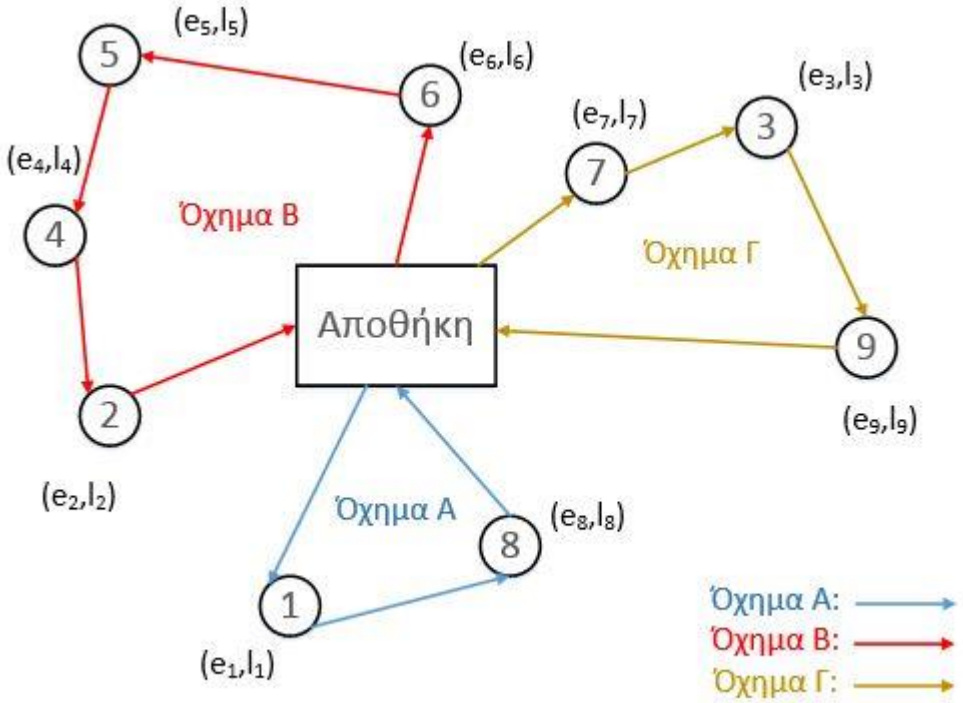
Έχουν μελετηθεί διάφορες παραλλαγές του VRPTW, μερικές από τις οποίες παρουσιάζονται συνοπτικά στη συνέχεια:

1. VRP με χρονικές προθεσμίες (VRP with deadlines-VRPTD): Σε αυτή την παραλλαγή, κάθε πελάτης έχει μια αργότερη προθεσμία στην οποία μπορεί να δεχτεί την παραγγελία. Η διαφορά με το VRPTW έγκειται στο γεγονός ότι δεν υπάρχει κάτω όριο για την παραλαβή(Thangiah, Osman et al. 1993, Thangiah, Vinayagamoorty et al. 1993).
2. VRP με ελαστικά παράθυρα χρόνου (VRP with soft time windows-VRPSTW) : Σε αυτή την παραλλαγή, οι πελάτες μπορούν να δεκτούν παραγγελία εκτός του χρονικού πλαισίου που έχουν ορίσει, αλλά με μια επιπλέον επιβάρυνση(penalty cost)(Toth and Vigo 2002).
3. VRP με αυστηρά παράθυρα χρόνου(VRP with hard time windows-VRPHTW): Σε αντίθεση με την προηγούμενη παραλλαγή του VRP, σε αυτό το πρόβλημα, δεν επιτρέπεται τα οχήματα να φθάσουν σε κάποιο πελάτη μετά το πέρας του χρονικού παραθύρου(Chen, Chen et al. 2000).
4. VRP εξαρτώμενο από το χρόνο (time dependent VRP-TDVRP) : Σε αυτό το VRP, όλες οι παράμετροι εξαρτώνται από το χρόνο και επηρεάζονται από καταστάσεις που συμβαίνουν στον πραγματικό κόσμο όπως ώρες αιχμής(Ichoua, Gendreau et al. 2003).

Το VRP με παράθυρα χρόνου είναι από τις πιο μεγάλες προκλήσεις στη δρομολόγηση οχημάτων. Αρκετοί μελετητές θεωρούν ως πρωταρχικό στόχο την ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και ως δευτερεύοντα στόχο την ελαχιστοποίηση της συνολικά διανυόμενης απόστασης. Ο χρόνος για την εκτέλεση των δρομολογίων δεν ελαχιστοποιείται και χρησιμοποιείται για τον έλεγχο ικανοποίησης των χρονικών παραθύρων των πελατών είτε πρόκειται για αυστηρά παράθυρα χρόνου(hard time windows), είτε για ελαστικά παράθυρα χρόνου(soft time windows).

Αρκετοί αλγόριθμοι έχουν αναπτυχθεί για την επίλυση του VRPTW, τόσο ευρετικοί, όσο και μεταευρετικοί, για προβλήματα 100 και παραπάνω πελατών(Jepsen, Petersen et al. 2008, Labadi, Prins et al. 2008, Nagata, Bräysy et al. 2010).

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα VRPTW με 9 πελάτες και 3 οχήματα. Κάθε πελάτης έχει ένα παράθυρο χρόνο, το οποίο ορίζεται από την νωρίτερη παραλαβή(e_i) και την αργότερη παραλαβή(l_i).



Σχήμα 5 Παράδειγμα VRPTW

3.2.6 VRP με στόλο ανομοιογενών οχημάτων (HVRP)

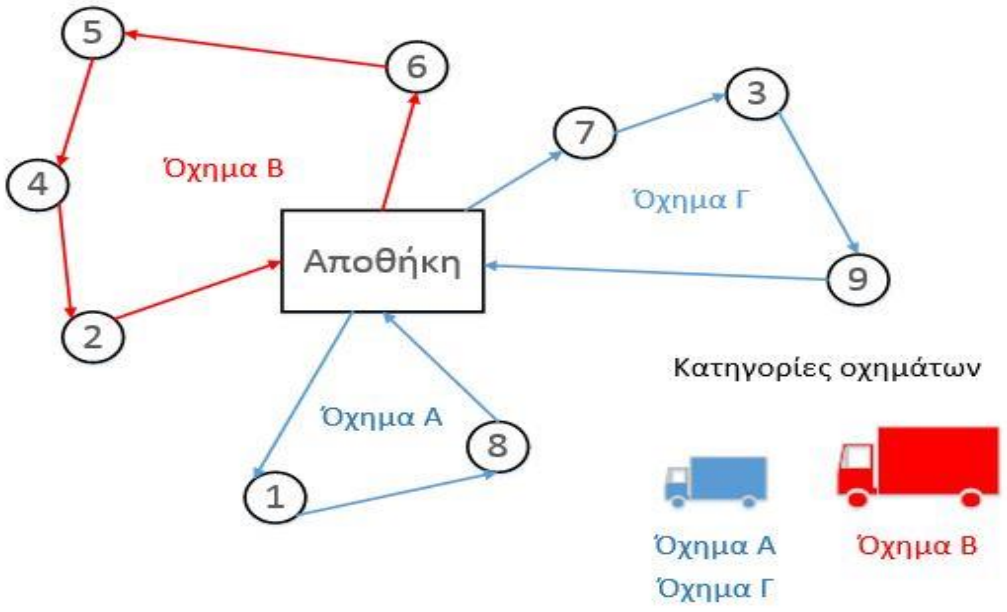
Το VRP με στόλο ανομοιογενών οχημάτων (Heterogeneous Fleet VRP-HVRP) είναι μια παραλλαγή του VRP κατά την οποία οι πελάτες εξυπηρετούνται από ένα στόλο οχημάτων διαφορετικών χαρακτηριστικών, όπως χωρητικότητα(capacity), σταθερό και μεταβλητό κόστος(fixed and variable cost), κατανάλωση καυσίμου(fuel consumption), αριθμό διαθέσιμων οχημάτων(fleet size)(Gendreau, Laporte et al. 1999, Lima, Goldbarg et al. 2004, Brandão 2011, Subramanian, Penna et al. 2012, Soonpracha, Mungwattana et al. 2014)

Σύμφωνα με τους Subramanian, Penna, Uchoa, και Ochi, ακόμη και αν ένας στόλος είναι ομοιογενής, στο μέλλον μπορεί να μετατραπεί σε ετερογενή, όταν νέα οχήματα αποκτηθούν. Επιπρόσθετα, κόστη όπως ασφάλειες, συντήρηση, κόστη λειτουργίας μπορούν να αποκτήσουν διαφορετικές τιμές, ανάλογα με το χρόνο χρήσης του στόλου. Έτσι διακρίνονται οι ακόλουθες παραλλαγές του HVRP(Subramanian, Penna et al. 2012):

1. **HVRPFV** με περιορισμένο αριθμό οχημάτων που απαρτίζουν το στόλο, σταθερά κόστη για την χρήση ενός οχήματος και μεταβλητά κόστη ανάλογα με την διανυόμενη απόσταση κάθε οχήματος.
2. **HVRPV** με περιορισμένο αριθμό οχημάτων που απαρτίζουν το στόλο, μεταβλητά κόστη ανάλογα με τη διανυόμενη απόσταση κάθε οχήματος, αλλά χωρίς σταθερά κόστη.
3. **FSMFV** με απεριόριστο αριθμό οχημάτων που απαρτίζουν το στόλο, σταθερά κόστη για την χρήση ενός οχήματος και μεταβλητά κόστη ανάλογα με την διανυόμενη απόσταση κάθε οχήματος.
4. **FSMF** με απεριόριστο αριθμό οχημάτων, με σταθερά κόστη για την χρήση ενός οχήματος, αλλά χωρίς μεταβλητά κόστη.
5. **FSMV** με απεριόριστο αριθμό οχημάων, με μεταβλητά κόστη ανάλογα με τη διανυόμενη απόσταση αλλά χωρίς σταθερά κόστη.

Για την επίλυση του HVRP έχουν αναπτυχθεί τόσο ευρετικοί όσο και μεταευρετικοί αλγόριθμοι, όπως Υβριδικοί Γενετικοί Αλγόριθμοι, επιτυγχάνοντας ένα καλό επίπεδο λύσεων. Ταυτόχρονα, αναπτύχθηκε ένας επαναληπτικός αλγόριθμος τοπικής αναζήτησης(iterated local search), συνδυαζόμενος με μια μεταβλητή διαδικασία γειτνίασης(Variable Neighborhood Decent procedure) και τυχαία διάταξη γειτόνων(random neighborhood ordering), ο οποίος μπορεί να λύσει κάθε παραλλαγή του HFVRP(Penna, Subramanian et al. 2013). Πάραυτα, λόγω της φύσης του προβλήματος δεν υπάρχουν αναλυτικοί αλγόριθμοι επίλυσης(Subramanian, Penna et al. 2012).

Άλλοι μελετητές εισάγουν την έννοια της φόρτωσης σε δύο διαστάσεις (2L-HVRP), προσομοιώνοντας ακόμη καλύτερα ένα πρόβλημα από τον πραγματικό κόσμο(Dominguez, Juan et al. 2014).



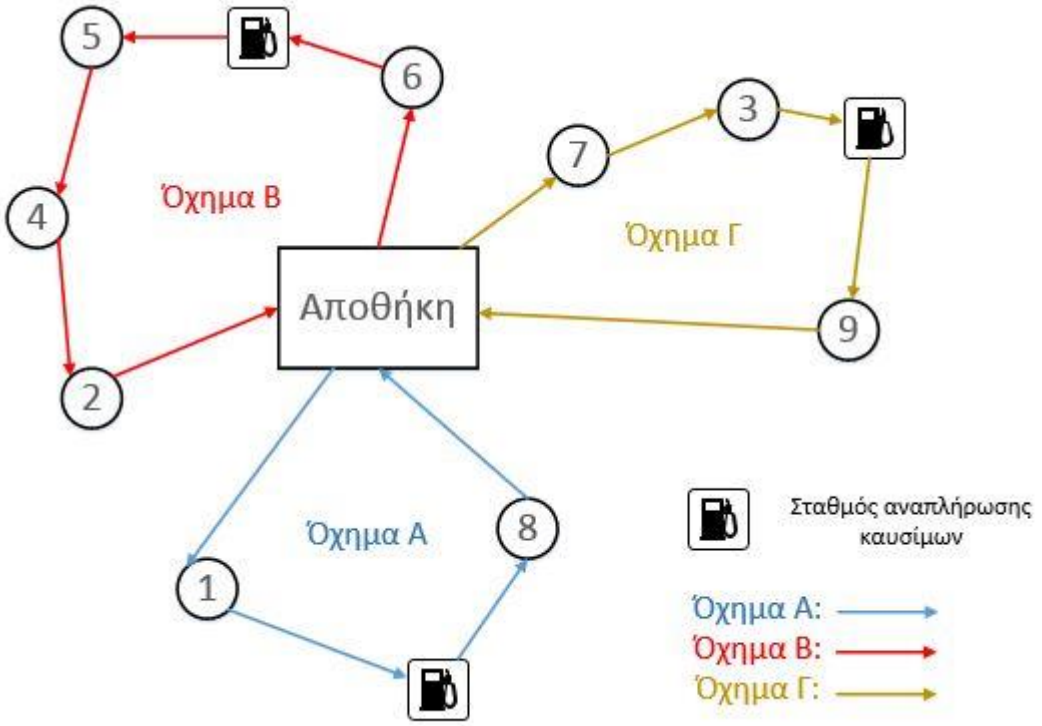
Σχήμα 6 Παράδειγμα HVRP

3.2.7 Πράσινο VRP (G-VRP)

Το ‘’Πράσινο’’ VRP(Green VRP-GVRP) είναι μια νέα εκδοχή του VRP,που αναπτύχθηκε τα τελευταία χρόνια και προέκυψε από την ανάγκη για μείωση των εκπομπών και κατεύθυνση σε μια πιο φιλική προς το περιβάλλον προσέγγιση. Συγκεκριμένα, ο τομέας των μεταφορών, σύμφωνα με έρευνα για τις ΗΠΑ είναι υπεύθυνος για το 28% των εκπομπών αέριου του θερμοκηπίου(greenhouse gas-GHG). Έτσι, γίνεται προσπάθεια από τη μια, να μειωθούν οι διανυόμενες αποστάσεις, και από την άλλη εισαγωγή νέων πιο καθαρών καυσίμων, τόσο για μείωση των εκπομπών ανά διανυόμενο χιλιόμετρο, όσο και για μείωση της κατανάλωσης καυσίμου(Erdoğan and Miller-Hooks 2012).

Το GVRP σε μια εκδοχή του ασχολείται με την εύρεση και χρησιμοποίηση σταθμών καυσίμων για ανεφοδιασμό, καθώς και ελαχιστοποίηση των διανυόμενων διαδρομών. Το GVRP αποτελεί πρόβλημα μικτού ακέραιου γραμμικού προγραμματισμού και εκτός από τους πελάτες και την κεντρική αποθήκη περιλαμβάνει έναν ή περισσότερους σταθμούς καυσίμων, στους οποίους τα οχήματα μπορούν να σταματήσουν, προκειμένου να συνεχίσουν τα δρομολόγια τους, χωρίς όμως να επιστρέψουν στην κεντρική αποθήκη(Erdoğan and Miller-Hooks 2012).

Για την καλύτερη κατανόηση του προβλήματος, δημιουργήθηκε το παρακάτω σχήμα με 9 πελάτες, 3 σταθμούς ανεφοδιασμού καυσίμων και 3 οχήματα.



Σχήμα 7 Παράδειγμα GVRP

Άλλες παραλλαγές του GVRP περιλαμβάνουν τη χρησιμοποίηση ηλεκτρικών οχημάτων (electric VRP-EVRP)(Schneider, Stenger et al. 2014).

Η έρευνα σε αυτό το πρόβλημα είναι σχετικά περιορισμένη επειδή άρχισε μόλις τα τελευταία 10 χρόνια. Οι αλγόριθμοι που έχουν αναπτυχθεί περιλαμβάνουν ευρετικούς αλγορίθμους, όπως Modified Clark&Wright και Density based Clustering Algorithm(DBCA) με γρήγορες και αρκετά καλές λύσεις, για να βοηθήσουν εταιρείες στην καλύτερη επιλογή των αποφάσεων τους και να δώσουν μια περιβαλλοντική διάσταση στο πρόβλημα. Οι αλγόριθμοι αυτοί μπορούν να αποτελέσουν βάση για πιο σύνθετους μεταευρετικούς αλγορίθμους , όπως μια αναζήτηση Tabu(Erdoğan and Miller-Hooks 2012).

3.2.8 Στοχαστικό VRP

Τα περισσότερα προβλήματα των παραλλαγών του VRP, προϋποθέτουν ντετερμινιστικά δεδομένα για την επίλυση τους, δηλαδή έχουν μια συγκεκριμένη τιμή, αμετάβλητη κατά τη διάρκεια, αλλά και μετά το τέλος των δρομολογίων. Παρ'όλα αυτά στην πράξη, δεδομένα όπως ο χρόνος του δρομολογίου από και προς την αποθήκη, η ζήτηση των πελατών, ο χρόνος εξυπηρέτησης τους, ακόμα και το πλήθος των πελατών και των οχημάτων εμπεριέχουν την έννοια της αβεβαιότητας και συνήθως μεταβάλλονται κατά την εκτέλεση των δρομολογίων. Έτσι, δημιουργήθηκε μια νέα παραλλαγή του VRP, γνωστή ως Στοχαστικό VRP (Stochastic VRP-SVRP).

Ένας από τους πρώτους μελετητές, ο Tillman, πρότεινε ένα αλγόριθμο για το πρόβλημα SVRP με πολλαπλές αποθήκες(Multi-Depot Stochastic VRP). Το συγκεκριμένο πρόβλημα περιλαμβάνει τις παραδοχές του VRP με πολλαπλές αποθήκες που αναλύθηκε νωρίτερα, αλλά επιπλέον η ζήτηση κάθε πελάτη δεν είναι ντετερμινιστικά καθορισμένη, ακολουθεί κανονική κατανομή και έχει μια μέση τιμή και μια τυπική απόκλιση(Moghaddam, Ruiz et al. 2012).

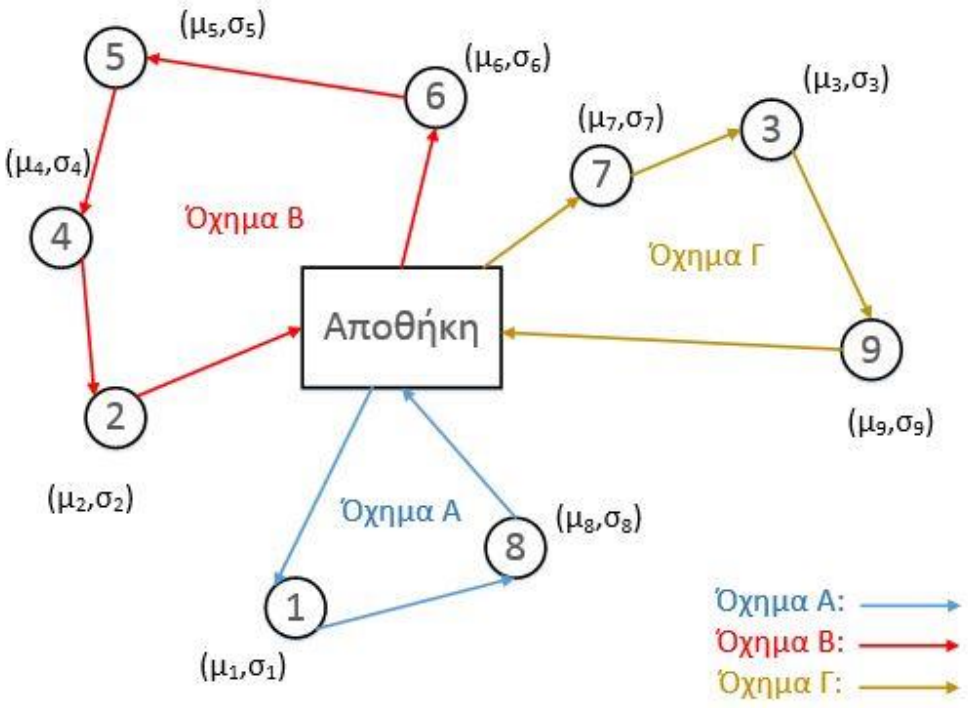
Γενικά, σε όλα τα προβλήματα SVRP, η αβεβαιότητα των αρχικών δεδομένων μοντελοποιείται από μια στατιστική κατανομή, με γνωστή μέση τιμή και τυπική απόκλιση. Σύμφωνα με τους μελετητές (Cordeau, Laporte et al. 2006, Pillac, Gendreau et al. 2013)οι παράγοντες αβεβαιότητας μπορούν να χωριστούν σε τρείς κατηγορίες:

1. Αβεβαιότητα εξυπηρέτησης πελατών: Κάθε πελάτης διέπεται από συγκεκριμένη πιθανότητα εξυπηρέτησης ή μη εξυπηρέτησης(Bertsimas 1992).
2. Αβεβαιότητα των χρόνων εξυπηρέτησης και των χρόνων δρομολογίων: Ο απαιτούμενος χρόνος εξυπηρέτησης κάθε πελάτη, αλλά και ο απαιτούμενος χρόνος δρομολογίου για να φτάσει το όχημα στον πελάτη, ακολουθούν κάποια κατανομή, με μέση τιμή και τυπική απόκλιση(Laporte, Louveaux et al. 1992, Kenyon and Morton 2003, Verweij, Ahmed et al. 2003).
3. Αβεβαιότητα της ζήτησης των πελατών: Η ζήτηση κάθε πελάτη, ενδέχεται να αλλάξει ακόμα και κατά τη διάρκεια του δρομολογίου(Dror, Laporte et al. 1989, Laporte, Louveaux et al. 2002, Christiansen and Lysgaard 2007, Mendoza, Castanier et al. 2010).

Οι κυριότερες προσεγγίσεις που ακολουθούνται για την επίλυση των SVRP είναι ο προγραμματισμός με πιθανοτικούς περιορισμούς(Chance Constrained Programming-CCP) και ο στοχαστικός προγραμματισμός με ανάδραση(Stochastic Programming with Recourse). Οι δύο αυτές μέθοδοι, βασίζονται σε διαδικασία δύο φάσεων, όπου στην πρώτη υπολογίζεται μια αρχική λύση για τη δρομολόγηση και στη δεύτερη, πραγματοποιούνται οι απαραίτητες διορθωτικές κινήσεις κατά τη διάρκεια του δρομολογίου. Η διαφορά των δύο μεθόδων έγκειται στο γεγονός ότι η πρώτη θέτει ένα άνω όριο στην πιθανότητα αποτυχίας και δεν ασχολείται καθόλου με το αναμενόμενο κόστος από τις διορθωτικές κινήσεις της δεύτερης φάσης, ενώ η δεύτερη μέθοδος προσπαθεί να ελαχιστοποιήσει το κόστος των διορθωτικών κινήσεων της δεύτερης φάσης. Με άλλα λόγια, διαφέρουν ως προς τον αντικειμενικό στόχο της πρώτης φάσης(Gendreau, Laporte et al. 1996, Labadie and Prins 2012).

Εναλλακτικές μέθοδοι επίλυσης έχουν αναπτυχθεί από διάφορους μελετητές(Gendreau, Laporte et al. 1996, Roberts and Hadjiconstantinou 1998, Park and Hong 2003, Sungur, Ordónez et al. 2008, Shen, Ordónez et al. 2009).

Στο παρακάτω σχήμα, παρουσιάζεται ένα παράδειγμα δρομολόγησης με στοχαστική ζήτηση των πελατών. Η ζήτηση κάθε πελάτη έχει μέση τιμή μ_i και τυπική απόκλιση σ_i.



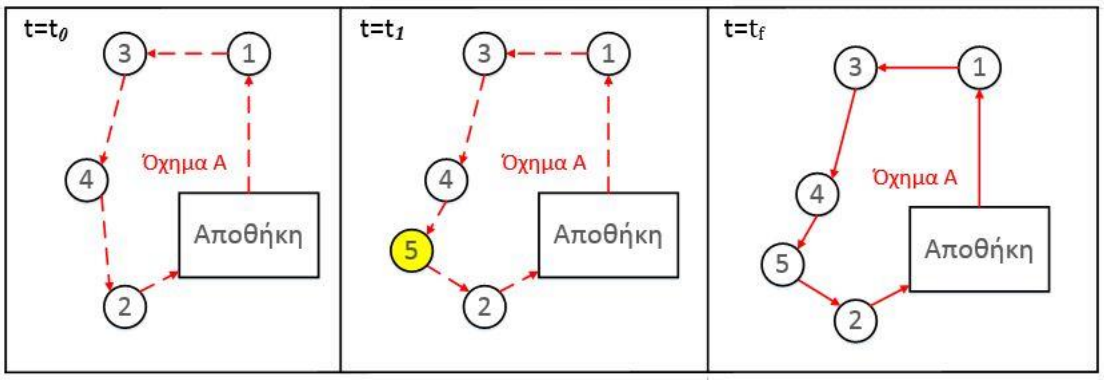
Σχήμα 8 Παράδειγμα SVRP

3.2.9 Δυναμικό VRP

Το δυναμικό VRP (Dynamic VRP-DVRP) είναι ακόμη μια παραλλαγή του VRP, κατά την οποία υπάρχει η δυνατότητα εισαγωγής νέων πελατών στο υπάρχον δρομολόγιο είτε κατά το σχεδιασμό, είτε ακόμα και κατά τη διάρκεια εκτέλεσης του δρομολογίου. Πρόκειται για ένα πρόβλημα που συναντάται πολύ συχνά στον πραγματικό κόσμο, λόγω της μη στατικότητας των δεδομένων που θεωρείται δεδομένη στις λοιπές παραλλαγές του VRP. Για την επίλυση τέτοιων προβλημάτων, ανά τακτά χρονικά διαστήματα, εκτελείται αλγόριθμος που προσθέτει τα νέα δεδομένα στο μαθηματικό μοντέλο και βελτιστοποιεί την αρχική διαδρομή (Pillac, Gendreau et al. 2013).

Το δυναμικό VRP συναντάται πρώτη φορά το 1977 από τους Wilson&Colvin στο δυναμικό πρόβλημα τόξων που σχετίζεται όμως άμεσα με το DVRP. Εκτός από την ελαχιστοποίηση των οχημάτων και της συνολικής διανυόμενης απόστασης, το DVRP προσπαθεί να ικανοποιήσει και άλλες απαιτήσεις, όπως την ελαχιστοποίηση του χρόνου απόκρισης του συστήματος, την ελαχιστοποίηση των απορριπτόμενων παραγγελιών, αλλά και άλλες απαιτήσεις ανάλογα με το υπό μελέτη πρόβλημα(Koskosidis, Powell et al. 1992, Gendreau, Guertin et al. 1999, van Hemert and La Poutré 2004, Montemanni, Gambardella et al. 2005, Novoa and Storer 2009, Wang, Huang et al. 2011, Ferrucci, Bock et al. 2013).

Το DVRP βρίσκει εφαρμογή σε διάφορα προβλήματα, όπως η δρομολόγηση μέσων μαζικής μεταφοράς και η διανομή φαρμάκων από φαρμακευτικές εταιρείες(Psaraftis 1995, de Magalhães and De Sousa 2006). Στο παρακάτω σχήμα φαίνεται αναλυτικά ένα παράδειγμα δυναμικού VRP. Αρχικά, το πλάνο περιλαμβάνει 4 πελάτες και κατά τη διάρκεια του δρομολογίου εμφανίζεται ένας ακόμη πελάτης. Το πλάνο δρομολόγησης τροποποιείται ανάλογα.



Σχήμα 9 Παράδειγμα DVRP

3.2.10 VRP με πολλαπλά δρομολόγια

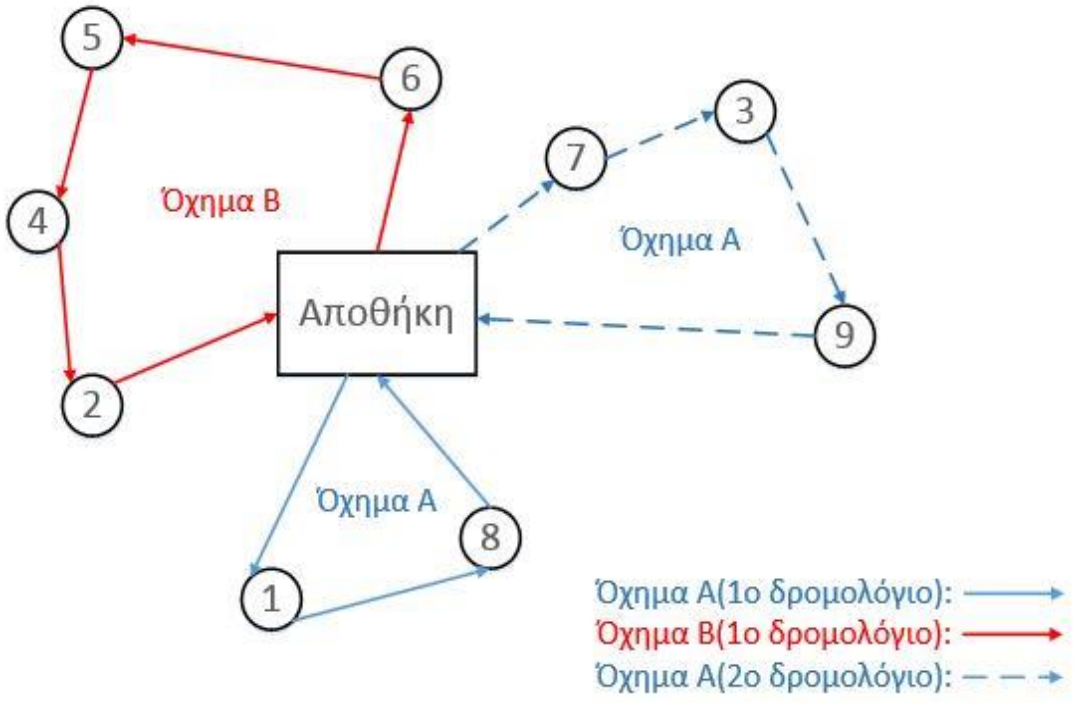
Το VRP με πολλαπλά δρομολόγια (Multi-trip VRP-MTVRP), δίνει τη δυνατότητα στα οχήματα, αφού επιστρέψουν στην κεντρική αποθήκη, να επαναχρησιμοποιηθούν σε νέα δρομολόγια, στην περίπτωση που ο απαιτούμενος αριθμός δρομολογίων είναι μεγαλύτερος από το μέγεθος του στόλου.

Οι περιορισμοί που διέπουν το MTVRP σύμφωνα με τους Cattaruzza, Absi, Feillet, και Vidal είναι:

1. Κάθε πελάτης εξυπηρετείται ακριβώς από ένα όχημα.
2. Το άθροισμα των ζητήσεων όλων των πελατών στην ίδια διαδρομή δεν μπορούν να ξεπερνούν τη χωρητικότητα του οχήματος.
3. Η συνολική διανυόμενη απόσταση όλων των διαδρομών κάθε οχήματος δεν μπορεί να ξεπερνά ένα όριο.
4. Κάθε διαδρομή πάντα ξεκινά και καταλήγει στην κεντρική αποθήκη.

Οι εφαρμογές που χρησιμοποιούν αυτό το μοντέλο είναι κυρίως αυτές, που τα οχήματα έχουν σχετικά μικρή χωρητικότητα, και επομένως βρίσκονται πίσω στην κεντρική αποθήκη πολύ πριν τη λήξη της βάρδιας.

Για την επίλυση του χρησιμοποιούνται τόσο ευρετικοί, όσο και μεταευρετικοί αλγόριθμοι και το πρόβλημα μελετάται εκτενώς τα τελευταία χρόνια(Brandao and Mercer 1997, Petch and Salhi 2003, Cordeau, Gendreau et al. 2005, Salhi and Petch 2007, Şen and Bülbül 2008, Cattaruzza, Absi et al. 2014).



Σχήμα 10 Παράδειγμα MTVRP

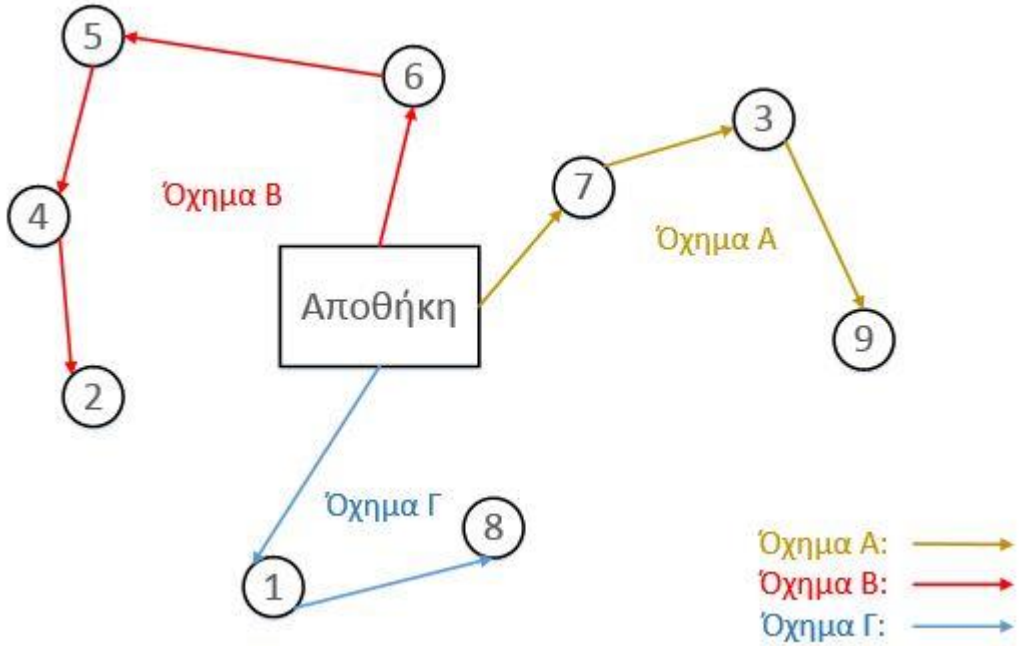
3.2.11 Ανοιχτό VRP

Το ανοιχτό VRP (Open VRP-OVRP) πρόκειται για μια εκδοχή του VRP κατά την οποία, τα οχήματα δεν επιστρέφουν μετά το πέρας του δρομολογίου πίσω στην κεντρική αποθήκη, ή στην περίπτωση που επιστρέφουν χρησιμοποιούν τη διαδρομή που ακολούθησαν κατά την εξυπηρέτηση των πελατών. Το πρόβλημα πλέον δεν σχηματίζει χαμιλτονιανή διαδρομή. Έτσι, το πρόβλημα πρακτικά ανάγεται σε πρόβλημα βέλτιστης διαδρομής, αφού οι πελάτες ανατεθούν σε συγκεκριμένα οχήματα για την εξυπηρέτηση τους(Repoussis, Tarantilis et al. 2010).

Η κύρια χρήση του OVRP αφορά εταιρείες logistics τρίτου τύπου(3d party logistics-3PL), διανομή εφημερίδων, μεταφορά μαθητών σε σχολεία, αφού τέτοιες εταιρείες χρησιμοποιούν οδηγούς υπεργολαβικά, με δικά τους οχήματα και επομένως δεν χρειάζεται να επιστρέψουν στην κεντρική αποθήκη(Wang, Wu et al. 2006).

Οι περισσότεροι μελετητές θεωρούν ότι το κόστος χρήσης επιπλέον οχημάτων είναι πολύ μεγαλύτερο του κόστους κίνησης των υπαρχόντων οχημάτων. Έτσι, πρωταρχικό στόχο βελτιστοποίησης αποτελεί ο αριθμός των απαιτούμενων οχημάτων και σε δεύτερο επίπεδο η συνολικά διανυόμενη απόσταση(Fu, Eglese et al. 2005, Fu, Eglese et al. 2006, Fleszar, Osman et al. 2009, Zachariadis and Kiranoudis 2010).

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα OVRP 3 οχημάτων και 9 πελατών.



Σχήμα 11 Παράδειγμα OVRP

3.2.12 VRP με παραδόσεις και παραλαβές (VRPPD)

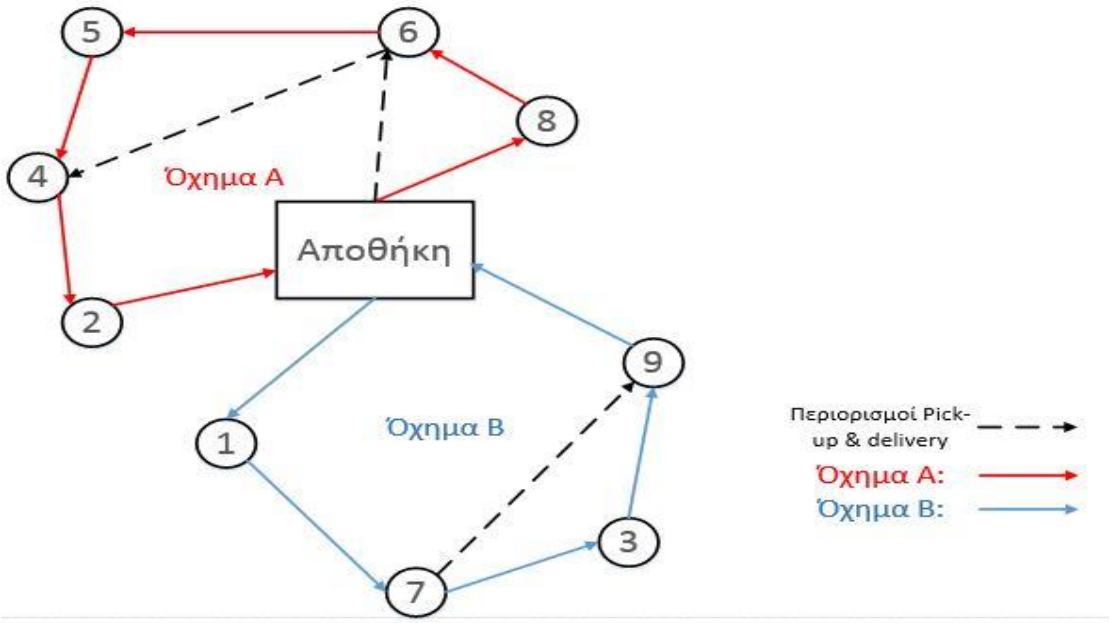
Το VRP με παραδόσεις και παραλαβές (Pick&Delivery VRP-PDVRP) είναι μια νέα ακόμη παραλλαγή του VRP που επιτρέπει την αντίστροφη ροή αγαθών σε ένα δίκτυο. Στην κλασική εκδοχή του VRP, κάθε πελάτης ζητάει να του παραδοθεί μια συγκεκριμένη ποσότητα αγαθών. Αντίθετα στο PDVRP κάθε πελάτης μπορεί να ζητήσει είτε παράδοση, είτε παραλαβή αγαθών είτε συνδυασμό τους. Έτσι, κάποια αγαθά από την κεντρική αποθήκη θα καταλήξουν στον πελάτη, και ορισμένα αγαθά από τον πελάτη θα καταλήξουν στην κεντρική αποθήκη(Battarra, Cordeau et al. 2014).

Βάσει των πιθανών συνδυασμών παραδόσεων και παραλαβών δημιουργούνται οι παρακάτω εκδοχές του PDVRP:

1. VRPSPD: VRP με ταυτόχρονες παραδόσεις και παραλαβές (VRP with Simultaneous Pickup and Delivery), στο οποίο κάθε πελάτης έχει συγκεκριμένη ποσότητα για παραλαβή και παράδοση(Subramanian, Drummond et al. 2010, Zachariadis and Kiranoudis 2011).
2. VRPMPD: VRP με μικτές παραδόσεις και παραλαβές (VRP with Mixed Pickup&Delivery), στο οποίο κάθε πελάτης έχει συγκεκριμένη ποσότητα είτε για παραλαβή είτε για παράδοση(Hong and Liu 2009).

Το PDVRP συναντάται σε δίκτυα συλλογής αγαθών, όπως για παράδειγμα στη συλλογή απορριμάτων.

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα, όπου με διακεκομμένη γραμμή αναπαριστώνται οι περιορισμοί παραλαβών και παραδόσεων, και με συνεχή γραμμή μια πιθανή λύση για τη δρομολόγηση οχημάτων.



Σχήμα 12 Παράδειγμα PDVRP

3.3 Μέθοδοι επίλυσης

Το VRP είναι ένα NP-hard πρόβλημα, ιδιαίτερα μελετημένο τα τελευταία χρόνια. Κάθε παραλλαγή του VRP, αποτελεί και αυτή πρόβλημα κατηγορίας NP-hard(Lenstra and Kan 1981). Για το λόγο αυτό, έχουν αναπτυχθεί τρεις βασικές κατηγορίες αλγορίθμων για την επίλυση τους(Cordeau, Gendreau et al. 2002):

1. **Αναλυτικές μέθοδοι**(Exact methods): Είναι αλγόριθμοι που εξετάζουν όλο το φάσμα δυνατών λύσεων και βρίσκουν την καλύτερη. Η διαδικασία είναι χρονοβόρα και με υπολογιστές τελευταίας γενιάς μπορούν να επιλυθούν προβλήματα με το πολύ 135 πελάτες.
2. **Ευρετικές μέθοδοι**(Heuristic methods): Είναι αλγόριθμοι που παρέχουν αποδεκτές λύσεις, κοντά στη βέλτιστη τιμή μέσα σε αποδεκτό χρόνο, χρησιμοποιώντας επαναληπτικές διαδικασίες και αξιολογώντας τις δημιουργηθείσες λύσεις.
3. **Μεταευρετικοί αλγόριθμοι**(Metaheuristic methods): Είναι αλγόριθμοι, που για πολλούς αποτελούν υποκατηγορία των ευρετικών μεθόδων, και χρησιμοποιούν ευρετικές μεθόδους με ειδικούς τρόπους για να οδηγηθούν σε αποδεκτό χρόνο σε καλής ποιότητας λύσεις.

   Στη συνέχεια γίνεται αναφορά σε μερικούς αλγόριθμος από κάθε κατηγορία, με μεγαλύτερη έμφαση στους ευρετικούς και μεταευρετικούς.

3.3.1 Αναλυτικές μέθοδοι

Οι αναλυτικές μέθοδοι, όπως αναφέρθηκε στην προηγούμενη ενότητα, είναι μέθοδοι που βρίσκουν πάντα τη βέλτιστη λύση, αλλά ο υπολογιστικός χρόνος που απαιτείται είναι υπερβολικά μεγάλος, με αποτέλεσμα να μην χρησιμοποιούνται στην πράξη. Οι αναλυτικές μέθοδοι που έχουν αναπτυχθεί είναι βασισμένες στη βελτιστοποίηση δικτύων, στο γραμμικό και ακέραιο προγραμματισμό. Οι κατευθύνσεις των αναλυτικών μεθόδων είναι ο δυναμικός προγραμματισμός, η ‘’χαλάρωση’’ περιορισμών(Lagrangian Relaxation) και η μέθοδος παραγωγής στηλών(Column Generation). Οι δύο τελευταίες κατευθύνσεις στηρίζονται στην αρχή της αποικοδόμησης (decomposition principle), δηλαδή το κυρίως πρόβλημα διασπάται σε πολλά μικρότερα για τη διευκόλυνση της επίλυσης του.

Η κυριότερη μέθοδος που χρησιμοποιείται για την επίλυση μεγάλης κλίμακας υπολογιστικών προβλημάτων είναι ο αλγόριθμος Branch&Bounds(B&B), ο οποίος αποτελεί και τη βάση για τους περισσότερους αναλυτικούς αλγορίθμους. Η βασική ιδέα, στην οποία στηρίζεται είναι ο χωρισμός του διαστήματος αναζήτησης(search space) σε μικρότερα διαστήματα(branching)και η αξιολόγηση του άνω και κάτω ορίου για καθένα από αυτά τα υποδιαστήματα(bounding). Αν αποδειχθεί οτι κάποιο υποδιάστημα δεν περιέχει τη βέλτιστη λύση, αυτό απορρίπτεται(pruning), ενώ σε αντίθετη περίπτωση το υποδιάστημα υφίσταται επιπρόσθετο διαχωρισμό και έλεγχο(further branching and bounding)(Fisher 1994, Miller 1995, Martinhon, Lucena et al. 2004, Baldacci, Toth et al. 2007).

Για την εκτίμηση του κάτω ορίου, χρησιμοποιείται η μέθοδος επίλυσης ενός ‘’χαλαρού’’ προβλήματος(relaxed problem), η οποία όπως ήδη αναφέρθηκε έγκειται στην αφαίρεση ενός αριθμού περιορισμών κατά τη διαμόρφωση του προβλήματος. Με αφαίρεση διαφορετικών περιορισμών, κατά τη διαμόρφωση του προβλήματος, δημιουργούνται διαφορετικά «χαλαρά» προβλήματα κάθε φορά. Τα νέα αυτά προβλήματα μπορούν να επιλυθούν πολύ γρήγορα, αλλά η ποιότητα των τιμών των κάτω ορίων που προκύπτουν, είναι πολύ κακή.

Για την βελτίωση των κάτω ορίων, χρησιμοποιείται η προαναφερθείσα μέθοδος Lagrangian Relaxation, κατά την οποία οι περιορισμοί που αφαιρούνται μετατρέπονται σε όρους που προστίθενται στην αντικειμενική συνάρτηση, συνήθως με τη μορφή ποινών. Έτσι, κάθε όρος της αντικειμενικής που προκύπτει από χαλάρωση περιορισμών πολλαπλασιάζεται με συντελεστή ποινής(penalty factor), ο οποίος αντιστοιχεί στο συντελεστή λ του Lagrange(Geoffrion 1974, Fisher 1994).

Στη μέθοδο παραγωγής στηλών(Column Generation), το VRP μοντελοποιείται μέσω δύο ξεχωριστών προβλημάτων:

I. Κύριο πρόβλημα(Master problem-MP),το οποίο αφορά το διαχωρισμό των πελατών σε υποσύνολα, καθένα από τα οποία εξυπηρετείται από ένα όχημα.
II. Δευτερεύον πρόβλημα(Sub problem-SP), το οποίο βελτιστοποιεί τη σειρά με την οποία κάθε όχημα εξυπηρετεί τους πελάτες που του έχουν ανατεθεί από το κυρίως πρόβλημα, ικανοποιώντας όμως τους περιορισμούς χωρητικότητας.

Σύμφωνα με τις παραπάνω τεχνικές, υπολογισμού των κάτω ορίων, η B&B μπορεί να επεκταθεί σε Branch&Cut, Branch&Price ή ακόμα και Branch&Cut&Price(B&C&P). Η Branch&Price αναφέρεται σε μια μέθοδο που βασίζεται στην κλασματοποίηση του προβλήματος και χρησιμοποιεί την Column Generation για την εύρεση των κάτω ορίων για κάθε «κλαδί» του αναπτυσσόμενου δέντρου. Η μέθοδος αυτή μπορεί να επιλύσει μέχρι και δίκτυα 50 πελατών(Agarwal, Mathur et al. 1989, Hadjiconstantinou, Christofides et al. 1995). Αντίθετα, στην Branch&Cut το κατώτερο όριο κάθε κλαδιού του δέντρου, βελτιώνεται επαναληπτικά, προσθέτοντας στο «χαλαρό» πρόβλημα ένα αριθμό ορθών ανισοτήτων που παραβιάζονται από τη δεδομένη λύση. Οι ανισότητες αυτές παράγονται από ευρετικές μεθόδους διαχωρισμού. Η μέθοδος αυτή, μπορεί να επιλύσει δίκτυα 135 πελατών(Letchford, Lysgaard et al. 2007, Baldacci, Mingozzi et al. 2012). Τέλος, η Branch&Cut&Price είναι ένας συνδυασμός των προηγούμενων δύο μεθόδων, όπου παράγονται καλύτερα κατώτατα όρια.

3.3.2 Ευρετικές μέθοδοι

Οι ευρετικές μέθοδοι είναι μια κατηγορία μεθόδων, οι οποίες βρίσκουν πάντα λύση στο πρόβλημα, όχι αναγκαστικά τη βέλτιστη, αλλά αρκετά κοντά, ούτως ώστε να μπορέσει να αξιοποιηθεί στην πράξη. Για το πρόβλημα δρομολόγησης οχημάτων συγκεκριμένα, οι ευρετικές μέθοδοι δίνουν λύση 5-35% χειρότερες από τη βέλτιστη λύση. Το βασικό τους πλεονέκτημα είναι ο μικρός χρόνος εκτέλεσης. Χωρίζονται σε κατασκευαστικές μεθόδους(construction heuristics) και μεθόδους βελτίωσης υπαρχόντων

λύσεων(improvement heuristics). Οι κατασκευαστικές μέθοδοι φτιάχνουν μια λύση, βάσει κάποιων κανόνων, αλλά δεν προσπαθούν να τη βελτιώσουν. Απαιτούν συνήθως ελάχιστο υπολογιστικό χρόνο, αλλά η λύση είναι συχνά κακής ποιότητας. Αντίθετα, οι μέθοδοι βελτίωσης λειτουργούν με ολοκληρωμένες υπάρχουσες λύσεις και προσπαθούν να τις βελτιώσουν χρησιμοποιώντας μια σειρά από τελεστές. Οι τελεστές(operators) είναι συνήθως πολύ απλές κινήσεις. Επειδή οι αλγόριθμοι βελτίωσης δεν μπορούν να παράγουν μια αρχική λύση και απλά βελτιώνουν τις υπάρχουσες λέγονται και μέθοδοι τοπικής αναζήτησης(local search methods) και καθοδηγούνται από την αντικειμενική συνάρτηση(Cordeau, Gendreau et al. 2002).

3.3.2.1 Κατασκευαστικές ευρετικές μέθοδοι

Οι κατασκευαστικές μέθοδοι πάνω στο πρόβλημα δρομολόγησης, δέχονται σαν είσοδο ένα σύνολο από μη δρομολογημένους πελάτες και ένα σύνολο από μη δρομολογημένα οχήματα και κατασκευάζουν μια νέα λύση που ικανοποιεί τους εκάστοτε περιορισμούς του προβλήματος. Ανάλογα με τον τρόπο κατασκευής οι μέθοδοι χωρίζονται σε:

I. Σειριακές: Οι μέθοδοι αυτές κατασκευάζουν κάθε διαδρομή ξεχωριστά (route by route approach). Ξεκινώντας από μια άδεια διαδρομή, προσθέτουν πελάτες, μέχρι να ολοκληρωθεί, δηλαδή μέχρι να παραβιαστεί κάποιος από τους περιορισμούς του προβλήματος. Συνεχίζουν με την επόμενη διαδρομή μέχρι να εισαχθούν όλοι οι πελάτες σε μια διαδρομή.
II. Παράλληλες: Οι μέθοδοι αυτές κατασκευάζουν παράλληλα διαδρομές(parallel approach). Ξεκινούν με τόσες άδειες διαδρομές, όσες και τα οχήματα και κατανέμουν όλους τους πελάτες σε μια από αυτές.

Στην επόμενη ενότητα παρουσιάζονται οι κυριότεροι αλγόριθμοι από τους κατασκευαστικούς ευρετικούς.

3.3.2.1.1 Αλγόριθμος Savings

Ο κατασκευαστικός αυτός αλγόριθμος προτάθηκε το 1964 από τους Clark&Wright και διαθέτει μια σειριακή και μια παράλληλη εκδοχή. Η παράλληλη εκδοχή δίνει πάντα καλύτερες λύσεις σε σχέση με τη σειριακή και για αυτό το λόγο η αναφορά θα περιοριστεί στην παράλληλη εκδοχή του αλγορίθμου(Bräysy and Gendreau 2005).

Ο αλγόριθμος ξεκινά δημιουργώντας τόσες διαδρομές, όσοι και οι πελάτες που συμμετέχουν στο πρόβλημα, δηλαδή θεωρείται ότι κάθε πελάτης θα εξυπηρετηθεί από ξεχωριστό όχημα. Στη συνέχεια υπολογίζεται το κέρδος Savings, δηλαδή η μείωση του κόστους που προκύπτει από τη συνένωση δύο διαδρομών με τον ακόλουθο τρόπο:

Έστω δύο πελάτες i και j που εξυπηρετούνται από διαφορετικά οχήματα σύμφωνα με τις ακόλουθες διαδρομές R_i (depot-i-depot), R_j(depot,j,depot). Η απόσταση μεταξύ δύο οποιονδήποτε σημείων x,y είναι d_{xy}. Το κόστος των αρχικών διαδρομών είναι $C_{Ri}=d_{0i}+d_{i0}$ και $C_{Rj}=d_{0j}+d_{j0}$.Το κόστος της νέας διαδρομής που θα προκύψει αν συνενωθούν τα i και j είναι

$C_{Rij}=d_{0i}+d_{ij}+ d_{j0}$. Έτσι, το κέρδος Savings για τη συνένωση της διαδρομής προκύπτει μετά από μερικές απλές πράξεις: $S_{Rij}=d_{0i}+d_{i0}- d_{ij}$.

Αφού υπολογιστούν τα κέρδη συνένωσης για κάθε δυνατό συνδυασμό ακμών, επιλέγεται η ακμή εκείνη με το μεγαλύτερο κέρδος συνένωσης, που ικανοποιεί τους εκάστοτε περιορισμούς του προβλήματος και οι δύο διαδρομές συνενώνονται. Αυτό συνεχίζεται διαδοχικά μέχρι να μην μπορεί να συνενωθεί καμία άλλη διαδρομή(Paessens 1988, Altınel and Öncan 2005).

Ο αλγόριθμος σε ψευδοκώδικα παρουσίαζεται στη συνέχεια:

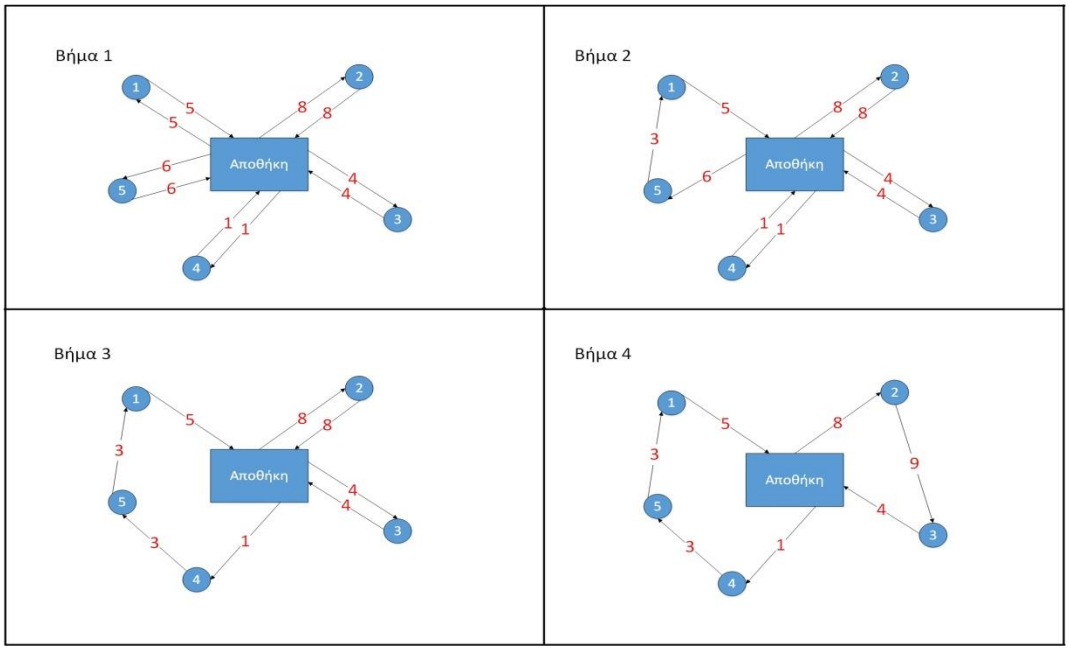
1. Για κάθε πελάτη i δημιούργησε διαδρομή Depot-i-Depot.
2. Υπολόγισε το κέρδος Savings για κάθε δυνατό ζεύγος ακμών.
3. Όσα τα κριτήρια τερματισμού δεν ισχύουν επανέλαβε:
    a. Επέλεξε τις διαδρομές με τη μεγαλύτερη μείωση κόστους, που μπορούν να συνενωθούν.
    b. Συνένωσε τις διαδρομές με τη μεγαλύτερη μείωση κόστους του προηγούμενο βήματος.
    c. Επέλεξε τις επόμενες διαδρομές για συνένωση.

Αλγόριθμος 1 Savings

Στο παρακάτω σχήμα παρουσιάζεται ένα παράδειγμα αλγορίθμου Savings με 5 πελάτες και 2 οχήματα. Το κόστος κάθε ακμής βρίσκεται πάνω από την ίδια την ακμή, η χωρητικότητα του φορτηγού είναι 80 τεμάχια και η ζήτηση των πελατών φαίνεται στο σχετικό πίνακα.

Πελάτης 1	Πελάτης 2	Πελάτης 3	Πελάτης 4	Πελάτης 5
30	35	35	20	25

Πινακας 1 Παράδειγμα Savings



Σχήμα 13 Savings Algorithm

3.3.2.1.2 Αλγόριθμος Mole and Jameson

Ο αλγόριθμος αυτός κατασκευάστηκε το 1976 από τους Mole και Jameson και βασίζεται στον Savings που περιγράφηκε στην προηγούμενη ενότητα. Ο αλγόριθμος αυτός ξεκινά από ένα μόνο κόμβο και δημιουργεί ένα δρομολόγιο με την κεντρική αποθήκη. Στη συνέχεια, προστίθενται ένας ένας κι άλλοι κόμβοι στο δρομολόγιο, βάσει συγκεκριμένων κανόνων μέχρι να μην μπορούν να προστεθούν άλλοι. Για να προστεθεί ένας κόμβος k υπολογίζεται αρχικά το κόστος εισαγωγής α και αναζητείται εκείνη η θέση στην οποία το κόστος αυτό είναι το ελάχιστο, δηλαδή εκείνη τα i και j για τα οποία το a(I,k,j) παίρνει την ελάχιστη τιμή. Η διαδικασία αυτή επαναλαμβάνεται για όλους τους κόμβους που δεν έχουν χρησιμοποιηθεί ήδη σε κάποιο δρομολόγιο. Αν υπάρχει κόμβος ή κόμβοι που μπορούν να εισαχθούν χωρίς να παραβιάσουν κάποιο κριτήριο(χωρητικότητα στη γενική περίπτωση) εξετάζεται ο καλύτερος κόμβος k*, ο οποίος μεγιστοποιεί το κέρδος β. Τέλος, χρησιμοποιείται κάποιος αλγόριθμος τοπικής αναζήτησης(συνήθως 2-opt) για τη βελτίωση της παραγόμενης λύσης(Mole and Jameson 1976, Rosenkrantz, Stearns et al. 1977, Campbell and Savelsbergh 2004).

Η διαφορά του Mole&Jameson αλγορίθμου σε σχέση με τον Savings είναι η χρησιμοποίηση όχι μόνο της μείωσης του κόστους αλλά και της εγγύτητας του νεοεισερχόμενου κόμβου. Επιπρόσθετα, το κέρδος εισαγωγής και το κόστος εισαγωγής είναι παραμετροποιημένα βάσει των παραμέτρων λ και μ, με αποτέλεσμα διαφορετικές τιμές των μεταβλητών αυτών να δημιουργούν διαφορετικού επιπέδου λύσεις.

Ο αλγόριθμος Mole&Jameson σε ψευδοκώδικα παρουσιάζεται στη συνέχεια:

1. Όσο υπάρχουν κόμβοι που δεν έχουν δρομολογηθεί επανέλαβε:
    a. Επέλεξε ένα τυχαίο κόμβο k από αυτούς που δεν έχουν δρομολογηθεί
    b. Δημιούργησε τη διαδρομή (0,k,0)
    c. Επανάλαβε για κάθε κόμβο από τους μη δρομολογημένους
        i. Βρές το κόστος εισαγωγής $a(i,k,j)=c_{ik}+c_{kj}-\lambda c_{ij}$
        ii. Βρες τα i* και j* για τα οποία το κόστος εισαγωγής γίνεται ελάχιστο
    d. Υπολόγισε το κέρδος $\beta(i^*,k,j^*)=\mu C_{0k}-\alpha(i^*,k,j^*)$
    e. Επέλεξε από τους κόμβους που δεν παραβιάζουν κάποιο κριτήριο, εκείνον με το μεγαλύτερο β
    f. Πρόσθεσε τον κόμβο που επέλεξες μεταξύ των i και j του βήματος c.i.
    g. Αν δεν μπορούν να εισαχθούν άλλοι κόμβοι στην υπάρχουσα διαδρομή, χρησιμοποίησε έναν 2-opt αλγόριθμο
    h. Επέστρεψε στο βήμα 1
2. Τέλος αλγορίθμου

Αλγόριθμος 2 Mole&Jameson

3.3.2.1.3 Αλγόριθμος Πλησιέστερου Γείτονα

Ο αλγόριθμος του πλησιέστερου γείτονα (Nearest Neighbor) είναι ένας από τους πιο απλούς ευρετικούς αλγορίθμους για το VRP. Ο αλγόριθμος λειτουργεί σειριακά. Αρχικά, ξεκινά τη δρομολόγηση, προσθέτοντας σε ένα νέο δρομολόγιο τον πελάτη εκείνο που βρίσκεται πιο κοντά στην κεντρική αποθήκη. Στη συνέχεια, προσθέτει επαναληπτικά στην ίδια διαδρομή τον πελάτη που βρίσκεται πιο κοντά στον τελευταίο πελάτη που προστέθηκε, μέχρι να μη μπορεί να εισαχθεί άλλος πελάτης στην παρούσα διαδρομή. Η διαδικασία αυτή επαναλαμβάνεται όσες φορές απαιτείται για να δρομολογηθούν όλοι οι πελάτες. Απαραίτητη προϋπόθεση για την εισαγωγή κάποιου πελάτη σε ένα δρομολόγιο είναι η μη υπέρβαση της χωρητικότητας του φορτηγού από την επιπλέον ζήτηση του τελευταίου πελάτη που δύναται να προστεθεί. Κάθε φορά που μια διαδρομή ολοκληρώνεται ο αλγόριθμος ξαναρχίζει από την αρχή και κατασκευάζει την επόμενη διαδρομή, χρησιμοποιώντας μόνο όσους πελάτες δεν έχουν ήδη δρομολογηθεί(Rosenkrantz, Stearns et al. 1977, Solomon 1987).

Ο αλγόριθμος του πλησιέστερου γείτονα χρησιμοποιείται με μικρές διαφορές για όλες τις παραλλαγές του VRP, αλλά η ποιότητα των λύσεων που παράγονται συχνά δεν είναι καλή. Αυτό οφείλεται κυρίως στη φύση του αλγορίθμου, αφού δύναται να μην εντοπίσει μικρότερα δρομολόγια, τα οποία όμως είναι εύκολα αναγνωρίσιμα από τον άνθρωπο. Για την επαλήθευση της ποιότητας της παραγόμενης λύσης χρησιμοποιούνται δύο τρόποι. Ο πρώτος τρόπος ελέγχει το μήκος των πρώτων δρομολογίων και το συγκρίνει με το μήκος των τελευταίων δρομολογίων που παράγονται. Αν το μήκος των δύο κατηγοριών είναι συγκρίσιμο η λύση είναι καλής ποιότητας. Ο δεύτερος τρόπος είναι ο υπολογισμός κατώτατου ορίου(lower bound) για το πρόβλημα όπως περιγράφηκε στις αναλυτικές μεθόδους και σύγκριση με την παραγόμενη λύση(Bräysy and Gendreau 2005).

Ακολουθεί ο αλγόριθμος Nearest Neighbor σε ψευδοκώδικα:

1. Όσο υπάρχουν πελάτες που δεν έχουν δρομολογηθεί επανέλαβε:
    a. Επέλεξε τον πιο κοντινό στην κεντρική αποθήκη, πελάτη k που δεν έχει δρομολογηθεί και η χωρητικότητα του δεν υπερβαίνει την χωρητικότητα του οχήματος.
    b. Δημιούργησε νέα διαδρομή (0,k,0)
    c. Όσο υπάρχουν πελάτες που δεν έχουν δρομολογηθεί και η χωρητικότητα του οχήματος είναι μεγαλύτερη από τη ζήτηση τους επανέλαβε:
        i. Επέλεξε τον πιο κοντινό από αυτούς τους πελάτες (m), στον τελευταίο κόμβο που προστέθηκε.
        ii. Πρόσθεσε το νέο πελάτη στο τέλος της διαδρομής, πριν την κεντρική αποθήκη και δημιούργησε διαδρομή (0,….,k,m,0).
        iii. Πήγαινε στο βήμα c.
    d. Ολοκλήρωση τρέχουσας διαδρομής.
    e. Πήγαινε στο βήμα 1.
2. Τέλος αλγορίθμου.

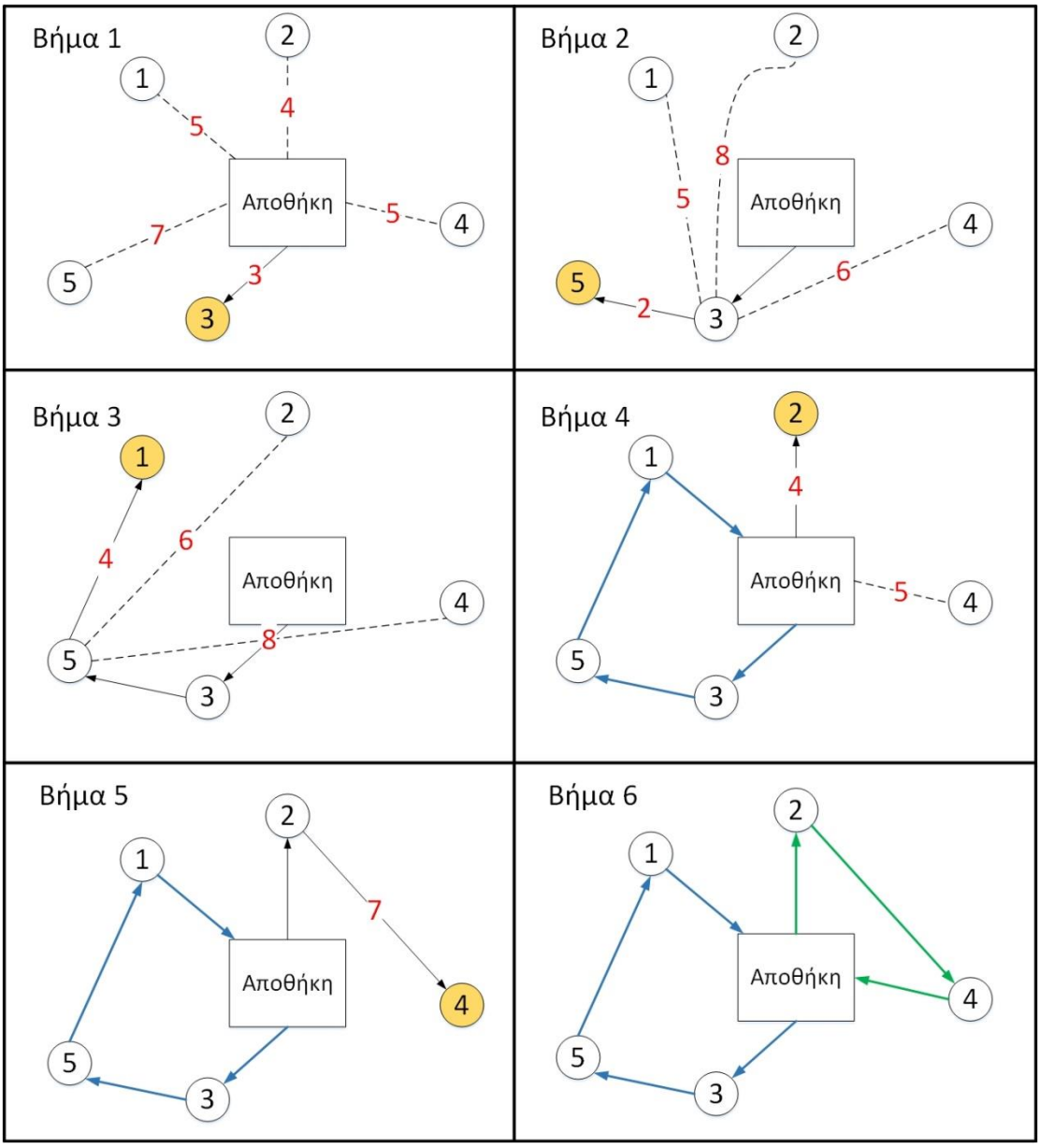
Αλγόριθμος 3 Nearest Neighbor

Στο παρακάτω σχήμα παρουσιάζεται ένα παράδειγμα του αλγορίθμου Nearest Neighbor για 5 πελάτες και 2 οχήματα. Η απόσταση κάθε πελάτη από τον τελευταίο που προστέθηκε ή από την κεντρική αποθήκη φαίνεται πάνω στην ακμή που τους συνδέει. Η χωρητικότητα κάθε οχήματος είναι 100 τεμάχια και η ζήτηση κάθε πελάτη φαίνεται στον παρακάτω πίνακα.

1	2	3	4	5
25	15	40	60	30

Πινακας 2 Παράδειγμα Nearest Neighbor

Όπως φαίνεται στο παρακάτω σχήμα, ο αλγόριθμος αρχικά επιλέγει τον πελάτη 3 και μετά διαδοχικά τους πελάτες 5 και 1. Επειδή η εναπομένουσα χωρητικότητα του οχήματος δεν επαρκεί για να προστεθεί κάποιος άλλος πελάτης, η διαδρομή τελειώνει και δημιουργείται νέο δρομολόγιο για τους υπόλοιπους πελάτες με τον ίδιο ακριβώς τρόπο.



Σχήμα 14 Παράδειγμα Nearest Neighbor

3.3.2.1.4 Αλγόριθμος Sweep

Ο αλγόριθμος Sweep αναπτύχθηκε το 1974 από τους Gillet και Miller. Αρχικά, ο αλγόριθμος θέτει μια αρχή, συνήθως την κεντρική αποθήκη και υπολογίζει την πολική γωνία όλων των πελατών ως προς αυτή. Στη συνέχεια προσθέτει πελάτες σαρώνοντας ωρολογιακά ή αντιωρολογιακά, μέχρι να μη μπορεί να προστεθεί άλλος πελάτης λόγω υπέρβασης της χωρητικότητας του οχήματος. Στην περίπτωση που δεν χωράει κάποιος πελάτης στο όχημα η διαδικασία ξεκινάει από την αρχή, για νέο δρομολόγιο. Η διαδικασία επαναλαμβάνεται μέχρι όλοι οι πελάτες να μπουν σε δρομολόγιο. Ακολουθεί ο αλγόριθμος σε ψευδοκώδικα:

1. Όσο υπάρχουν πελάτες που δεν έχουν δρομολογηθεί επανέλαβε:
    a. Επέλεξε κόμβο ως προς τον οποίο θα γίνουν οι υπολογισμοί.
    b. Υπολόγισε την πολική γωνία για κάθε πελάτη ως προς αυτό τον κόμβο.
    c. Όσο η εναπομείνουσα χωρητικότητα του οχήματος είναι μεγαλύτερη από τη ζήτηση του πελάτη με τη μικρότερη γωνία που δεν έχει δρομολογηθεί επανέλαβε:
        i. Πρόσθεσε τον πελάτη στη διαδρομή.
        ii. Πήγαινε στο βήμα c.
    d. Πήγαινε στο βήμα 1.
2. Τέλος Αλγορίθμου

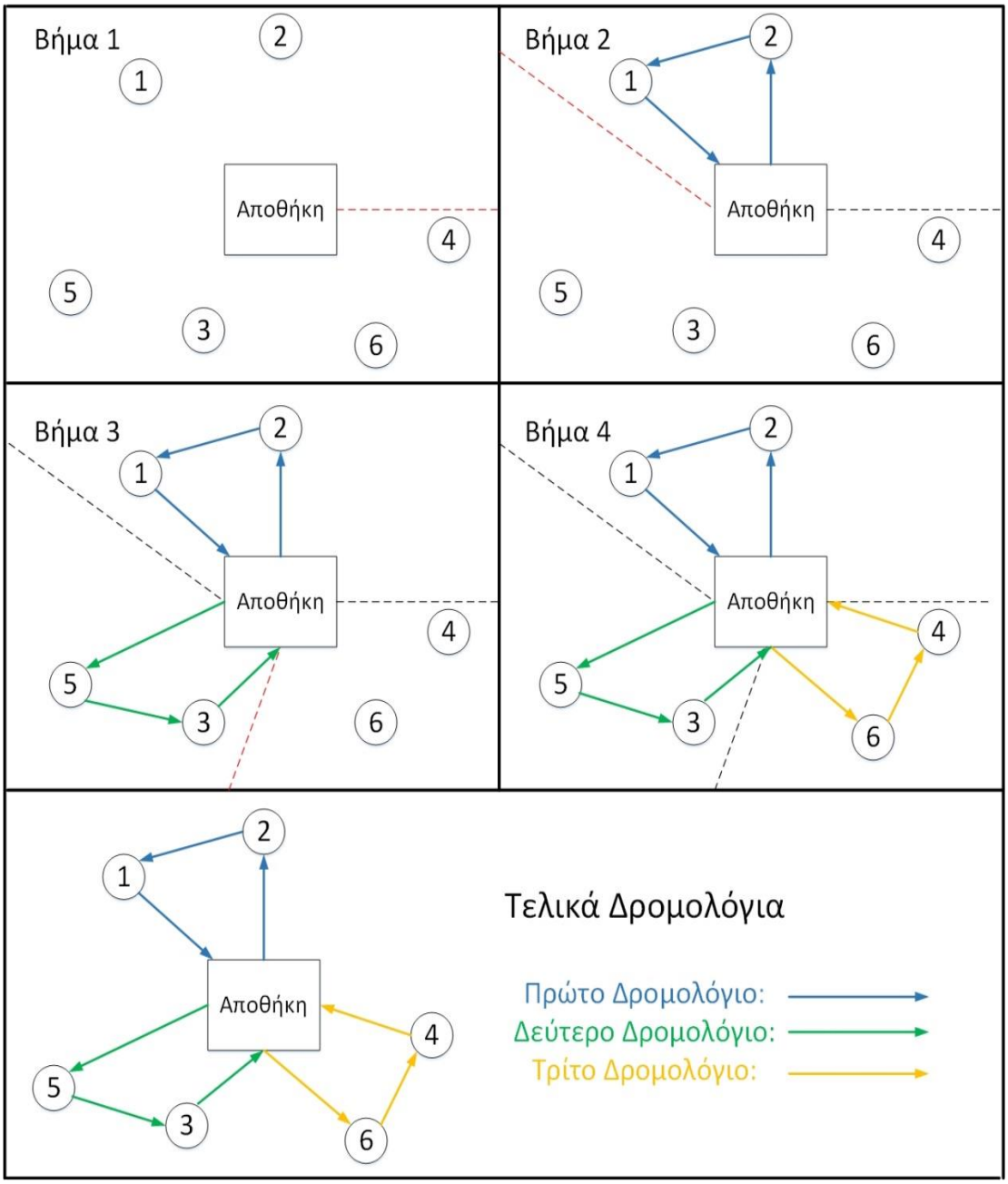
Αλγόριθμος 4 Sweep

Ο αλγόριθμος αυτός έχει πολλά μειονεκτήματα σε σχέση με τους υπόλοιπους αλγορίθμους που περιγράφηκαν. Αρχικά, ο υπολογισμός της πολικής γωνίας προϋποθέτει ευκλείδειο σύστημα συντεταγμένων, γεγονός που δύσκολα συναντάται σε πραγματικά σενάρια. Επιπρόσθετα, ο αλγόριθμος δεν δημιουργεί δρομολόγια καθώς επεξεργάζεται τους κόμβους, με αποτέλεσμα η διαδικασία να γίνεται χρονοβόρα. Τέλος, ο αλγόριθμος δεν λαμβάνει υπόψη του τα μήκη των διαδρομών και μπορεί να δημιουργήσει ανέφικτες λύσεις αν παρουσιαστεί κάποιος περιορισμός μέγιστης διανυόμενης απόστασης για κάθε όχημα. Πάραυτα, είναι ένας γρήγορα υλοποιήσιμος αλγόριθμος, και μπορεί με κατάλληλες τεχνικές να βρει μια ικανοποιητική λύση για το πρόβλημα(Solomon 1987, Laporte, Gendreau et al. 2000, Renaud and Boctor 2002).

Στο παρακάτω σχήμα παρουσιάζεται ένα παράδειγμα αλγορίθμου Sweep. Ως κόμβος αφετηρίας για τους υπολογισμούς χρησιμοποιείται η κεντρική αποθήκη. Στο πρώτο βήμα ο αλγόριθμος υπολογίζει την πολική γωνία κάθε πελάτη ως προς την αποθήκη. Ο αλγόριθμος επιλέχθηκε να κινείται αντιωρολογιακά. Η χωρητικότητα του φορτηγού είναι 80 τεμάχια και η ζήτηση των πελατών φαίνεται στο παρακάτω πίνακα:

1	2	3	4	5	6
45	42	50	45	38	45

Πινακας 3 Παράδειγμα Sweep



Σχήμα 15 Παράδειγμα Sweep

3.3.2.2 Μέθοδοι Τοπικής Αναζήτησης

Οι μέθοδοι τοπικής αναζήτησης, όπως έχει περιγραφεί σε προηγούμενη ενότητα, είναι μέθοδοι βελτίωσης υπαρχόντων λύσεων και δεν μπορούν να λειτουργήσουν χωρίς μια αρχική λύση. Χρησιμοποιούν μια σειρά από τελεστές, τους οποίους μπορούν να εφαρμόσουν σε ένα μόνο δρομολόγιο ή και σε περισσότερα του ενός. Ο πιο συχνός διαχωρισμός τους είναι:

1. Intra Route: Οι μέθοδοι τοπικής αναζήτησης ερευνούν γειτονικές λύσεις της αρχικής, αλλά οι τελεστές που πραγματοποιούνται, δεν επηρεάζουν διαφορετικά

μεταξύ τους δρομολόγια. Με άλλα λόγια, επηρεάζεται μόνο η σειρά εξυπηρέτησης των πελατών και δεν ανταλλάσσονται πελάτες μεταξύ των οχημάτων.
2. Inter Route: Οι μέθοδοι τοπικής αναζήτησης μπορούν να επηρεάσουν τόσο τη σειρά εξυπηρέτησης των πελατών, όσο και το όχημα που τους εξυπηρετεί.

Η τοπικά βέλτιστη λύση που παράγεται από κάποια μέθοδο τοπικής αναζήτησης δύναται να διαφέρει σημαντικά από την ολική βέλτιστη λύση του εκάστοτε προβλήματος, αφού οι μέθοδοι αυτές πραγματοποιούν αλλαγές εφόσον οι αλλαγές αυτές προκαλούν βελτίωση στη λύση. Έτσι, η ποιότητα των παραγόμενων λύσεων για όλες τις μεθόδους τοπικής αναζήτησης εξαρτάται από την ποιότητα της αρχικής λύσης αλλά και από τον τρόπο που παράγονται νέες λύσεις(Bräysy and Gendreau 2005).

Οι μέθοδοι inter-route μπορούν να διαχωριστούν σε τέσσερις κατηγορίες:

1. String cross: η μέθοδος αυτή ανταλλάσσει δύο αλυσίδες που ενώνουν κόμβους αν αυτές διασταυρώνονται μεταξύ τους.
2. String exchange: η μέθοδος αυτή ανταλλάσσει δύο αλυσίδες μεταξύ τους.
3. String relocation: η μέθοδος αυτή μεταφέρει 2 αλυσίδες που ενώνουν κόμβους σε διαφορετικό δρομολόγιο.
4. String mix: η μέθοδος αυτή συνδυάζει τις δύο προηγούμενες.

Από τις παραπάνω μεθόδους, η πιο συχνά χρησιμοποιούμενη είναι η string relocation. Λόγω της απλότητας, του μικρού υπολογιστικού χρόνου που απαιτεί αλλά και λόγω της ισχύος της, θεωρείται το θεμελιώδες συστατικό των περισσότερων τελεστών (Van Breedam 1994).

3.3.2.2.1 Μέθοδος 2-opt

Η μέθοδος 2-opt είναι μια μέθοδος τοπικής αναζήτησης και ανήκει με βάση την κατηγοριοποίηση της προηγούμενης ενότητας, στις Intra-route μεθόδους. Δέχεται ως είσοδο, ένα δρομολόγιο μιας λύσης και επηρεάζει τη σειρά με την οποία εξυπηρετούνται οι πελάτες, προσπαθώντας να βελτιώσει την παραγόμενη λύση. Η μέθοδος δεν αλλάζει το σύνολο των πελατών που εξυπηρετούνται, αλλά μόνο τη σειρά τους και για αυτό χρησιμοποιείται σε κάθε δρομολόγιο της τελικής λύσης.

Συγκεκριμένα, η μέθοδος 2-opt λειτουργεί επαναληπτικά, εναλλάσσοντας σε κάθε επανάληψη έναν πελάτη με το γειτονικό του ως προς τη σειρά προτεραιότητας.

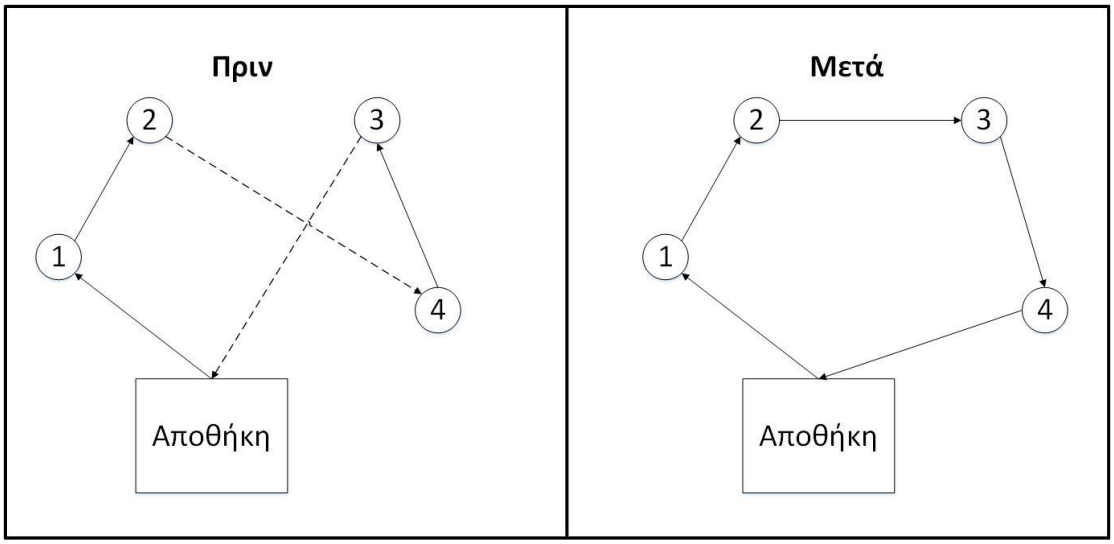
Ο αλγόριθμος 2-opt γενικά μπορεί να επεκταθεί σε λ-opt, όπου αντί να εναλλάσσονται προτεραιότητες δύο κόμβων μεταξύ τους, εναλλάσσονται κυκλικά λ προτεραιότητες. Μια από τις πιο συχνές εφαρμογές του λ-opt αλγορίθμου είναι ο 3-opt. Παρ'όλα αυτά, ο 3-opt τις περισσότερες φορές αντικαθίσταται από τον πολύ απλούστερο 2-opt αλγόριθμο μιας και οι παραγόμενες λύσεις για διαδρομές 100 πελάτων είναι μόνο 2% καλύτερες σε σχέση με τον 2-opt αλλά απαιτούν πολύ περισσότερο υπολογιστικό χρόνο(Lin 1965, Gendreau, Hertz et al. 1992, Thompson and Psaraftis 1993, Lin, Lee et al. 2009, Elhamifar and Vidal 2013).

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Όσο υπάρχει βελτίωση επανέλαβε:
    a. Θέσε ως καλύτερη απόσταση την απόσταση της υπάρχουσας διαδρομής.
    b. Επανέλαβε από(i=0,i<αριθμό των κόμβων-1):
        i. Επανάλαβε από (k=i+1,k<αριθμό των κόμβων-1)
            - Δημιούργησε νέα διαδρομή εναλλάσοντας τα i και k
            - Αν η απόσταση της νέας διαδρομής είναι μικρότερη της καλύτερης απόστασης, θέσε ως καλύτερη απόσταση τη νέα διαδρομή.
            - Πήγαινε στο βήμα i
        ii. Πήγαινε στο βήμα b.
    c. Πήγαινε στο βήμα 1.
2. Τέλος αλγορίθμου.

Αλγόριθμος 5 2-opt

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα 2-opt αλγορίθμου, το οποίο εναλλάσσει την προτεραιότητα των κόμβων 4 και 3 για να μειώσει τη συνολική απόσταση



Σχήμα 16 Παράδειγμα 2-opt

3.3.3 Μεταευρετικές μέθοδοι

Οι μεταευρετικές μέθοδοι, ή γενικές ευρετικές μέθοδοι γνωρίζουν μεγάλη ανάπτυξη τις τελευταίες δύο δεκαετίες. Πρόκειται για μια νέα προσέγγιση στην επίλυση δύσκολων υπολογιστικών προβλημάτων. Το κυριότερο πλεονέκτημα τους σε σχέση με τους συμβατικούς ευρετικούς αλγορίθμους είναι η δυνατότητα τους να αντιμετωπίζουν οποιοδήποτε σχεδόν πρόβλημα σαν ‘’μαύρο κουτί’’ και να το επιλύουν, ανεξάρτητα από το είδος του προβλήματος. Αυτό δεν συνεπάγεται σε καμία περίπτωση οτι δεν απαιτούν σωστή ρύθμιση για τη επίλυση διαφορετικών προβλημάτων(fine tuning).

Οι κυριότεροι μεταευρετικοί αλγόριθμοι που υλοποιούνται σε ένα πρόβλημα VRP περιλαμβάνουν τον αλγόριθμο αναζήτησης Ταμπού(Tabu search), τον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων(Particle Swarm Optimization), τον αλγόριθμο βελτιστοποίησης Αποικίας μυρμηγκιών(Ant Colony Optimization), το γενετικό αλγόριθμο(Genetic Algorithm), καθώς και τον αλγόριθμο προσομοίωσης ανόπτησης(simulated annealing). Η επιτυχία της εκάστοτε μεθόδου εξαρτάται από ποικίλους παράγοντες, όπως η δυνατότητα προσαρμογής του προβλήματος, η ικανότητα να συμπεριλάβουν συγκεκριμένους περιορισμούς, ο χρόνος εκτέλεσης, η ευκολία υλοποίησης τους και η επιτυχής παραμετροποίηση. Καθένας από τους παραπάνω αλγόριθμους θα περιγραφεί εν συντομία στη συνέχεια του κεφαλαίου, εκτός από τον γενετικό αλγόριθμο, ο οποίος αναλύεται στο επόμενο κεφάλαιο λόγω της χρήσης του στην παρούσα διπλωματική εργασία.

Οι μεταευρετικές μέθοδοι βελτιστοποίησης ξεκινούν συνήθως από ένα σύνολο εφικτών λύσεων και με κατάλληλες μεθόδους και χειρισμούς προσπαθούν να τις βελτιώσουν. Οι μέθοδοι είναι επαναληπτικές και συνεχίζονται μέχρι να βρεθεί μια πολύ καλής ποιότητας λύση ή να παρέλθει συγκεκριμένος αριθμός επαναλήψεων. Η κύρια διαφορά τους σε σχέση με τους απλούς ευρετικούς αλγορίθμους είναι η στοχαστικότητα. Ειδικότερα, οι ευρετικοί αλγόριθμοι, στις περισσότερες περιπτώσεις, βρίσκουν ένα τοπικό βέλτιστο, ενώ οι μεταευρετικοί προσπαθούν να εντοπίσουν το ολικό βέλτιστο.

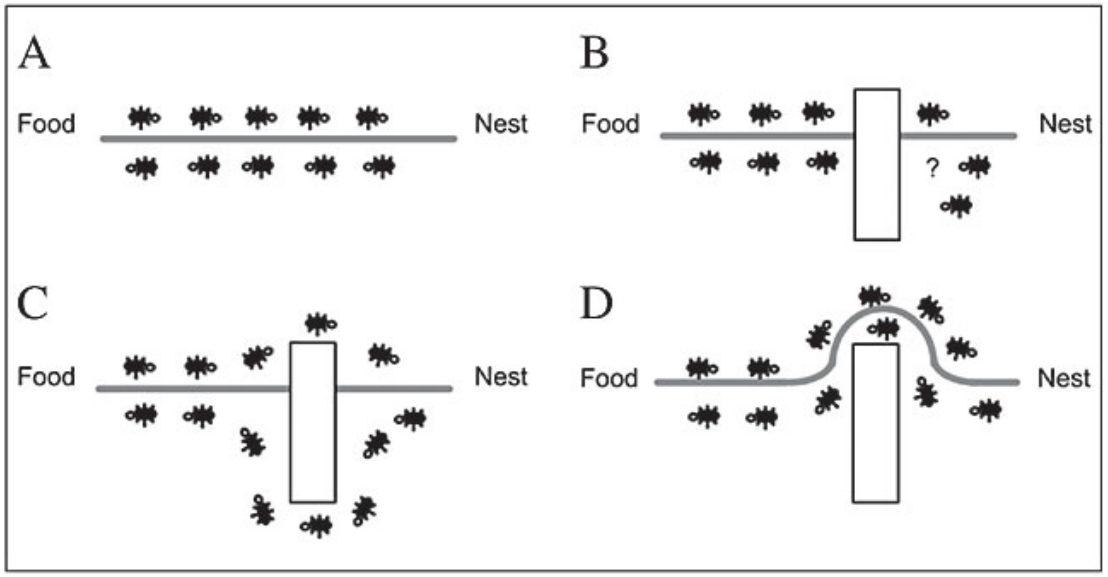
Οι μεταευρετικοί αλγόριθμοι διακρίνονται από δύο κύρια χαρακτηριστικά, την επίταση(intensification) και τη διαφοροποίηση(diversification) ή την εκμετάλλευση και την εξερεύνηση. Η διαφοροποίηση ή εξερεύνηση είναι η δυνατότητα που έχουν να ερευνούν μεγαλύτερο φάσμα λύσεων σε σχέση με τους ευρετικούς, ακόμα και εκτός του πεδίου των δυνατών λύσεων με σκοπό να βρουν τελικά το ολικό βέλτιστο. Αντίθετα, η επίταση ή εκμετάλλευση είναι η δυνατότητα τοπικής αναζήτησης για το ολικό βέλτιστο σε μια περιοχή, εαν είναι γνωστό ότι υπάρχει μια καλής ποιότητας λύση στην περιοχή αυτή(Dorigo and Stützle 2003, Yang 2010).

Οι μεταευρετικές μέθοδοι χρησιμοποιούνται ευρύτατα σε πληθώρα εφαρμογών, κυρίως για σύνθετα υπολογιστικά προβλήματα και η ανάπτυξη τους είναι ραγδαία. Η παρούσα διπλωματική ασχολείται με την υλοποίηση ενός συνδυασμού μεταευρετικών αλγορίθμων, προσπαθώντας να κατασκευάσει ένα νέο υβριδικό μεταευρετικό αλγόριθμο για την επίλυση του προβλήματος που θα παρουσιαστεί στη συνέχεια.

3.3.3.1 Βελτιστοποίηση αποικίας μυρμηγκιών(ACO)

Από το 1990 ξεκίνησε η ιδέα μίμησης της συμπεριφοράς των μυρμηγκιών για την εύρεση λύσεων σε προβλήματα βελτιστοποίησης. Η μέθοδος αυτή χρησιμοποιήθηκε ευρύτατα στο πρόβλημα του περιοδεύοντος πωλητή. Η προσομοίωση προήλθε από τον τρόπο αναζήτησης της τροφής τους, αλλά και από τον τρόπο επιστροφής στη φωλιά τους. Συγκεκριμένα, τα μυρμήγκια αρχικά εξερευνούν μια μεγάλη περιοχή γύρω από τη φωλιά τους, πραγματοποιώντας τυχαίες διαδρομές. Όταν κάποιο μυρμήγκι καταφέρει και εντοπίσει κάποια πηγή φαγητού(ανεξαρτήτως μεγέθους) την αξιολογεί(ποιότητα και ποσότητα), παίρνει ένα μέρος της και το μεταφέρει πίσω στη φωλιά. Στο ταξίδι της επιστροφής, το μυρμήγκι αφήνει πίσω του μια χημική ουσία γνωστή ως φερομόνη(pheromone). Η πυκνότητα της φερομόνης, εξαρτάται από την αξιολόγηση της ποιότητας και της ποσότητας της τροφής που εντόπισε. Ο ρόλος της φερομόνης είναι να καθοδηγήσει τα υπόλοιπα μυρμήγκια στην πηγή της τροφής. Έτσι, όσο καλύτερη είναι η ποιότητα/ποσότητα της τροφής που βρέθηκε, τόσο περισσότερη φερομόνη θα συσσωρευτεί στο μονοπάτι, με αποτέλεσμα να ακολουθήσουν ολοένα και περισσότερα μυρμήγκια. Ταυτόχρονα, οι πηγές τροφής που βρίσκονται πιο κοντά στη φωλιά, θα έχουν περισσότερες επισκέψεις μυρμηγκιών με αποτέλεσμα οι φερομόνες να αυξάνονται γρηγορότερα. Το τελικό αποτέλεσμα αυτής της διαδικασίας είναι η σταδιακή βελτίωση του απαιτούμενου χρόνου μεταφοράς της τροφής στη φωλιά.

Στο σχήμα που ακολουθεί φαίνεται μια βελτιστοποίηση της διανυόμενης απόστασης από τα μυρμήγκια.



Σχήμα 17 Παράδειγμα ACO

Η τεχνική αναζήτησης τροφής μπορεί να μετατραπεί σε αλγόριθμο με λίγες απλές αναλογίες. Έτσι:

1. Η περιοχή αναζήτησης τροφής των μυρμηγκιών μετατρέπεται σε περιοχή αναζήτησης λύσεων για κάποιο πρόβλημα βελτιστοποίησης.
2. Η ποιότητα/ποσότητα της πηγής της τροφής μετατρέπεται σε αντικειμενική συνάρτηση του προβλήματος.
3. Η φερομόνη μετατρέπεται σε χρήση μνήμης κατά την επίλυση του προβλήματος.

Ακολουθεί ο ψευδοκώδικας βελτιστοποίησης αποικίας μυρμηγκιών:

1. Αρχικοποίησε το ίχνος της φερομόνης.
2. Όσο τα κριτήρια τερματισμού δεν ισχύουν επανέλαβε:
    a. Κατασκεύασε μια λύση για κάθε μυρμήγκι με βάση τα ίχνη της φερομόνης.
    b. Πραγματοποίησε μια τοπική έρευνα για κάθε μυρμήγκι.
    c. Αξιολόγησε τη παραγόμενη λύση.
    d. Ανανέωσε τη φερομόνη βάσει της αξιολόγησης.
    e. Πήγαινε στο βήμα 2.
3. Τερματισμός αλγορίθμου.

Αλγόριθμος 6 ACO

Το σημαντικότερο στοιχείο ενός αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών είναι η σωστή διαχείριση του ίχνους της φερομόνης. Ο αλγόριθμος ACO στην απλή του μορφή, συνδυάζει τα ίχνη φερομόνης με την αντικειμενική συνάρτηση κατά την κατασκευή νέων λύσεων για κάθε μυρμήγκι. Εν συνεχεία, προσπαθώντας να προσομοιωθεί όσο καλύτερα γίνεται η διαδικασία εξάτμισης της φερομόνης, όλα τα ίχνη της μειώνονται σύμφωνα με ένα συντελεστή εξάτμισης. Τα ίχνη φερομόνης, τα οποία αντιστοιχούν στις νέες λύσεις που κατασκευάζονται από τα μυρμήγκια αυξάνουν αναλογικά με την ποιότητα της καινούριας λύσης. Βασισμένες πάνω σε αυτές τις θεμελιώδεις αρχές, έχουν προταθεί πολλές παραλλαγές του αλγορίθμου προσομοίωσης μυρμηγκιών(Baowen, Shen-min et al. 2006, Dong and Xiang 2006, Xiao and Jiang-qing 2012).

Ο αλγόριθμος ACO, όπως και οι άλλοι μεταευρετικοί αλγόριθμοι, χρησιμοποιεί την έννοια της στοχαστικότητας, με αποτέλεσμα κάθε μυρμήγκι να ψάχνει ένα ευρύτερο χώρο από πιθανές λύσεις, συγκριτικά με τους απλούς ευρετικούς που περιορίζονται σε συγκεκριμένους χώρους. Ταυτόχρονα, ο αλγόριθμος μπορεί να βοηθηθεί χρησιμοποιώντας στην αρχικοποίηση των μυρμηγκιών λύσεις που προέρχονται από ευρετικούς αλγόριθμους, σε μια προσπάθεια, τα μυρμήγκια να κατευθυνθούν σε πολλά υποσχόμενες λύσεις. Επιπρόσθετα, η αναζήτηση των μυρμηγκιών στις αρχικές επαναλήψεις μπορεί να επηρρεάσει την εκμάθηση των μυρμηγκιών στις επόμενες γενιές, διατηρώντας ένα είδος μνήμης(Dorigo and Stützle 2010).

Ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών μπορεί να χρησιμοποιηθεί σε πληθώρα εφαρμογών. Από θεωρητικής άποψης, ο αλγόριθμος μπορεί να εφαρμοστεί σε κάθε διακριτό πρόβλημα βελτιστοποίησης, για το οποίο όμως μπορεί να εφευρεθεί κάποιος κατασκευαστικός μηχανισμός(Dorigo and Stützle 2010).

Ακολουθεί το διάγραμμα ροής του ACO:

Σχήμα 18 Διάγραμμα Ροής ACO

3.3.3.2 Αναζήτηση Ταμπού(TS)

Η αναζήτηση Ταμπού(Tabu search) ή προσαρμοστικός προγραμματισμός μνήμης (adaptive memory programming) είναι μια μέθοδος επίλυσης περίπλοκων προβλημάτων βελτιστοποίησης και ανήκει στην κατηγορία των μεταευρετικών αλγορίθμων. Στόχος της είναι η αναγνώριση των καλύτερων αποφάσεων ή κινήσεων, προκειμένου να βελτιστοποιηθεί μια αντικειμενική συνάρτηση.

Πολλές πρακτικές εφαρμογές βελιστοποίησης επιλύονται με χρήση της αναζήτησης ταμπού στα πεδία των οικονομικών, των επιχειρήσεων και των επιστημών. Η πολυπλοκότητα και η σημασία των προβλημάτων που μπορεί να επιλύσει η μέθοδος αυτή, οδήγησαν σε συστηματική έρευνα τις τελευταίες δεκαετίες με σκοπό τη βελτιστοποίηση της μεθόδου, ως προς την ποιότητα των παραγόμενων λύσεων καθώς και τη μείωση του απαιτούμενου υπολογιστικού χρόνου.

Η αναζήτηση Ταμπού είναι μία από τις κυριαρχούσες τεχνολογίες για τη διαχείριση των προβλημάτων βελτιστοποίησης, τα οποία είναι δύσκολο ή ακόμα και αδύνατο να λυθούν με συμβατικές μεθόδους. Ένα ακόμα χαρακτηριστικό της μεθόδου αυτής, το οποίο επισκιάζεται συχνά από τον προσαρμοστικό προγραμματισμό μνήμης είναι η χρήση τεχνικών, σχεδιασμένων να εκμεταλλευτούν την προσαρμοζόμενη μνήμη. Η βασική ιδέα των τεχνικών αυτών είναι η δυνατότητα αποτελεσματικής αναζήτησης βέλτιστων λύσεων, μέσω μιας διαδικασίας εκμάθησης(Glover and Laguna 2013).

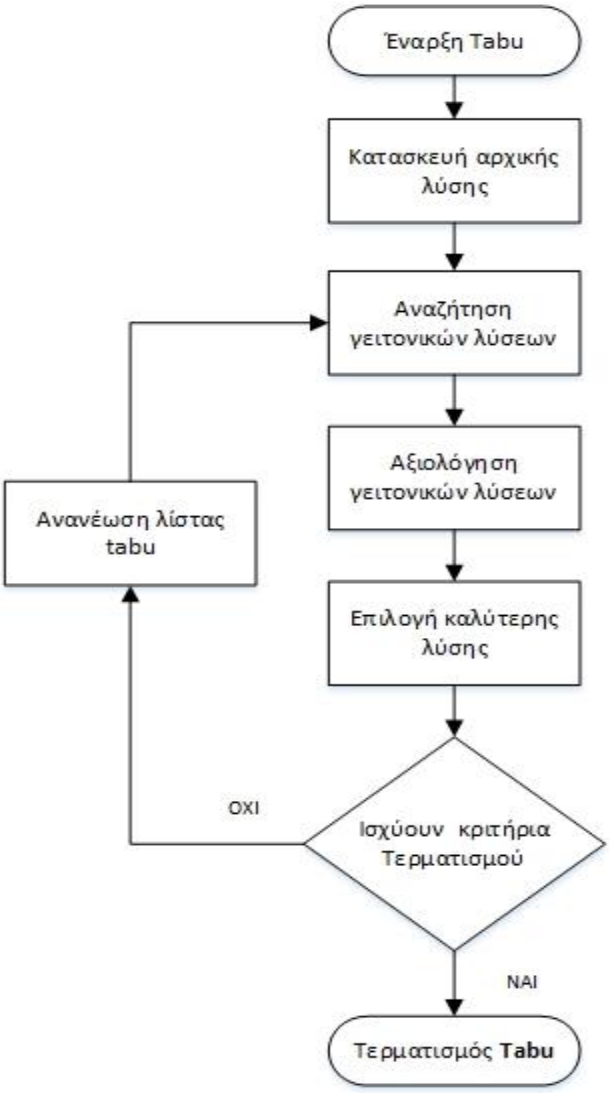
Η βασική ιδέα της μεθόδου είναι να τροποποιεί τοπικά και επαναλαμβανόμενα τη λύση, ενώ ταυτόχρονα αποθηκεύει σε μνήμη τις μετατροπές που έγιναν, ούτως ώστε να αποφευχθεί η επιστροφή σε μια λύση που έχει ήδη εξετασθεί. Συγκεκριμένα, οι μετατροπές που δέχεται η λύση διατηρούνται σε μια λίστα ταμπού, δηλαδή μια λίστα απαγορευμένων μετατροπών και ο αλγόριθμος απαγορεύει τη χρήση τους για ένα συγκεκριμένο αριθμό επαναλήψεων.Για παράδειγμα, στην περίπτωση του VRP, αν ένας πελάτης μετακινηθεί από ένα δρομολόγιο σε ένα άλλο, η ανάποδη κίνηση, η μετακίνηση δηλαδή του ίδιου πελάτη στο αρχικό δρομολόγιο, μπορεί να καταγραφεί σαν κίνηση ταμπού και να μπει στη λίστα, αποτρέποντας τη χρήση της για συγκεκριμένο αριθμό επαναλήψεων. Έτσι, υλοποιείται η βασική ιδέα της βραχυπρόθεσμης και προσαρμοζόμενης μνήμης που περιγράφηκε νωρίτερα(Glover 1989, Cordeau, Laporte et al. 2001).

Το κριτήριο τερματισμού της μεθόδου, αντιστοιχεί συνήθως σε ένα συγκεκριμένο αριθμό επαναλήψεων, ή πιο συχνά σε ένα αριθμό επαναλήψεων, κατά τον οποίο η καλύτερη λύση που έχει βρεθεί, δεν βελτιώθηκε. Κάθε βήμα του αλγορίθμου μπορεί να είναι από πολύ απλό μέχρι και πολύ σύνθετο, ανάλογα πάντα με την υλοποίηση του(Rochat and Taillard 1995).

Ακολουθεί ο ψευδοκώδικας της αναζήτησης ταμπού:

1. Κατασκεύασε μια αρχική λύση
2. Όσο τα κριτήρια τερματισμού δεν ισχύουν επανέλαβε:
    a. Βρες μια σειρά από γειτονικές λύσεις χωρίς να κάνεις κίνηση που βρίσκεται στη λίστα ταμπού.
    b. Αξιολόγησε τις γειτονικές λύσεις.
    c. Επέλεξε την καλύτερη από αυτές και θέσε την ως αρχική λύση.
    d. Ανανέωσε τη λίστα ταμπού.
    e. Πήγαινε στο βήμα 2.
3. Τέλος αλγορίθμου

Αλγόριθμος 7 Tabu search



Σχήμα 19 Διάγραμμα Ροής Tabu Search

3.3.3.3 Προσομοιωμένη Ανόπτηση(SA)

Ο αλγόριθμος προσομοίωσης ανόπτησης(Simulated Annealing) είναι εμπνευσμένος από την φυσική ανόπτηση των στερεών σωμάτων και χρησιμοποιείται για την επίλυση του VRP από τότε που πρωτάθηκε ο αλγόριθμος με ιδιαίτερα καλά αποτελέσματα (Kirkpatrick, Gelatt et al. 1983). Ο αλγόριθμος αυτός συνδυάζει την τοπική αναζήτηση με μια απλή στρατηγική διαφοροποίησης, η οποία πιθανοτικά επιτρέπει την αλλοίωση της παραγόμενης λύσης κατά τη διάρκεια της αναζήτησης. Σε κάθε επανάληψη t, μια λύση X_t μετακινείται σε μια νέα τυχαία γειτονιά x_{t+1} και αποδέχεται τη μετακίνηση με μια πιθανότητα. Συγκεκριμένα, αν η λύση στη γειτονιά x_{t+1} έχει καλύτερη αντικειμενική συνάρτηση από τη γειτονιά X_t , η κίνηση είναι πάντα αποδεκτή. Αντίθετα, αν η νέα γειτονιά έχει χειρότερη αντικειμενική συνάρτηση, η πιθανότητα αποδοχής είναι $e^{-\frac{f(x_{t+1})-f(x_t)}{\theta_t}}$, όπου $f(x_t)$ και $f(x_{t+1})$ είναι οι τιμές των αντικειμενικών συναρτήσεων και θ_t είναι η παράμετρος θερμοκρασίας η οποία ξεκινά από μια μεγάλη θετική τιμή και κινείται εκθετικά προς το μηδέν, προσομοιώνοντας μια διαδικασία ψύξης. Το κριτήριο αποδοχής αναφέρεται ως κριτήριο αποδοχής Metropolis στη βιβλιογραφία.

Μια από τις πιο συνηθισμένες μετατροπές στη διαδικασία ψύξης είναι ο πολλαπλασιασμός της παραπάνω σχέσης με ένα παράγοντα α, ο οποίος είναι ένας αριθμός μεταξύ του 0 και του 1, συνήθως πολύ κοντά στο 1, μετά από ένα συγκεκριμένο αριθμό επαναλήψεων. Έτσι, στην αρχή των επαναλήψεων μεγάλες αλλοιώσεις των λύσεων γίνονται αποδεκτές λόγω της υψηλής ‘’θερμοκρασίας’’. Όσο η θερμοκρασία μειώνεται, όλο και μικρότερες αλλοιώσεις γίνονται αποδεκτές.

Εκτός από το κριτήριο Metropolis, υπάρχουν και άλλα κριτήρια που χρησιμοποιούνται όπως η αποδοχή ορίου(threshold accepting), το οποίο οδηγεί σε ντετερμινιστική παραλλαγή του αλγορίθμου προσομοίωσης ανόπτησης. Συγκεκριμένα, για την αποδοχή ορίου, ορίζεται μια τιμή, η οποία αποτελεί το άνω όριο όλων των αλλοιώσεων που μπορούν να συμβούν(Dueck and Scheuer 1990).

Ο αλγόριθμος προσομοίωσης ανόπτησης χάρη στην απλότητα αλλά και στην ευελιξία του χρησιμοποιείται ευρύτατα για την επίλυση πολλών παραλλαγών του προβλήματος δρομολόγησης οχημάτων και είναι ένας από τους κυριότερους μεταευρετικούς αλγορίθμους που χρησιμοποιούνται σε υβριδικούς αλγορίθμους(Suman and Kumar 2006, Zeng, Ong et al. 2007, Lin, Lee et al. 2009).

Ακολουθεί ο ψευδοκώδικας της SA:

1. Επέλεξε μια αρχική λύση.
2. Όσο βρίσκεσαι πάνω από το κατώτατο όριο θερμοκρασίας επανέλαβε:
    a. Όσο βρίσκεσαι κάτω από τις επιτρεπόμενες προσπάθειες για τη συγκεκριμένη θερμοκρασία επανέλαβε:
        i. Πήγαινε σε μια νέα γειτονιά.
        ii. Αν η λύση της είναι καλύτερη από την τωρινή, θέσε αυτή ως τωρινή.

iii. Πήγαινε στο βήμα a.

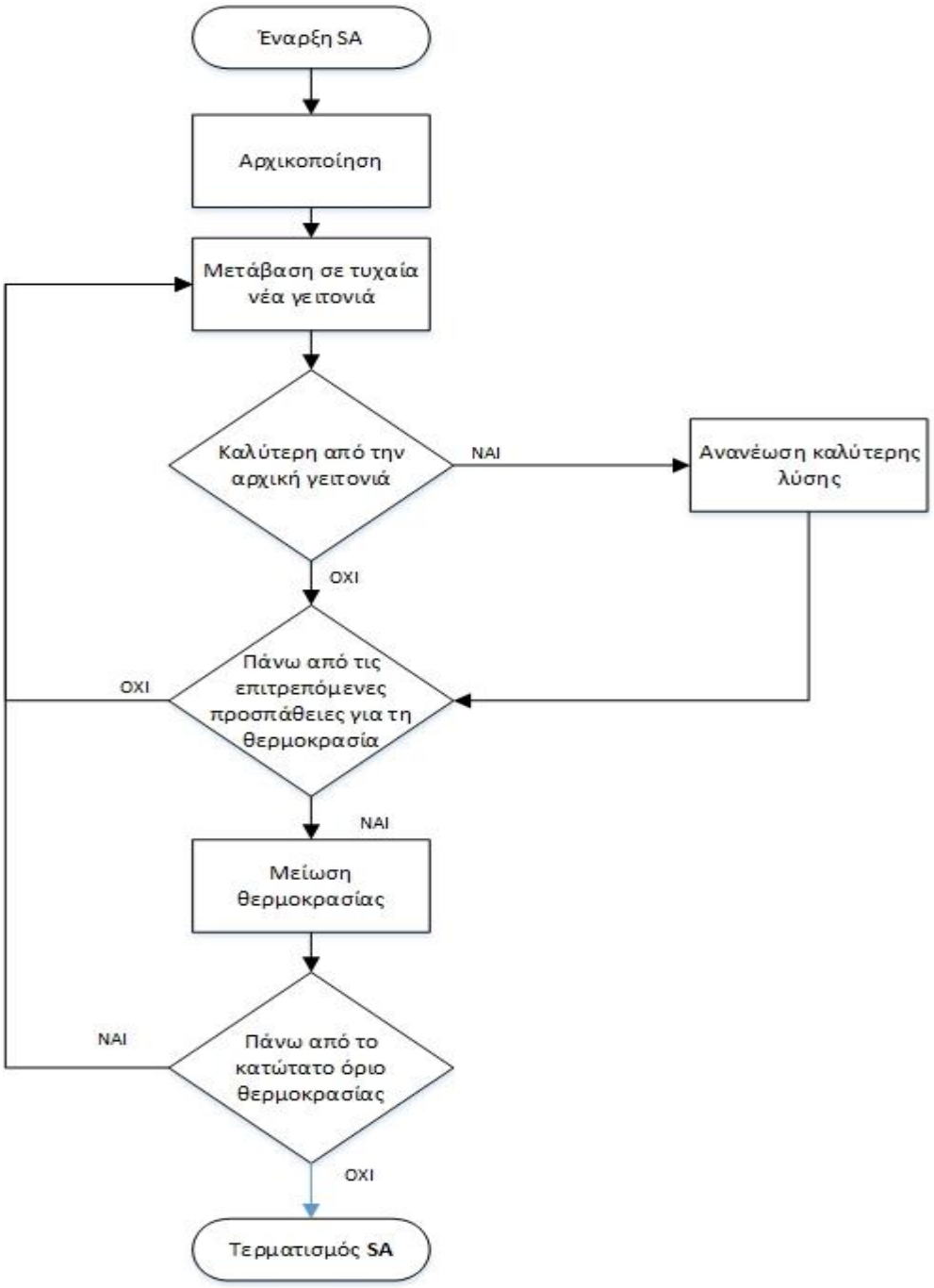
b. Μείωσε τη θερμοκρασία

c. Πήγαινε στο βήμα 2.

3. Τέλος Αλγορίθμου.

Αλγόριθμος 8 Simulated annealing

Ακολουθεί το διάγραμμα ροής του αλγορίθμου προσομοίωσης ανόπτησης:



Σχήμα 20 Διάγραμμα ροής SA

3.3.3.4 Βελτιστοποίηση Σμήνους σωματιδίων(PSO)

Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων(Particle Swarm Optimization) αναπτύχθηκε το 1995 και είναι εμπνευσμένος από τον τρόπο που ένα σμήνος πουλιών προσεγγίζουν την τροφή τους. Έτσι, προσομοιώνει ένα σμήνος σωματιδίων, τα οποία έχουν συγκεκριμένη θέση στο χώρο και συγκεκριμένη ταχύτητα και τείνουν να κινηθούν προς δύο κατευθύνσεις, μια τοπική βέλτιστη του εκάστοτε σωματιδίου και μια ολική βέλτιστη όλων των σωματιδίων.

Ο αλγόριθμος δέχεται σαν είσοδο ένα πλήθος P διανυσμάτων πραγματικών αριθμών διάστασης N και μια μέθοδο αξιολόγησης για τα διανύσματα αυτά. Τα διανύσματα αυτά αναπαριστούν διάφορες λύσεις του προβλήματος, ενώ η μέθοδος αξιολόγησης αντιπροσωπεύει την αντικειμενική συνάρτηση. Η αντικειμενική συνάρτηση σε διαφοροποίηση με τους περισσότερους άλλους μεταευρετικούς αλγορίθμους περιλαμβάνει και τους περιορισμούς, υπό μορφή συνάρτησης ποινής(penalty function).

Κάθε σωματίδιο χαρακτηρίζεται από ένα διάνυσμα θέσης και ένα διάνυσμα ταχύτητας ίδιας διάστασης με το διάνυσμα της θέσης. Σε κάθε επανάληψη, η θέση των σωματιδίων μεταβάλλεται ανάλογα με την ταχύτητα τους. Ο υπολογισμός των νέων θέσεων και ταχυτήτων σε κάθε επανάληψη περιγράφεται από τις παρακάτω εξισώσεις.

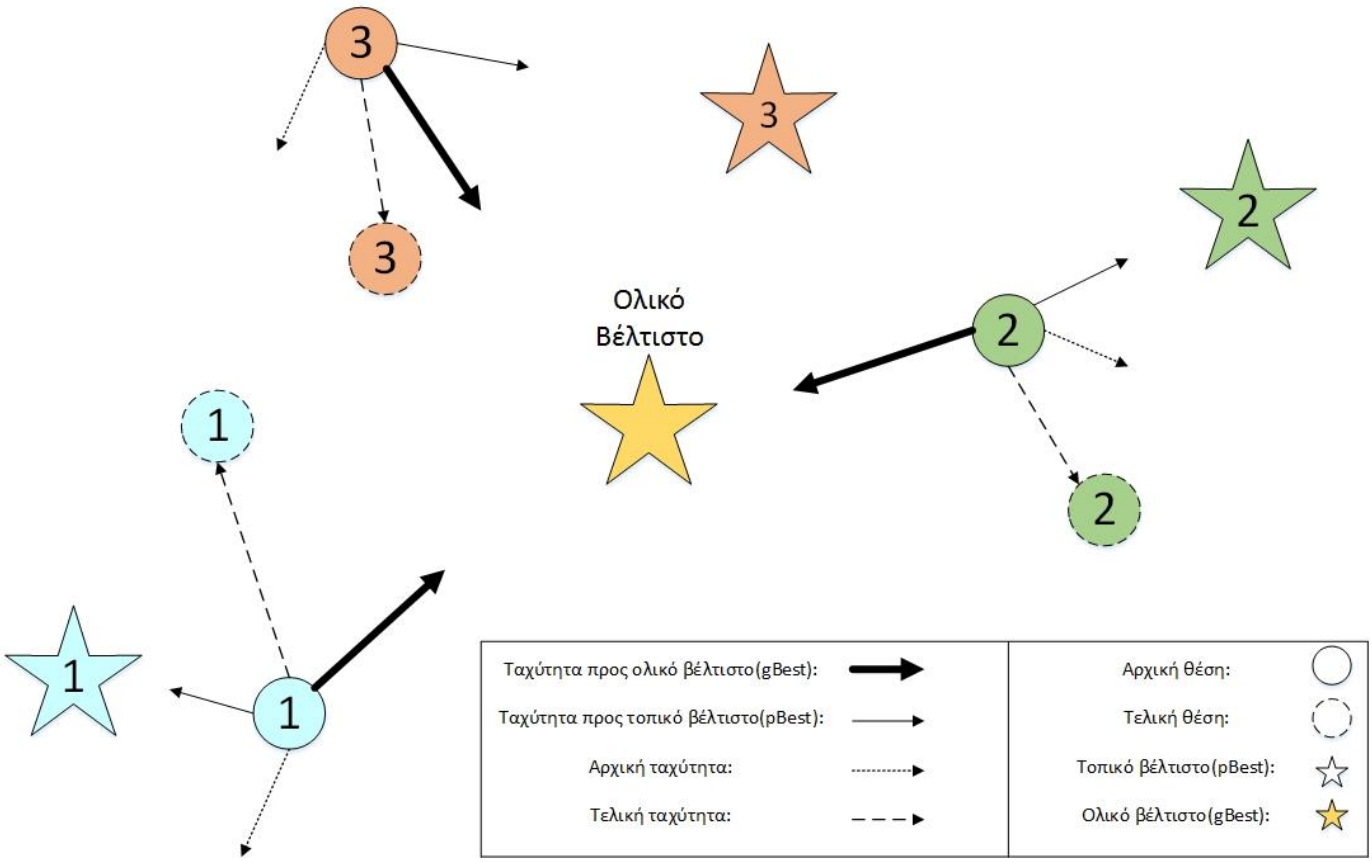
$$V_i^d = wV_i^d + a_1 r_1 \left(pBest_i^d - X_i^d\right) + a_2 r_2 \left(gBest^d - X_i^d\right) \quad (3\text{-}1)$$

$$X_i^d = X_i^d + V_i^d \quad (3\text{-}2)$$

Όπου:

w	συντελεστής αδράνειας (inertia weight)
a_1	Συντελεστής σύγκλισης του σωματιδίου προς τη βέλτιστη θέση από την οποία έχει περάσει
α_2	Συντελεστής σύγκλισης του σωματιδίου προς τη βέλτιστη θέση ολόκληρου του σμήνους σωματιδίων
r_1, r_2	Τυχαίοι πραγματικοί αριθμοί μεταξύ του 0 και του 1
$pBest_i^d$	η διάσταση d της βέλτιστης θέσης από την οποία έχει περάσει το σωματίδιο i
$gBest_i^d$	η διάσταση d της βέλτιστης θέσης ολόκληρου του σμήνους σωματιδίων

Ακολουθεί ένα επεξηγηματικό σχήμα της PSO με 3 σωματίδια:



Σχήμα 21 Παράδειγμα PSO

Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων αναπτύχθηκε για προβλήματα με συνεχείς μεταβλητές απόφασης. Παρόλα αυτά ο αλγόριθμος δύναται να προσαρμοστεί στο πρόβλημα δρομολόγησης οχημάτων, το οποίο είναι πρόβλημα με ακέραιες μεταβλητές απόφασης. Για να γίνει αυτό, πρέπει η αναπαράσταση της λύσης του ακέραιου προβλήματος να μετατραπεί σε διάνυσμα θέσης στον αλγόριθμο ή αντίστροφα ο ίδιος ο αλγόριθμος πρέπει να προσαρμοστεί ώστε να μπορεί να διαχειριστεί ακέραια διανύσματα θέσης και ταχύτητας.

Οι πιο δημοφιλείς προσεγγίσεις στη βιβλιογραφία είναι η προσέγγιση των A.-I Chen, Yang και Wu, η προσέγγιση των Belmecheri, Prins, Yalaoui και Amodeo, η προσέγγιση του Gong και τέλος η προσέγγιση των Marinakis,Marinaki και Dounias.

Η πρώτη προσέγγιση αναπαριστά μια λύση ως ένα δυαδικό διάνυσμα με διάσταση N*K, όπου N είναι το πλήθος των πελατών και K ο πλήθος των οχημάτων. Η μεταβλητή D=α*N+b είναι 1, αν το όχημα α επισκέπτεται τον πελάτη b, αλλιώς είναι 0(Chen, Yang et al. 2006).

Στη δεύτερη προσέγγιση, αντί να αναπαρασταθεί η λύση του VRP ως διάνυσμα πραγματικών αριθμών, προσαρμόζεται ο ίδιος ο αλγόριθμος ώστε κάθε σωματίδιο να αναπαριστά μια σειρά προτίμησης των πελατών για εξυπηρέτηση. Για την αξιολόγηση του κάθε σωματιδίου, η σειρά προτίμησης, δηλαδή το διάνυσμα θέσης του, εισάγεται σε

ευρετικό αλγόριθμο ο οποίος κατασκευάζει ντετερμινιστικά μια λύση και την αξιολογεί(Belmecheri, Prins et al. 2010, Belmecheri, Prins et al. 2013).

Η τρίτη προσέγγιση δίνει έμφαση στην αναπαράσταση της λύσης, ούτως ώστε να μην είναι δυνατή η παραγωγή αδύνατης λύσης. Η κωδικοποίηση της λύσης δημιουργεί μια αλληλουχία πελατών, οι οποίοι θα εξυπηρετηθούν όλοι από το ίδιο όχημα και εισάγεται στον αλγόριθμο ως διάνυσμα με διάσταση ίση με το συνολικό αριθμό των πελατών του προβλήματος. Η θέση κάθε σωματιδίου παριστάνει τη σειρά προτεραιότητας με την οποία θα εξυπηρετηθεί κάθε πελάτης. Κατά την αποκωδικοποίηση, αρχίζουν και κατασκευάζονται σειριακά διαδρομές ξεκινώντας από τον πελάτη με τη μεγαλύτερη προτεραιότητα και προσθέτοντας πελάτες(σε αύξουσα σειρά προτεραιότητας), μέχρι η εναπομένουσα χωρητικότητα του οχήματος να μην επαρκεί για τον επόμενο πελάτη, με αποτέλεσμα το δρομολόγιο να κλείσει. Στη συνέχεια, κατασκευάζεται νέο δρομολόγιο από τον τελευταίο πελάτη του προηγούμενου δρομολογίου που δεν κατάφερε να εισαχθεί και η διαδικασία συνεχίζεται μέχρι όλοι οι πελάτες να εισαχθούν σε ένα δρομολόγιο(Gong, Zhang et al. 2012).

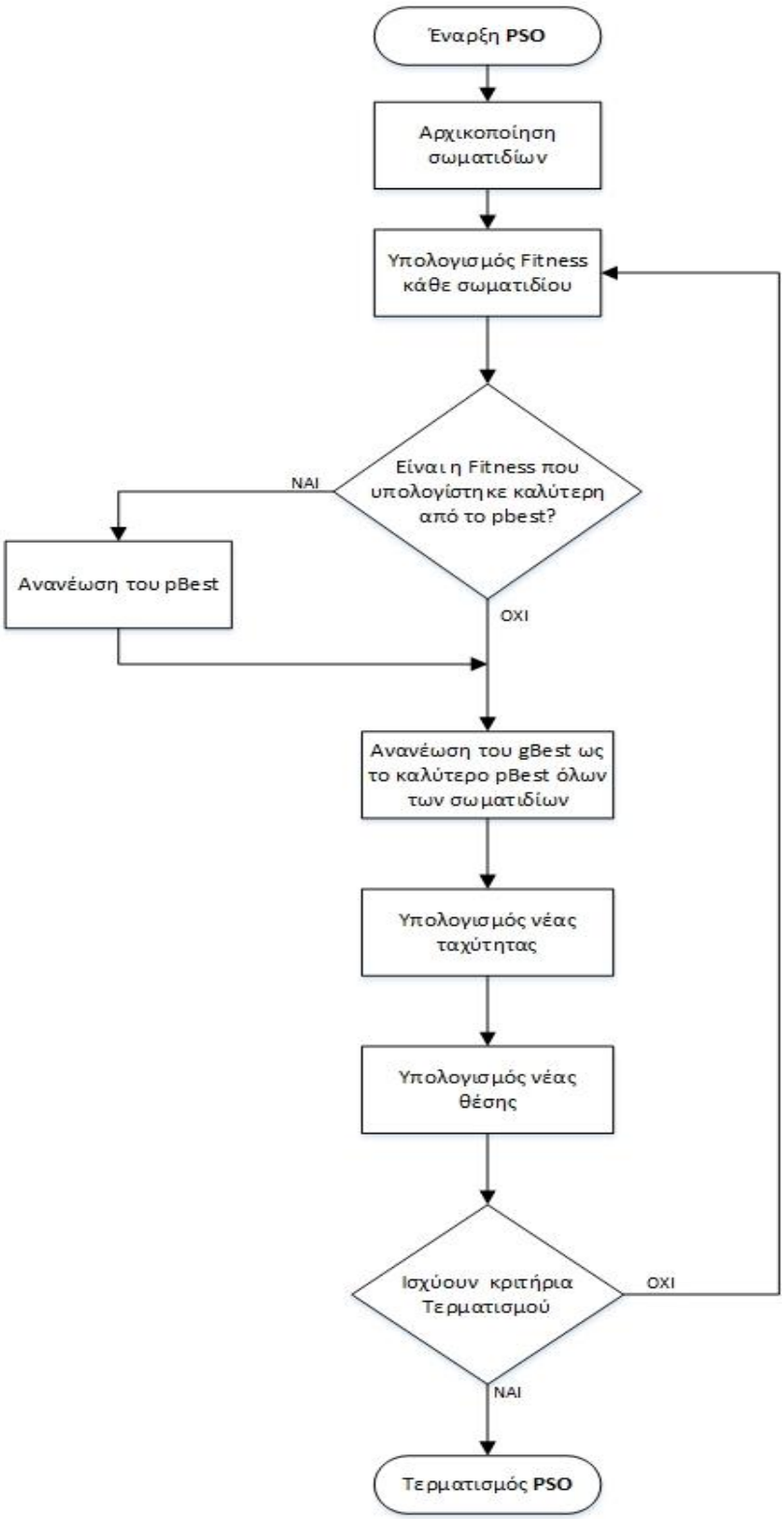
Η τελευταία προσέγγιση αναπαριστά μια λύση του VRP ως ένα δυαδικό διάνυσμα N διαστάσεων, όπου N το πλήθος των πελατών που εξυπηρετεί το όχημα που εκτελεί τη διαδρομή. Οι τιμές δηλώνουν τη σειρά εξυπηρέτησης και παίρνουν ακέραιες τιμές, ενώ μετά την κωδικοποίηση προσαρμόζονται ώστε να να έχουν τιμές εντός του διαστήματος (0,1) με τρόπο τέτοιο, ώστε να διατηρείται η προτεραιότητα των πελατών. Ιδιαίτερη έμφαση δίνεται σε μια μέθοδο κατά την εκτέλεση της μετατόπισης κάθε σωματιδίου, η οποία αποτελεί μια μέθοδο τοπικής αναζήτησης. Η μέθοδος αυτή δέχεται μια αρχική λύση και μια λύση στόχο. Ξεκινώντας από την αρχική λύση χρησιμοποιεί ένα μηχανισμό εναλλαγής της σειράς εξυπηρέτησης των πελατών, προκειμένου να τη μετατρέψει στη λύση στόχο. Αν κατά τη διάρκεια εκτέλεσης, βρεθεί ενδιάμεση λύση καλύτερη από τη λύση στόχο, η λύση παίρνει αυτή την τιμή και η μέθοδος τερματίζει(Marinakis, Marinaki et al. 2010).

Ακολουθεί η ψευδογλώσσα του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων:

1. Αρχικοποίησε τα σωματίδια.
2. Όσο τα κριτήρια τερματισμού δεν ισχύουν επανέλαβε:
    a. Υπολόγισε τη fitness κάθε σωματιδίου.
    b. Αν η Fitness που υπολόγισες είναι η καλύτερη fitness μέχρι τώρα για αυτό το σωματίδιο, θέσε την ως pBest.
    c. Ανανέωσε το gBest ως το καλύτερο pBest όλων των σωματιδίων.
    d. Υπολόγισε τη νέα ταχύτητα.
    e. Υπολόγισε τη νέα θέση.
    f. Πήγαινε στο βήμα 2.
3. Τέλος Αλγορίθμου.

Αλγόριθμος 9 Particle Swarm Optimization

Ακολουθεί το διάγραμμα ροής της PSO:



Σχήμα 22 Διάγραμμα Ροής PSO

4 Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι είναι μια ακόμη κατηγορία μεταευρετικών αλγορίθμων και βασίζονται στη θεωρία της εξέλιξης. Η βασική αρχή τους είναι ότι οι δυνατότερες οντότητες επιβιώνουν. Ένας γενετικός αλγόριθμος υποδιαιρείται σε επιμέρους συστατικά μέρη, τα οποία χρησιμοποιούνται κατά την εκτέλεση του αλγορίθμου. Τα μέρη αυτά είναι η αναπαράσταση(representation), η αντικειμενική συνάρτηση(fitness value), η αρχικοποίηση(initialization), η επιλογή(selection) και διάφοροι τελεστές όπως η διασταύρωση(crossover), η μετάλλαξη(mutation) και η αναστροφή(inversion).

Η εξελικτική διαδικασία που ακολουθείται είναι η προσομοίωση της αμφιγονικής αναπαραγωγής ειδών. Η διαδικασία αυτή μπορεί να περιγραφεί ως εξής: ένα νέο άτομο, διαφορετικό από τους γονείς του, δημιουργείται βάσει της διασταύρωσης χαρακτηριστικών των γονέων αλλά και της μετάλλαξης ορισμένων χαρακτηριστικών του. Το νέο άτομο, το οποίο ονομάζεται απόγονος κληρονομεί χαρακτηριστικά και από τους δύο γονείς του. Κάθε απόγονος ανάλογα με την ποιότητα των χαρακτηριστικών που κληρονόμησε έχει διαφορετική πιθανότητα να επιβιώσει. Έτσι, ένας απόγονος με καλά χαρακτηριστικά έχει μεγαλύτερη πιθανότητα να επιβιώσει συγκριτικά με ένα απόγονο που κληρονόμησε άσχημα χαρακτηριστικά. Ο απόγονος με τη σειρά του, αν επιβιώσει, θα παίξει πλέον το ρόλο του γονέα και θα μεταφέρει τα χαρακτηριστικά του στους απογόνους του. Σκοπός της όλης διαδικασίας είναι η δημιουργία καλύτερων γενεών από τις προηγούμενες.

Η αναλογία της εξελικτικής διαδικασίας με τον αλγόριθμο ξεκίνησε το 1975 και γίνεται με τον παρακάτω τρόπο(Holland 1975):

Κάθε άτομο μπορεί να ταυτιστεί με μια λύση ενός προβλήματος βελτιστοποίησης. Η λύση αυτή αναπαρίσταται με συγκεκριμένο τρόπο. Για την αναπαραγωγή και τη δημιουργία απογόνων χρησιμοποιούνται συγκεκριμένοι τελεστές, η οποίοι επιτρέπουν τη μετάδοση συγκεκριμένων χαρακτηριστικών από κάθε λύση στην επόμενη γενιά με τυχαίο όμως τρόπο. Ταυτόχρονα, για τη διαφοροποίηση των λύσεων και την εξερεύνηση μεγαλύτερου χώρου δυνατών λύσεων, η μετάλλαξη με τυχαίο πάλι τρόπο δύναται να αλλάξει ένα κομμάτι του απογόνου. Ο αλγόριθμος μετά τη δημιουργία της επόμενης γενιάς εξετάζει την ποιότητα τόσο των γονέων, όσο και των απογόνων βάσει της αντικειμενικής τους συνάρτησης και επιλέγει βάσει αυτών το νέο πληθυσμό που θα παίξει το ρόλο του γονέα στην επόμενη επανάληψη(Syswerda 1991, Hwang 2002, Baker and Ayechew 2003, Berger and Barkaoui 2003, Ho, Ho et al. 2008).

4.1 Περιγραφή

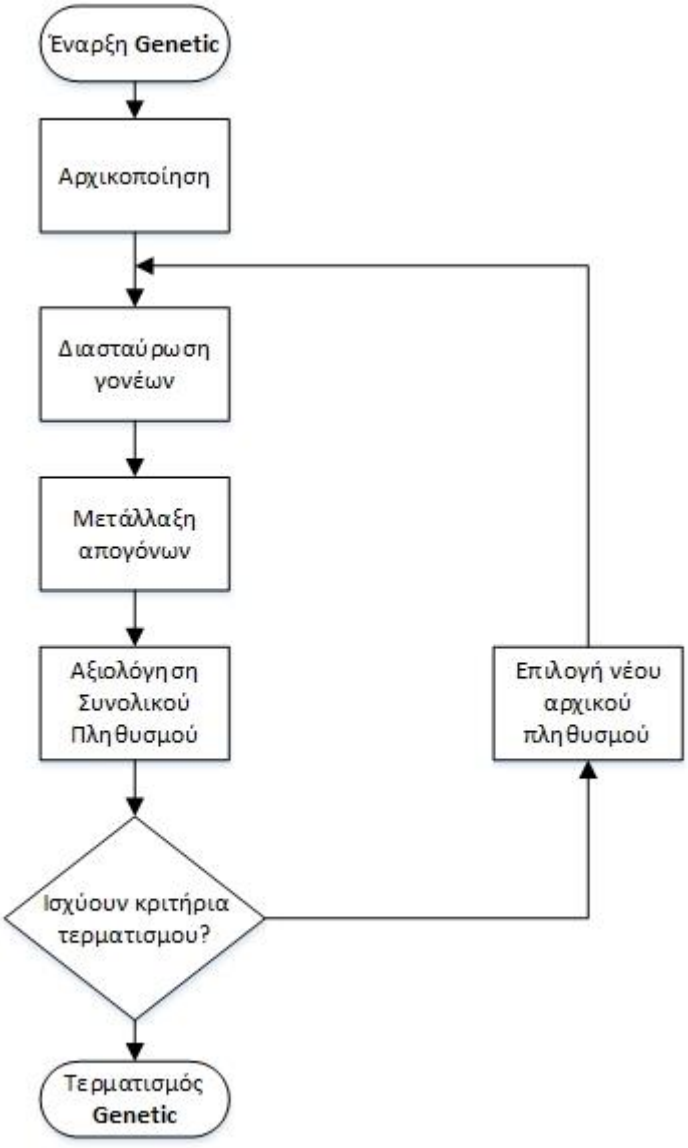
Ο γενετικός αλγόριθμος αφού δημιουργηθεί ένα αρχικός πληθυσμός, λειτουργεί επαναληπτικά. Αρχικά, διασταυρώνει τους γονείς ανά δύο και δημιουργεί το νέο πληθυσμό. Στη συνέχεια, μεταλλάσσει τα γονίδια ορισμένων απογόνων με βάση κάποια πιθανότητα. Ο συνολικός πληθυσμός που παράγεται, δηλαδή ο νέος μαζί με τον παλιό, αξιολογείται με βάση μια αντικειμενική συνάρτηση. Τέλος, επιλέγεται ένα κομμάτι του συνολικού πληθυσμού, συνήθως ίσο σε μέγεθος με τον αρχικό πληθυσμό, το οποίο χρησιμοποιείται ως γονείς για την παραγωγή της νέας γενιάς και ο κύκλος ξαναρχίζει.

Ακολουθεί ο γενετικός αλγόριθμος σε ψευδοκώδικα:

1. Φτιάξε τον αρχικό πληθυσμό.
2. Όσο τα κριτήρια τερματισμού δεν ισχύουν επανέλαβε:
    a. Διασταύρωσε τους γονείς ανά δύο και δημιούργησε το νέο πληθυσμό.
    b. Μετάλλαξε ορισμένα γονίδια από μερικούς απογόνους βάσει μιας πιθανότητας.
    c. Αξιολόγησε το συνολικό πληθυσμό.
    d. Επέλεξε τον πληθυσμό που θα λειτουργήσει ως αρχικός πληθυσμός στη νέα γενιά.
    e. Πήγαινε στο βήμα 2.
3. Τέλος Αλγορίθμου.

Αλγόριθμος 10 Genetic Algorithm

Ακολουθεί το διάγραμμα ροής του γενετικού αλγορίθμου:



Σχήμα 23 Διάγραμμα Ροής genetic

4.2 Αρχικός Πληθυσμός

Το πρώτο βήμα κάθε γενετικού αλγορίθμου είναι η δημιουργία ενός αρχικού πληθυσμού. Συνήθως, ο αρχικός πληθυσμός αποτελείται από τελείως τυχαία διαμορφωμένες λύσεις. Το μέγεθος του πληθυσμού είναι δύσκολο να ορισθεί αλλά συνήθως προτιμάται ένας αρκετά μεγάλος αρχικός πληθυσμός. Όσο μεγαλώνει ο αρχικός πληθυσμός τόσο μεγαλύτερο μέρος του χώρου των δυνατών λύσεων θα καλυφθεί, αλλά η διαδικασία θα γίνει πιο αργή για την εύρεση μιας καλής λύσης. Γι'αυτόν το λόγο, υπάρχει η δυνατότητα περιορισμού του χώρου των δυνατών λύσεων, αν είναι γνωστό το μέρος στο οποίο θα αναζητηθεί η βέλτιστη λύση. Έτσι πολλές φορές, μέρος του αρχικού πληθυσμού αντικαθίσταται από λύσεις ευρετικών μεθόδων.

Για το VRP λόγω της πολυπλοκότητας του προβλήματος, χρησιμοποιούνται συνήθως μερικές από τις ευρετικές μεθόδους που περιγράφησαν σε προηγούμενη ενότητα, όπως ο αλγόριθμος Savings, ο αλγόριθμος Mole&Jameson καθώς και ο αλγόριθμος Sweep. Η ποιότητα των ευρετικών αυτών λύσεων είναι προφανώς καλύτερη από τις τυχαία διαμορφωμένες λύσεις. Παρόλα αυτά, οι τυχαίες λύσεις δύναται να έχουν χαρακτηριστικά που τελικά θα οδηγήσουν στη βέλτιστη λύση σε σχέση με τις απλές ευρετικές λύσεις που θα οδηγήσουν μόνο σε μια καλής ποιότητας λύση. Αποτέλεσμα όλων των παραπάνω είναι η υιοθέτηση ενός συνδυασμού ευρετικών και τυχαία διαμορφωμένων λύσεων με σκοπό τόσο τη μείωση του χώρου αναζήτησης, όσο και τη διαφοροποίηση των χαρακτηριστικών του πληθυσμού για την εύρεση της βέλτιστης λύσης(Baker and Ayechew 2003).

Στο σχήμα που ακολουθεί φαίνεται πως μια λύση με κακής ποιότητας αντικειμενική συνάρτηση (απόγονος 11) μπορεί να οδηγήσει σε πολύ καλύτερη λύση(θέση 12) σε σχέση με κάποια λύση με πολύ καλύτερη αντικειμενική συνάρτηση(απόγονος 9).

Σχήμα 24 Παράδειγμα αρχικού πληθυσμού

4.3 Αναπαράσταση λύσεων

Για την λειτουργία του γενετικού αλγορίθμου, απαραίτητη προϋπόθεση είναι η αναπαράσταση των λύσεων σε μια ακολουθία που λέγεται χρωμόσωμα. Για να μετατραπεί μια λύση σε χρωμόσωμα απαιτείται μια διαδικασία κωδικοποίησης, ενώ αντίθετα για να μετατραπεί ένα χρωμόσωμα σε λύση απαιτείται μια διαδικασία αποκωδικοποίησης.

Υπάρχουν πολλοί τρόποι αναπαράστασης του χρωμοσώματος στη βιβλιογραφία, ο καθένας με τα δικά του πλεονεκτήματα και μειονεκτήματα. Μια καλή κωδικοποίηση των λύσεων ενός VRP προβλήματος πρέπει να αναγνωρίζει τον αριθμό των οχημάτων, τους πελάτες που εξυπηρετούνται από κάθε όχημα και τη σειρά με την οποία εξυπηρετούνται.

Μία από τις συνήθεις κωδικοποιήσεις στους γενετικούς αλγορίθμους είναι η δυαδική, όπου το χρωμόσωμα αποτελείται από μια ακολουθία μηδέν και ένα τα οποία προσδιορίζουν τη λύση με κατάλληλο τρόπο. Η κωδικοποίηση αυτή μπορεί να χρησιμοποιηθεί και στο πρόβλημα δρομολόγησης οχημάτων αλλά δεν θεωρείται αποδοτική μέθοδος αναπαράστασης των λύσεων.

Η πιο συνηθισμένη μέθοδος, η οποία θα χρησιμοποιηθεί και στην παρούσα διπλωματική είναι η αρίθμηση των πελατών και η αναπαράσταση διαδρομών στο χρωμόσωμα με κατάλληλο τρόπο. Πιο συγκεκριμένα, κάθε πελάτης χαρακτηρίζεται από ένα μοναδικό αριθμό, και το χρωμόσωμα αποτελείται από μια ακολουθία των αριθμών των πελατών. Η διάσταση του χρωμοσώματος, προφανώς, είναι ίση με τον αριθμό των πελατών. Κάθε πελάτης βρίσκεται σε ακριβώς μια θέση στο χρωμόσωμα, αφού εξυπηρετείται μόνο μια φορά. Ξεκινώντας από την αρχή του χρωμοσώματος, ανατίθενται πελάτες στο πρώτο όχημα, σύμφωνα με την ακολουθία αριθμών. Όταν η εναπομένουσα (Salhi and Petch 2007) χωρητικότητα του οχήματος δεν είναι αρκετή για την κάλυψη της ζήτησης ενός πελάτη, το δρομολόγιο του οχήματος ολοκληρώνεται και αρχίζει το δρομολόγιο του επόμενου οχήματος από τον τελευταίο πελάτη που δεν χώρεσε στο προηγούμενο δρομολόγιο. Η διαδικασία αυτή συνεχίζεται, μέχρι το τέλος του χρωμοσώματος, μέχρι όλοι οι πελάτες να εισαχθούν σε κάποιο δρομολόγιο. Για την αποκωδικοποίηση της λύσης εκτός από τη διαδρομή που δημιουργήθηκε σύμφωνα με το χρωμόσωμα, προστίθεται στην αρχή και το τέλος η κεντρική αποθήκη, η οποία σύμφωνα με τους κανόνες του απλού VRP αποτελεί αφετηρία και τερματισμό όλων των οχημάτων(Ombuki, Ross et al. 2006).

Ακολουθεί ένα παράδειγμα αυτής της αναπαράστασης χρωμοσώματος:

Σχήμα 25 Παράδειγμα αναπαράστασης

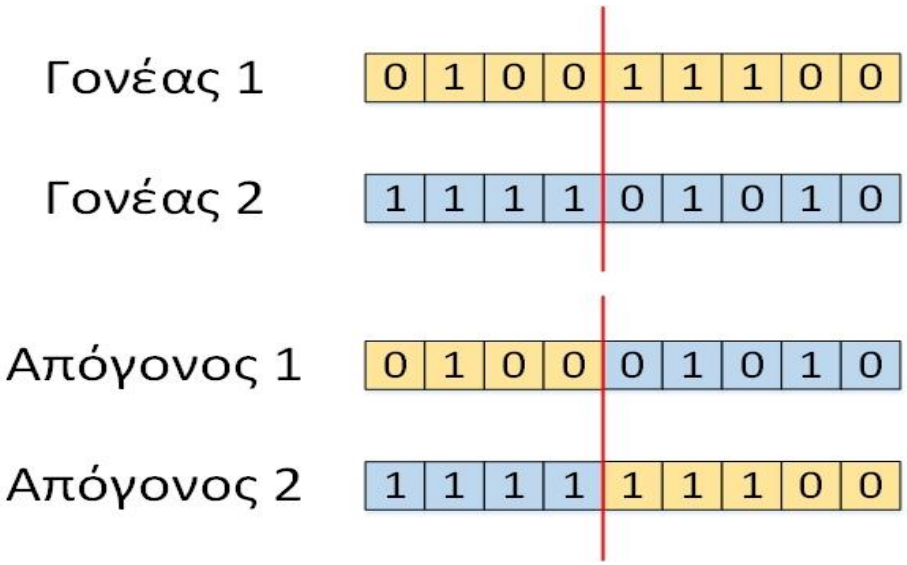
4.4 Διασταύρωση(Crossover)

Η διασταύρωση(crossover) είναι ο βασικός γενετικός τελεστής, ο οποίος αναπαριστά την αναπαραγωγή μεταξύ των γονέων. Χρησιμοποιείται σε ένα ζεύγος λύσεων και τις ανασυνδυάζει με συγκεκριμένο τρόπο, προκειμένου να παράγει έναν ή περισσότερους απογόνους. Οι απόγονοι μοιράζονται μερικά χαρακτηριστικά από τους γονείς τους και με αυτό τον τρόπο τα χαρακτηριστικά μεταφέρονται στις επόμενες γενιές. Παρόλα αυτά δεν είναι δυνατή η δημιουργία νέων χαρακτηριστικών.

Η λειτουργικότητα της διασταύρωσης είναι άρρηκτα συνδεδεμένη με τον τρόπο αναπαράστασης των δεδομένων και η αποτελεσματικότητα του εξαρτάται από το πόσο καλά έχει προσαρμοστεί στο εκάστοτε πρόβλημα. Για το VRP έχουν αναπτυχθεί στη βιβλιογραφία αρκετοί τύποι διασταυρώσεων. Η παρούσα διπλωματική χρησιμοποιεί βάσει του τρόπου αναπαράστασης των δεδομένων την διασταύρωση PMX. Παρακάτω αναλύονται η απλή διασταύρωση που συναντάται στους γενετικούς αλγορίθμους καθώς και η διασταύρωση PMX(Üçoluk 1997).

4.4.1 Απλή Διασταύρωση

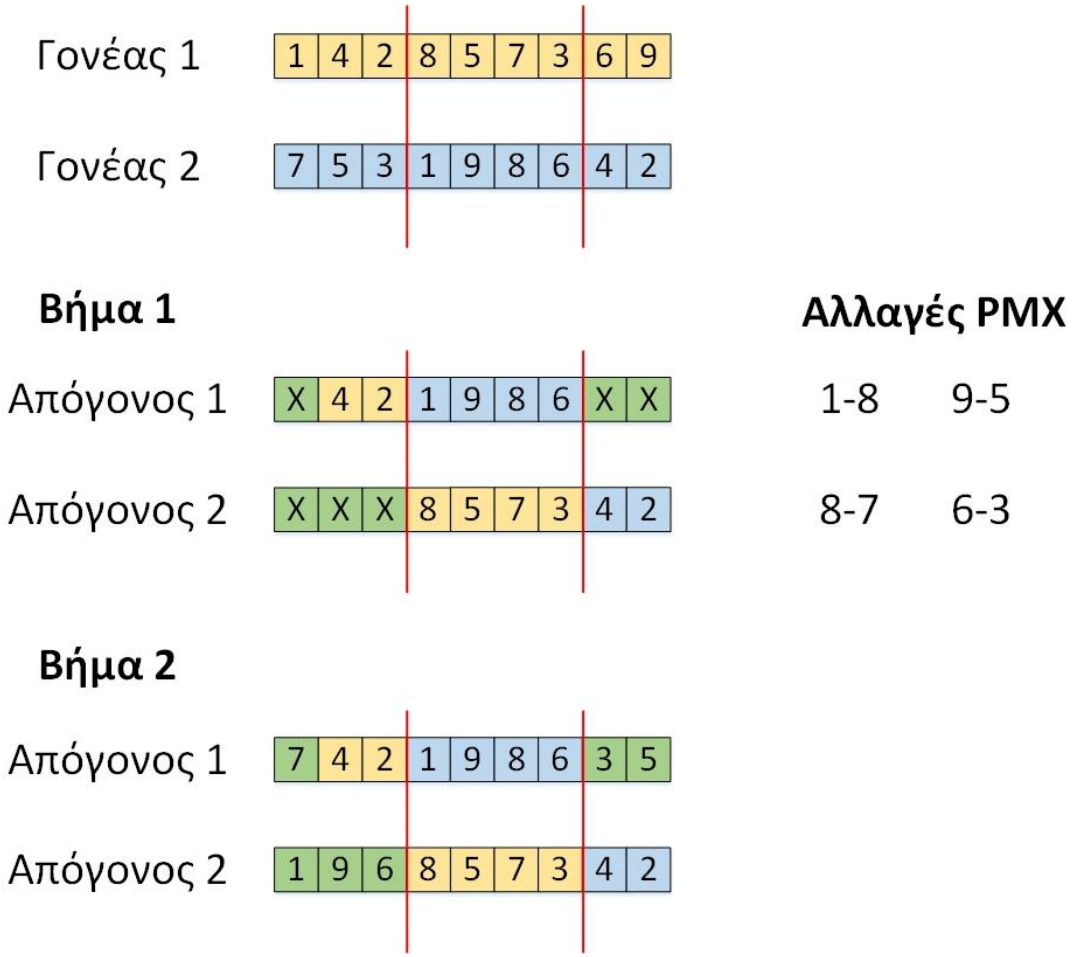
Η απλή διασταύρωση(simple crossover) ξεκινά με δύο λύσεις-γονείς, επιλέγει ένα σημείο κοπής(cut point), το οποίο θα χρησιμοποιηθεί για να διαχωριστεί κάθε γονέας σε δύο μέρη. Οι δύο απόγονοι που δημιουργούνται χρησιμοποιούν, ο μεν πρώτος το πρώτο μέρος του πρώτου γονέα και το δεύτερο μέρος του δεύτερου γονέα, ενώ ο δεύτερος το πρώτο μέρος του δεύτερου γονέα και το δεύτερο μέρος του πρώτου γονέα(Üçoluk 1997).



Σχήμα 26 Παράδειγμα Simple crossover

4.4.2 Διασταύρωση PMX(PMX crossover)

Η διασταύρωση PMX(Partially Matched Crossover) είναι μια παραλλαγή της απλής διασταύρωσης που περιγράφηκε στην προηγούμενη ενότητα. Η μέθοδος αρχικά επιλέγει δύο γονείς και δύο σημεία κοπής σε πλήρη αντιστοιχία με την απλή διασταύρωση. Εν συνεχεία τα μέρη των δύο γονέων, που βρίσκονται μεταξύ των σημείων κοπής εναλλάσσονται για τη δημιουργία των δύο απογόνων. Τα υπόλοιπα μέρη του χρωμοσώματος δεν μπορούν να αντιγραφούν ένα προς ένα, λόγω της δημιουργίας μη εφικτών λύσεων στη γενική περίπτωση. Οι μη εφικτές λύσεις δημιουργούνται από την ύπαρξη κάποιου πελάτη δύο φορές και την μη ύπαρξη κάποιου άλλου πελάτη στη λύση. Έτσι, αρχικά συμπληρώνονται όλοι οι πελάτες, οι οποίοι δεν περιλαμβάνονται στο ενδιάμεσο τμήμα(μεταξύ των δύο σημείων κοπής) κανενός εκ των γονέων. Στη συνέχεια, οι υπόλοιποι πελάτες συμπληρώνονται σύμφωνα με τις μεταβολές στα ενδιάμεσα τμήματα, σε πλήρη αντιστοιχία. Παρακάτω, φαίνεται ένα παράδειγμα της διασταύρωσης PMX. Στο πρώτο βήμα συμπληρώνονται τα ενδιάμεσα τμήματα, καθώς και όσοι πελάτες δεν περιλαμβάνονται σε κανένα από τα ενδιάμεσα τμήματα των γονέων. Στο δεύτερο βήμα συμπληρώνονται οι υπόλοιποι πελάτες, σύμφωνα με τις αλλαγές που περιγράφονται δίπλα από το πρώτο βήμα(Üçoluk 2002, Kumar 2012).



Σχήμα 27 Παράδειγμα PMX crossover

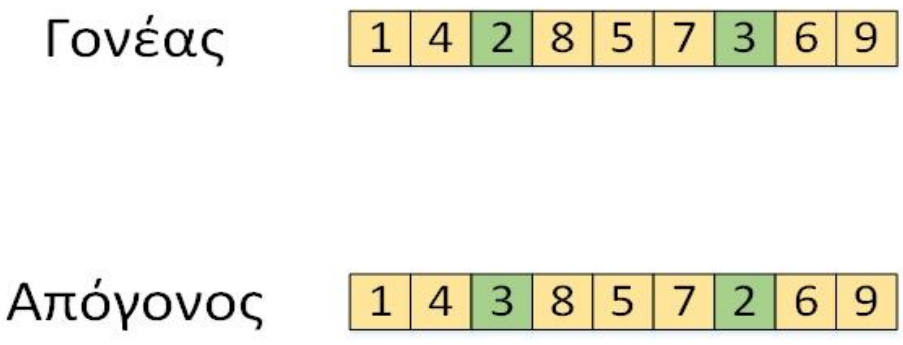
4.5 Μετάλλαξη(Mutation)

Η μετάλλαξη(mutation) είναι ένας άλλος τελεστής που χρησιμοποιείται κατά κόρον στους γενετικούς αλγορίθμους. Ο τελεστής αυτός χρησιμοποιείται σε κάθε λύση ξεχωριστά βάσει πιθανοτήτων. Συγκεκριμένα, όταν χρησιμοποιείται η μετάλλαξη, προσθέτει νέα χαρακτηριστικά στον πληθυσμό που δεν υπήρχαν πριν. Σε αντίθεση με τη διασταύρωση, δεν χρησιμοποιείται σε ζευγάρια λύσεων και σκοπός του δεν είναι η διατήρηση χαρακτηριστικών στις επόμενες γενιές αλλά η δημιουργία νέων με σκοπό τη διαφοροποίηση του πληθυσμού και την εξερεύνηση μεγαλύτερου χώρου δυνατών λύσεων. Παρόλα αυτά, οι πολύ μεγάλες αλλαγές στον πληθυσμό δεν είναι αποδεκτές, γιατί ο τελεστής λειτουργεί σαν μέθοδος τυχαίας αναζήτησης σε αυτή την περίπτωση.

Στο πρόβλημα δρομολόγησης οχημάτων, με βάση την αναπαράσταση των λύσεων που προτάθηκε σε προηγούμενη ενότητα, η μετάλλαξη επιλέγει τυχαία δύο πελάτες και ανταλλάσσει τις θέσεις τους. Έτσι, είναι δυνατή η δημιουργία εντελώς διαφορετικών λύσεων, όπου δύο πελάτες αλλάζουν όχι μόνο σειρά εξυπηρέτησης αλλά πιθανώς και όχημα από το οποίο θα εξυπηρετηθούν.

Η μετάλλαξη χρησιμοποιείται συνήθως μετά τη διασταύρωση στους απογόνους που δημιουργούνται αν αυτοί επιλεγούν. Το αποτέλεσμα της, επηρεάζει την αντικειμενική συνάρτηση του χρωμοσώματος στο οποίο χρησιμοποιήθηκε, είτε θετικά, είτε αρνητικά εφόσον βασίζεται στην τυχαιότητα. Πάραυτα η τυχαιότητα μπορεί να επηρεάσει αρνητικά την αποτελεσματικότητα του αλγορίθμου. Αντίθετα, η έλλειψη μετάλλαξης μπορεί επίσης να οδηγήσει τον αλγόριθμο σε πρόωρη σύγκλιση και σε αδυναμία εύρεσης του ολικού βελτίστου. Για αυτό το λόγο, πρέπει να γίνεται μια εξισορρόπηση στη χρήση της μετάλλαξης κατά την παραμετροποίηση του αλγορίθμου(Bäck 1992, Srinivas and Patnaik 1994, Whitley 1994).

Ακολουθεί ένα παράδειγμα μετάλλαξης με βάση την κωδικοποίηση που προτάθηκε σε προηγούμενη ενότητα:



Σχήμα 28 Παράδειγμα mutation

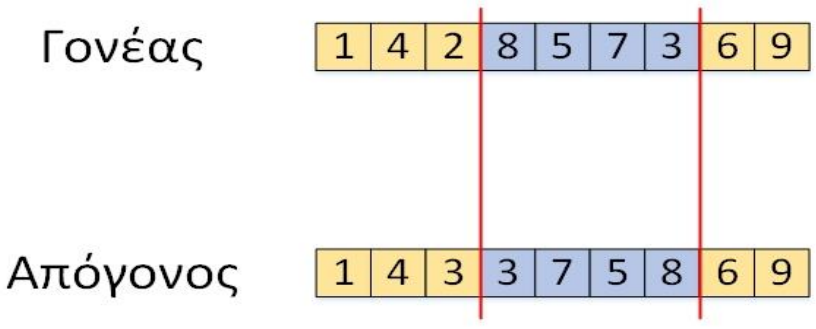
4.6 Αναστροφή(Inversion)

Ο τελευταίος τελεστής που χρησιμοποιείται σε ορισμένους γενετικούς αλγορίθμους είναι η αναστροφή(inversion). Αντίστοιχα με τη μετάλλαξη, χρησιμοποιείται σε κάθε λύση ξεχωριστά βάσει πιθανοτήτων. Σκοπός της αναστροφής δεν είναι ούτε η μεταβίβαση χαρακτηριστικών των γονέων στις επόμενες γενιές, αλλά ούτε και η δημιουργία εντελώς νέων χαρακτηριστικών. Αντίθετα, η αναστροφή προσπαθεί να βελτιώσει τα χαρακτηριστικά του υπάρχοντος πληθυσμού κάνοντας μικρές αλλαγές(Tao and Michalewicz 1998).

Η διαδικασία που ακολουθείται ξεκινά με την επιλογή δύο σημείων κοπής σε ένα μέλος του πληθυσμού, συνήθως σε απογόνους που μόλις δημιουργήθηκαν. Στη συνέχεια, τα στοιχεία του τμήματος, ανάμεσα στα δύο σημεία κοπής, αντιστρέφονται. Για να χρησιμοποιηθεί ο τελεστής της αναστροφής, πρέπει η παραγόμενη λύση, ο νέος απόγονος δηλαδή να είναι καλύτερος από τον προηγούμενο.

Σε ένα πρόβλημα VRP, ο παραπάνω έλεγχος γίνεται βάσει των αποστάσεων ή του κόστους. Ειδικότερα, σε περίπτωση χρήσης των αποστάσεων, γίνεται έλεγχος του αθροίσματος της απόστασης του πρώτου στοιχείου με το προηγούμενο του και του τελευταίου με το επόμενο του, με το άθροισμα της απόστασης του νέου πρώτου στοιχείου με το προηγούμενο του και του νέου τελευταίου στοιχείου με το επόμενο του(Watterson, Ewens et al. 1982).

Ακολουθεί ένα παράδειγμα αναστροφής. Ο έλεγχος που αναφέρθηκε νωρίτερα, σε αυτό το παράδειγμα είναι το άθροισμα της απόστασης του πελάτη (2) και του πελάτη (8) και της απόστασης του πελάτη (3) και (6), με το άθροισμα της απόστασης του πελάτη (2) και του πελάτη (3) και της απόστασης του πελάτη (8) και (6).



Σχήμα 29 Παράδειγμα inversion

4.7 Αντικειμενική συνάρτηση(Fitness)

Η αντικειμενική συνάρτηση χρησιμοποιείται για την αξιολόγηση και την επιλογή εκείνων των μελών του συνολικού πληθυσμού, που θα επιζήσουν και θα αποτελέσουν τους γονείς στην επόμενη επανάληψη. Η αντικειμενική συνάρτηση αποτελεί το μέτρο της ποιότητας των λύσεων και επιτρέπει τη σύγκριση μεταξύ τους.

Για την αποφυγή πρόωρης σύγκλισης χρησιμοποιείται συχνά η κλιμάκωση(scaling) της αντικειμενικής συνάρτησης. Η κλιμάκωση αυτή είναι ιδιαίτερα χρήσιμη σε μετέπειτα επαναλήψεις, όπου η αντικειμενική συνάρτηση του πληθυσμού πλησιάζει τη βέλτιστη τιμή με αποτέλεσμα τα μέτρια και τα καλά μέλη του πληθυσμού να έχουν σχεδόν ίδιες πιθανότητες να επιλεγούν.

Υπάρχουν αρκετοί τρόποι κλιμάκωσης της αντικειμενικής συνάρτησης, όπως γραμμική κλιμάκωση, με ή χωρίς περικοπή του σ κλιμάκωση, κλιμάκωση που ακολουθεί το νόμο της δύναμης.

Η γραμμική κλιμάκωση μπορεί να περιγραφεί ως :

$$f_i' = a \cdot f_i + b \qquad (4\text{-}1)$$

Όπου, α και b είναι δύο επιλεγόμενες τιμές, ούτως ώστε η μέση τιμή της αντικειμενικής συνάρτησης και η κλιμακούμενη αντικειμενική συνάρτηση να ταυτίζονται. Η γραμμική μέθοδος παράγει αρκετά καλά αποτελέσματα αλλά δημιουργεί προβλήματα στις τελευταίες επαναλήψεις, όπου μέλη του πληθυσμού έχουν πολύ χαμηλές αντικειμενικές συναρτήσεις, κοντά η μια στην άλλη με αποτέλεσμα την δημιουργία αρνητικών τιμών της αντικειμενικής συνάρτησης. Τέλος, οι τιμές α, b εξαρτώνται μόνο από τον πληθυσμό και όχι από το πρόβλημα που επιλύεται.

Η αντικειμενική συνάρτηση είναι εύκολο να κατασκευασθεί για ορισμένα προβλήματα. Ειδικά για προβλήματα βελτιστοποίησης, η αντικειμενική συνάρτηση ταυτίζεται με την αντικειμενική συνάρτηση του μαθηματικού μοντέλου του προβλήματος. Έτσι, για το VRP, η αντικειμενική συνάρτηση είναι το συνολικό κόστος ή η συνολική απόσταση που διανύθηκε. Παρόλα αυτά εκτός της αντικειμενικής συνάρτησης της μαθηματικής μοντελοποίησης πρέπει να ληφθούν υπόψη και οι περιορισμοί κάθε προβλήματος.

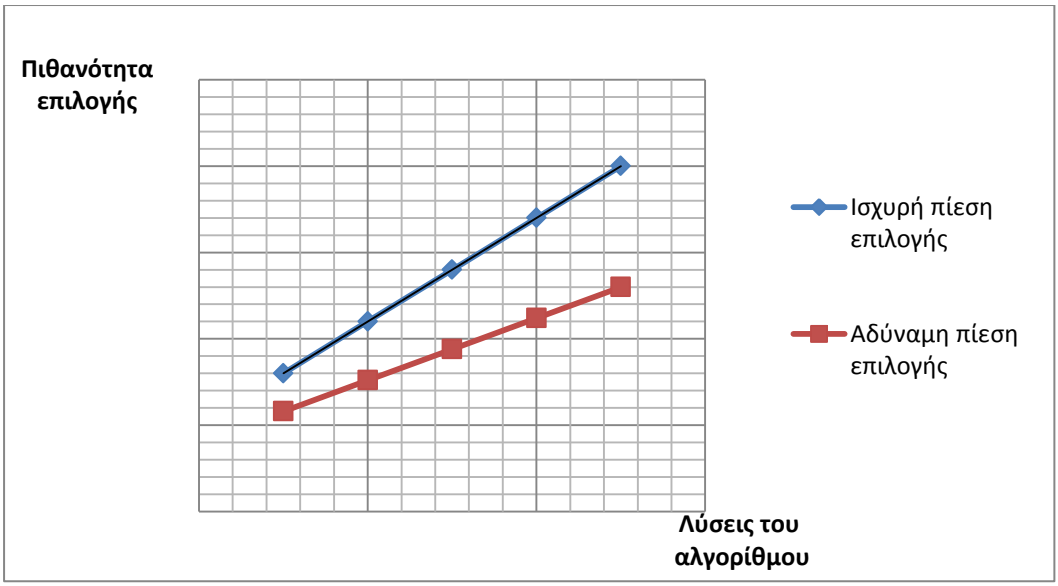
Για τη διαχείριση των περιορισμών του προβλήματος, υπάρχουν δύο τρόποι. Ο πρώτος τρόπος είναι η απόρριψη όλων των μελών του πληθυσμού που δημιουργούν μη εφικτές λύσεις, μέθοδος που χρησιμοποιείται για προβλήματα με μικρούς χώρους εφικτών λύσεων. Ο δεύτερος τρόπος είναι η υιοθέτηση ποινής για παράβαση κάποιου περιορισμού. Η ποινή αυτή χειροτερεύει την ποιότητα της παραγόμενης λύσης, αλλά επιτρέπει στον αλγόριθμο να ψάξει σε χώρους εκτός των δυνατών λύσεων, γεγονός που συχνά οδηγεί στην εύρεση του ολικού βελτίστου του προβλήματος.

4.8 Επιλογή (Selection)

Η επιλογή του νέου πληθυσμού στο τέλος κάθε επανάληψης είναι μια διαδικασία εξαρτώμενη από δύο βασικούς παράγοντες, την διαφοροποίηση του πληθυσμού και την πίεση επιλογής(selective pressure). Οι παράγοντες αυτοί είναι άρρηκτα συνδεδεμένοι μεταξύ τους. Αν ο πληθυσμός γίνει υπερβολικά ομοιογενής, ο μόνος παράγοντας διαφοροποίησης θα είναι η μετάλλαξη. Έτσι, κατά την επιλογή της μεθόδου επιλογής του νέου πληθυσμού πρέπει να δοθεί ιδιαίτερη σημασία.

Ένα από τα μεγαλύτερα προβλήματα με τη χρήση γενετικού αλγορίθμου είναι η πρόωρη σύγκλιση. Η σύγκλιση είναι ένας τρόπος μέτρησης της ταχύτητας με την οποία βελτιώνεται ο πληθυσμός. Μια πολύ γρήγορη ταχύτητα συνεπάγεται θανάτωση των πιο αδύναμων μελών του πληθυσμού, προτού περάσουν τα χαρακτηριστικά τους στις επόμενες γενιές. Έτσι, η πίεση επιλογής είναι ένας τρόπος μέτρησης του πόσο συχνά επιλέγονται οι καλύτεροι απόγονοι συγκριτικά με τους πιο αδύναμους. Η πίεση επιλογής διακρίνεται σε ισχυρή όταν επιλέγονται οι ισχυροί απόγονοι τις περισσότερες φορές και αδύναμη όταν οι αδύναμοι απόγονοι έχουν αρκετά μεγάλη πιθανότητα επιβίωσης(Prins 2004).

Στο παρακάτω διάγραμμα φαίνεται ένα παράδειγμα 5 λύσεων. Ο κατακόρυφος άξονας δείχνει την πιθανότητα επιλογής κάθε λύσης. Η καμπύλη (1) δείχνει μια ισχυρή πίεση επιλογής, ενώ η καμπύλη (2) μια πιο αδύναμη.



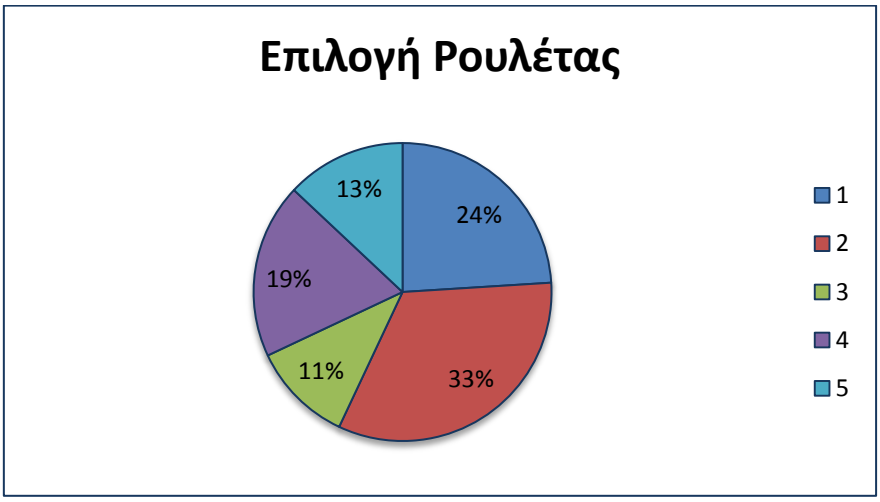
Σχήμα 30 Παράδειγμα Επιλογής Πίεσης

Οι κυριότερες μέθοδοι επιλογής για το γενετικό αλγόριθμο είναι η μέθοδος ρουλέτας, η μέθοδος ιεράρχησης και η μέθοδος επιλογής τουρνουά.

4.8.1 Μέθοδος επιλογής ρουλέτας

Η μέθοδος επιλογής ρουλέτας(roulette wheel method) είναι η πιο συχνά χρησιμοποιούμενη μέθοδος επιλογής στους γενετικούς αλγορίθμους. Η βασική ιδέα της μεθόδου είναι η αναλογική ανάθεση μιας πιθανότητας σε κάθε μέλος του πληθυσμού βάσει της αντικειμενικής του συνάρτησης(Prins 2004).

Το ακόλουθο σχήμα αναπαριστά ένα παράδειγμα επιλογής ρουλέτας. Κάθε μέλος του πληθυσμού(1,2,3,4,5) βρίσκεται ανάλογα με την αντικειμενική του συνάρτηση σε συγκεκριμένο κομμάτι του τροχού. Κατά το ''γύρισμα'' του τροχού η πιθανότητα να επιλεγεί το μέλος (2) του πληθυσμού είναι πολύ μεγαλύτερη απ' ότι να επιλεγεί το μέλος (3) του πληθυσμού.



Σχήμα 31 Παράδειγμα μεθόδου ρουλέτας

Το μειονέκτημα της μεθόδου είναι η απευθείας χρήση της τιμής της αντικειμενικής συνάρτησης. Έτσι, σε προβλήματα όπου η λύση έχει πολύ μικρή τιμή συγκριτικά με τις υπόλοιπες λύσεις, η πιθανότητα επιλογής είναι τόσο μικρή, ώστε πρακτικά δεν επιλέγεται ποτέ.

4.8.2 Μέθοδος ιεράρχησης

Η μέθοδος ιεράρχησης(ranking) είναι μια παραλλαγή της μεθόδου επιλογής ρουλέτας με πολύ βελτιωμένα αποτελέσματα. Πρόκειται για μια μέθοδο επιλογής πίεσης, η οποία δίνει πιθανότητα σε ολόκληρο τον πληθυσμό, συγκρίνοντας τις ποιότητες των λύσεων και όχι τις αντικειμενικές τους συναρτήσεις. Έτσι, τα μέλη του πληθυσμού ιεραρχούνται βάσει της αντικειμενικής τους συνάρτησης σε αύξουσα σειρά. Εν συνεχεία, υπολογίζεται η πιθανότητα καθενός μέλους να επιλεχθεί βάσει της εξίσωσης:

$$p(k) = \frac{2 \cdot k}{M \cdot (M+1)} \qquad (4\text{-}2)$$

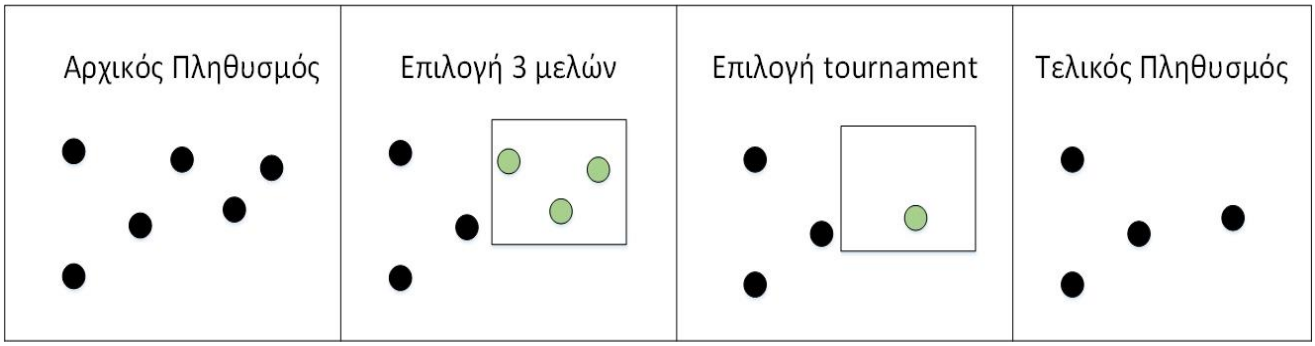
Στην παραπάνω εξίσωση, η σταθερά k αντιπροσωπεύει το k-οστό στοιχεία μετά την ιεράρχηση και ο αριθμός M το μέγεθος του πληθυσμού. Έτσι, το καλύτερο στοιχείο το οποίο βρίσκεται στη θέση M έχει πιθανότητα $\frac{2}{M+1}$ ενώ το χειρότερο στοιχείο που βρίσκεται στη θέση 1 έχει πιθανότητα $\frac{2}{M(M+1)}$.

Το πλεονέκτημα της μεθόδου ιεράρχησης είναι ο καλύτερος έλεγχος της πίεσης επιλογής συγκριτικά με τη μέθοδο ρουλέτας. Το βασικό της μειονέκτημα είναι η ομοιομορφία με την οποία χειρίζεται όλες τις λύσεις, χωρίς να λαμβάνεται υπόψη το μέγεθος του προβλήματος.

4.8.3 Μέθοδος επιλογής τουρνουά

Η μέθοδος επιλογής τουρνουά(Tournament Selection) είναι ένας αποτελεσματικός συνδυασμός των δύο προηγούμενων μεθόδων. Αρχικά, επιλέγονται k υποσύνολα του γενικού πληθυσμού, όπου k είναι ο αριθμός των απογόνων που δημιουργήθηκαν στην τελευταία επανάληψη. Καθένα από αυτά τα υποσύνολα πρέπει να περιέχει τουλάχιστον 2 μέλη του πληθυσμού, ώστε να είναι δυνατή η σύγκριση. Το μέγεθος των υποσυνόλων επηρεάζει την πίεση επιλογής, με την πίεση να αυξάνεται όσο αυξάνεται ο αριθμός των μελών σε κάθε υποσύνολο. Σε κάθε υποσύνολο τα επιμέρους στοιχεία του ‘’διαγωνίζονται’’ για την επιλογή. Για την τελική επιλογή, χρησιμοποιείται το καλύτερο στοιχείο κάθε υποσυνόλου. Η μέθοδος δύναται να λειτουργήσει και αντίστροφα, δηλαδή να επιλεγεί το χειρότερο στοιχείο, όχι για να επιβιώσει, αλλά για να θανατωθεί(Miller and Goldberg 1995).

Ακολουθεί ένα παράδειγμα επιλογής τουρνουά κατά την οποία το μέγεθος του υποσυνόλου είναι 3.



Σχήμα 32 Παράδειγμα Tournament Selection

5 Ορισμός του υπό μελέτη προβλήματος

Το πράσινο πρόβλημα δρομολόγησης στόλου ετερογενών οχημάτων σε αστικό περιβάλλον είναι μια σύνθετη παραλλαγή του απλού προβλήματος δρομολόγησης οχημάτων(VRP). Έτσι, αρχικά παρουσιάζονται οι παραδοχές που ισχύουν και στα δύο προβλήματα και μετά οι επιπλέον παραδοχές του προβλήματος που πραγματεύεται η παρούσα διπλωματική.

Στο πρόβλημα δρομολόγησης οχημάτων μια επιχείρηση διαθέτει ένα στόλο από οχήματα, με τα οποία καλύπτει τη ζήτηση ενός συνόλου πελατών, με προϊόντα αποθηκευμένα σε μια κεντρική αποθήκη. Για το **απλό VRP** ισχύουν οι παρακάτω περιορισμοί:

1. Κάθε πελάτης εξυπηρετείται από ένα και μόνο όχημα και δεν υπάρχει η δυνατότητα πολλαπλών παραδόσεων.
2. Όλα τα οχήματα είναι πανομοιότυπα, έχουν ίδια χαρακτηριστικά και χωρητικότητα.
3. Κάθε όχημα ξεκινάει από την κεντρική αποθήκη, εκτελεί ένα δρομολόγιο και καταλήγει πάλι σε αυτήν στο τέλος του δρομολογίου του. Κανένα όχημα δεν μπορεί να ξαναχρησιμοποιηθεί σε επόμενα δρομολόγια, ακόμα και αν έχει επιστρέψει στην κεντρική αποθήκη, εφόσον έχει ήδη εκτελέσει ένα δρομολόγιο για τη συγκεκριμένη ημέρα.
4. Η ζήτηση κάθε πελάτη είναι μικρότερη ή το πολύ ίση με τη χωρητικότητα των οχημάτων.
5. Το άθροισμα των ζητήσεων όλων των πελατών που θα εξυπηρετηθούν από ένα όχημα δεν μπορεί να ξεπερνάει τη χωρητικότητα του.
6. Δεν υπάρχει δυνατότητα μεταφοράς παραγγελιών από μια ημέρα στην επόμενη. Όλοι οι πελάτες πρέπει να εξυπηρετηθούν την ίδια μέρα και διαφορετικές μέρες θεωρούνται πλήρως ανεξάρτητες μεταξύ τους.

Το απλό πρόβλημα δρομολόγησης οχημάτων χρησιμοποιεί μια πολύ απλουστευμένη κατάσταση της πραγματικότητας, η οποία δεν μπορεί να βρει άμεση εφαρμογή σε καταστάσεις και επιχειρήσεις στον πραγματικό κόσμο. Αρχικά, μια επιχείρηση διαθέτει συνήθως ένα στόλο ετερογενών οχημάτων με διαφορετικά χαρακτηριστικά, ο οποίος μάλιστα αλλάζει με την πάροδο του χρόνου, λόγω αντικατάστασης παλαιότερων οχημάτων, εκσυγχρονισμού του στόλου κλπ. Σε αστικό περιβάλλον τα είδη των οχημάτων που χρησιμοποιούνται είναι λίγα, επειδή οι μεγαλύτεροι σε όγκο τύποι οχημάτων είναι είτε δύσκολο να κυκλοφορήσουν, είτε η κυκλοφορία τους δεσμεύεται από νομικούς περιορισμούς.

Οι επιχειρήσεις διανομής προϊόντων που δραστηριοποιούνται σε αστικό περιβάλλον, διαθέτουν συνήθως ένα μεγάλο σύνολο πελατών, οι οποίοι βρίσκονται σε ιδιαίτερα κοντινές αποστάσεις μεταξύ τους. Ταυτόχρονα, η ζήτηση τους είναι μεταβλητή κάθε μέρα και δεσμεύεται από ποινικές ρήτρες για την μη ικανοποίηση της. Αποτέλεσμα των παραπάνω, είναι η αναγκαστική εξυπηρέτηση των πελατών κάθε μέρα, ακόμη και με χρησιμοποίηση εταιρειών logistics(third party logistics-3PL) όταν ο στόλος της εταιρείας δεν επαρκεί για την κάλυψη των ημερήσιων αναγκών.

Το πρόβλημα πράσινης δρομολόγησης οχημάτων είναι εμπνευσμένο από το περιβάλλον της Αθήνας, το οποίο χαρακτηρίζεται από αυξημένη ποσότητα ρύπων. Για αυτό το λόγο, αποφασίστηκε η υιοθέτηση μιας πράσινης διάστασης στο πρόβλημα. Έτσι, τίθεται ως προτεραιότητα η τήρηση συγκεκριμένου ορίου εκπομπών διοξειδίου του άνθρακα (CO_2) ακόμη και με αύξηση της συνολικά διανυόμενης απόστασης του στόλου των οχημάτων με την αντίστοιχη αύξηση του κόστους.

Το περιβάλλον της Αθήνας χαρακτηρίζεται από ζώνες κίνησης, οι οποίες διέπονται από συγκεκριμένη μέση ταχύτητα αλλά και δεδομένα γεωγραφικά όρια, τα οποία όμως μεταβάλλονται κατά τη διάρκεια της μέρας. Μια ζώνη κίνησης έχει συγκεκριμένες ώρες αιχμής, οι οποίες αυξάνουν σημαντικά το χρόνο εκτέλεσης των δρομολογίων αλλά και τις εκπομπές διοξειδίου του άνθρακα, δημιουργώντας πρόβλημα στην επιχείρηση, ως προς την τήρηση του ορίου που προαναφέρθηκε. Έτσι, οι ζώνες κίνησης επηρεάζουν τις τελικές αποφάσεις ως προς τα δρομολόγια που θα χρησιμοποιηθούν.

Η ημέρα χωρίζεται σε ίσης διάρκειας περιόδους προκειμένου να μπορεί να υπολογιστεί η ταχύτητα και να είναι σταθερή κατά τη διάρκεια μιας περιόδου. Κάθε ημέρα χαρακτηρίζεται από μια βάρδια, η οποία έχει ένα χρόνο έναρξης και ένα χρόνο περάτωσης. Κάθε όχημα της επιχείρησης πρέπει να ξεκινήσει μετά την έναρξη της βάρδιας από την κεντρική αποθήκη, να εξυπηρετήσει όλους τους πελάτες που του έχουν ανατεθεί και να επιστρέψει πίσω στην κεντρική αποθήκη πριν από τη λήξη της βάρδιας, προκειμένου να προετοιμαστεί για την επόμενη. Κάθε όχημα διαθέτει συγκεκριμένο χρόνο φόρτωσης και εκφόρτωσης που εξαρτάται τόσο από την κατηγορία του, όσο και από το άθροισμα των ποσοτήτων των προϊόντων που πρόκειται να φορτωθούν ή να ξεφορτωθούν.

Έτσι, σε αντίθεση με το κλασικό VRP, εκτός από την επιλογή συγκεκριμένων δρομολογίων και την ανάθεση πελατών σε συγκεκριμένα οχήματα, το σύνθετο πρόβλημα λαμβάνει υπόψη του τις περιόδους στις οποίες θα κινηθεί ένα όχημα μέσα στη μέρα για να αποφύγει τις ώρες αιχμής, τις ζώνες κίνησης κάθε δρομολογίου, οι οποίες επηρεάζουν τις συνολικές εκπομπές διοξειδίου του άνθρακα αλλά και το χρόνο εκτέλεσης του δρομολογίου, την αδυναμία των πελατών να παραλάβουν προϊόντα άλλη μέρα με τις αντίστοιχες ποινικές ρήτρες και την υιοθέτηση ορίου για τις συνολικές εκπομπές διοξειδίου του άνθρακα του στόλου των οχημάτων της επιχείρησης.

Συνοπτικά, η παρούσα διπλωματική υιοθετεί τις παρακάτω επιπλέον παραδοχές:

1. Στόλος ετερογενών οχημάτων
2. Ζώνες κίνησης για κάθε ακμή
3. Ώρες αιχμής και ταχύτητες για κάθε ζώνη κίνησης
4. Ανώτατο όριο εκπομπων CO_2
5. Δυνατότητα Υπεργολαβικής ανάθεσης μεταφορών
6. Περιόδοι εκτέλεσης δρομολογίων

6 Μοντελοποίηση Προβλήματος

6.1 Περιγραφή του προβλήματος

Με δεδομένο, ένα **αριθμό N πελατών**, ένα μίγμα **ετερογενών φορτηγών H**, που αποτελείται από **H_1 μικρά φορτηγά** και **H_2 μεγάλα φορτηγά**, θα επισκεφθούν **όλους τους πελάτες ξεκινώντας και καταλήγοντας σε μια κεντρική αποθήκη(depot)**. Κάθε είδος φορτηγού έχει **συγκεκριμένη χωρητικότητα C_h, χρόνο φόρτωσης t_{load}** και **χρόνο εκφόρτωσης t_{unload}**. Οι τιμές για τους **χρόνους αναφέρονται σε πλήρες φορτίο** και η **αναγωγή τους σε μικρότερα φορτία γίνεται γραμμικά**.

Το πρόβλημα μπορεί να απεικονισθεί σε **ένα γράφο G(N,A)** , όπου **N είναι το σύνολο των κόμβων (πελατών)**, ενώ **A είναι το σύνολο των ακμών**(δρόμων) που ενώνουν τους κόμβους. Για κάθε i-j ακμή, **ονομάζεται D_{ij}, η απόσταση που ενώνει τον κόμβο i με τον κόμβο j**. Κάθε πελάτης i έχει **συγκεκριμένη ζήτηση R_i** , η οποία πρέπει να ικανοποιηθεί πλήρως από ακριβώς ένα όχημα. Επιπλέον, **κάθε πελάτης i ανήκει σε μια ζώνη κίνησης Z και χαρακτηρίζεται ως iz**.

Οι ζώνες κίνησης είναι 3 κατηγοριών: **α) Υψηλής Κίνησης** (high traffic zone), **β) Μέτριας κίνησης** (medium traffic zone) και **γ) Χαμηλής κίνησης** (low traffic zone) και **κάθε μια έχει δικές της ώρες αιχμής (Peak hours)** οι οποίες επηρεάζουν την ταχύτητα κίνησης στη ζώνη αυτή.

Ο **χρονικός ορίζοντας** μελέτης(**1 ημέρα**) διαιρείται σε **m χρονικές περιόδους** ούτως ώστε, σε μια δεδομένη χρονική περίοδο η ταχύτητα σε μια συγκεκριμένη διαδρομή να είναι σταθερή, αλλά διαφορετική ανάμεσα σε δύο περιόδους. Έτσι**, κάθε περίοδος k ανήκει στο σύνολο των K περιόδων και χαρακτηρίζεται από το χρόνο έναρξης b_k και το χρόνο τερματισμού της e_k.**

Η ταχύτητα εκτός από την χρονική περίοδο στην οποία εκτελείται το δρομολόγιο, εξαρτάται και από τη ζώνη κίνησης (traffic zone) στην οποία ανήκει κάθε κομμάτι της διαδρομής. Γενικά, **μια διαδρομή ij διασπάται σε B-1 μικρότερες διαδρομές**, οριζόμενες **από B σημεία**, ούτως ώστε **κάθε μικρή διαδρομή $(i=p_0,p_1),(p_1,p_2),(p_2,p_3)\ldots(p_{B-1},j=p_B)$ να ανήκει στην ίδια ζώνη κίνησης και επομένως να έχει σταθερή ταχύτητα** σε συγκεκριμένη χρονική περίοδο μέσα στη μέρα.

Για τον υπολογισμό των εκπομπών διοξειδίου του άνθρακα(CO_2), **χρησιμοποιείται η ταχύτητα του οχήματος h** την χρονική περίοδο k καθώς εκτελεί τη διαδρομή ij σε κάθε μια από τις μικρότερες διαδρομές που χωρίζεται η απόσταση αυτή βάσει των ζωνών κίνησης τις οποίες περιλαμβάνει η διαδρομή ij. Ο ακριβής υπολογισμός γίνεται βάσει ενός μοντέλου ελαχιστοποίησης των εκπομπών για το πρόβλημα δρομολόγησης οχημάτων{Figliozzi, 2010 #192}. Ο τύπος υπολογισμού για μια διαδρομή i-j με σταθερή ταχύτητα V_{ij} είναι:

$$Vol_{co2} = (a_0 + a_1 V_{ij} + a_2 V_{ij}^3 + \frac{a_3}{V_{ij}^2}) d_{ij} \qquad (6\text{-}1)$$

Οι σταθερές (α_o,α_1,α_2,α_3) παίρνουν τις τιμές (1,576;-17,6;0,00117;36,067) για το συγκεκριμένο πρόβλημα που μελετάται και με χρήση πολυωνυμικών όρων μπορούν να υπολογιστούν και για άλλους τύπους οχημάτων.

Το πρόβλημα περιλαμβάνει **δύο είδη αποφάσεων**:i) **αν θα χρησιμοποιηθεί το όχημα ή όχι** ii) **ποια διαδρομή θα ακολουθήσει ένα όχημα για να επισκεφθεί τους πελάτες** αν τελικά επιλεγεί στη λύση και εξ'αυτών έπονται ως υπολογιζόμενα μεγέθη **η ώρα αναχώρησης και άφιξης** κάθε οχήματος της λύσης σε κάθε κόμβο του δικτύου και **οι αποστάσεις που πρέπει να διανυθούν σε κάθε ακμή** σε κάθε μια από τις χρονικές περιόδους.

Στόχος του προβλήματος είναι η **ελαχιστοποίηση της συνολικά διανυόμενης απόστασης**, ενώ ταυτόχρονα ο **συνολικός όγκος των εκπομπών διοξειδίου του άνθρακα να βρίσκεται κάτω από το όριο** το οποίο τίθεται από την εταιρεία.

6.2 Πίνακας Μεγεθών

Παρουσιάζεται ο πίνακας με όλα τα μεγέθη που χρησιμοποιούνται στο μοντέλο:

Κόμβος 0	Κεντρική Αποθήκη
C={1,2....,N+1}	Σύνολο Πελατών
C_m	Σύνολο Μεσαίων Πελατών
C_b	Σύνολο Μεγάλων Πελατών
R	Ζήτηση
R_m	Ζήτηση Μεσαίων Πελατών
R_b	Ζήτηση Μεγάλων Πελατών
Z	Ζώνη Κίνησης
Iz	Κατάστημα i που ανήκει στη ζώνη z
V	Όχημα
V_s	Μικρό όχημα
V_m	Μεγάλο Όχημα
C_s	Ποσοστό Κόστους οχήματος υπεργολαβίας σε σχέση με το κόστος ιδιόκτητου οχήματος
t_{load}	Χρόνος φόρτωσης οχήματος
t_{unload}	Χρόνος εκφόρτωσης οχήματος
C_h	Χωρητικότητα Οχήματος h
g_i	Χρόνος εξυπηρέτησης πελάτη i
X_{ij}	Το δρομολόγιο ij εκτελείται ή οχι
Y_{ijh}	Το δρομολόγιο ij εκτελείται ή οχι από το όχημα h
X_{ijkh}	Το δρομολόγιο ij εκτελείται ή οχι από το όχημα h την περίοδο k
D_{ijkh}	Η απόσταση που διανύεται στο δρομολόγιο ij από το όχημα h την περίοδο k
τ_{ijkh}	Ο χρόνος που χρειάζεται για να εκτελεσθεί το δρομολόγιο ij από το όχημα h την περίοδο k
$\tau^{(z)}_{p_{(n)}p_{(n+1)}kh}$	Ο χρόνος εκτέλεσης της διαδρομής p(n)p(n+1) ου οχήματος h, την περίοδο k στη ζώνη κίνησης z
$VP^{(z)}_{p_{(n)}p_{(n+1)}kh}$	Η σταθερή ταχύτητα του οχήματος h, την περίοδο k η οποία δεν ανήκει στις ώρες αιχμής, κατά τη διαδρομή ij στη ζώνη κίνησης z
$VR^{(z)}_{p_{(n)}p_{(n+1)}kh}$	Η σταθερή ταχύτητα του οχήματος h, την περίοδο k η οποία ανήκει στις ώρες αιχμής, κατά τη διαδρομή ij στη ζώνη κίνησης z
l_i	Χρόνος αποχώρησης από τον κόμβο i
a_i	Χρόνος άφιξης στον κόμβο i
b_k	Χρόνος έναρξης της περίοδου k
e_k	Χρόνος λήξης της περίοδου k
α_i	Σταθερά υπολογισμού εκπομπών CO_2
Vol_{ijkh}	Όγκος εκπομπών CO_2 του οχήματος h, την περίοδο k κατά την εκτέλεση της διαδρομής ij.
$Co2_{lim}$	Όριο εκπομπών διοξειδίου του άνθρακα

Πινακας 4 Ορολογία μαθηματικού μοντέλου

6.3 Μεταβλητές Απόφασης

1. X_{ij}: Δυαδική Μεταβλητή = {1, αν εκτελείται το δρομολογιο ij, αλλιώς 0}
2. Y_{ijh}: Δυαδική Μεταβλητή = {1, αν εκτελείται το δρομολογιο ij από το όχημα h, αλλιώς 0}
3. X_{ijkh}: Δυαδική Μεταβλητή = {1, αν εκτελείται το δρομολογιο ij από το όχημα h την περίοδο k, αλλιώς 0}

6.4 Βοηθητικές μεταβλητές

1. d_{ijkh} : Συνεχής Μεταβλητή που δείχνει τη διανυόμενη απόσταση κατά το δρομολόγιο ij από το όχημα h την περίοδο k
2. τ_{ijkh} : Συνεχής μεταβλητή που δείχνει το χρόνο εκτέλεσης της απόστασης d_{ijkh}
3. l_i : Συνεχής Μεταβλητή που δείχνει τον χρόνο αναχώρησης από τον κόμβο i
4. a_i : Συνεχής Μεταβλητή που δείχνει το χρόνο άφιξης στον κόμβο i

6.5 Αντικειμενική συνάρτηση

Στόχος του προβλήματος είναι η ελαχιστοποίηση της συνολικά διανυόμενης απόστασης από όλα τα οχήματα, τόσο του στόλους της επιχείρησης όσο και των οχημάτων που χρησιμοποιούνται υπεργολαβικά στη διάρκεια όλων των περιόδων, κατά την εξυπηρέτηση όλων των πελατών.

$$\min(F) = \sum_{h=1}^{q}\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} d_{ijhk} + C_s \cdot \sum_{h=1}^{o}\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} d_{ijhk} \qquad (6\text{-}2)$$

6.6 Περιορισμοί

$$\sum_{j=0}^{n} X_{ij} = 1 \qquad \forall\, i \in N' \qquad (6\text{-}3)$$

$$\sum_{i=0}^{n} X_{ij} = 1 \qquad \forall\, j \in N' \qquad (6\text{-}4)$$

$$Xij = \sum_{h=1}^{q} Y_{ijh} \qquad \forall\, i,j \in A \qquad (6\text{-}5)$$

$$Y_{ijh} \geq X_{ijhk} \qquad \forall (\, i,j) \in A, k \in K, h \in H \qquad (6\text{-}6)$$

$$Y_{ijh} \leq \sum_{k=1}^{m} X_{ijhk} \qquad \forall\, i,j \in A, h \in H \qquad (6\text{-}7)$$

$$\sum_{i=0:i\neq j}^{n} Y_{ijh} = \sum_{i=0:i\neq j}^{n} Y_{jih} \qquad \forall\, j \in N, h\ \in H \qquad (6\text{-}8)$$

$$\sum_{j=1}^{n} Y_{0jh} \leq 1 \qquad \forall\, h \in H \qquad (6\text{-}9)$$

$$d_{ijkh}\ \leq D_{ij} \cdot X_{ijkh} \qquad \forall (\, i,j) \in A, k \in K, h \in H \qquad (6\text{-}10)$$

$$\sum_{k=1}^{m}\sum_{h=1}^{q} d_{ijkh} = X_{ij} \cdot D_{ij} \qquad \forall\, i,j \in A \qquad (6\text{-}11)$$

$$\sum_{(i,j)\in A}^{n} \tau_{ijkh} \leq e_k - b_k \qquad \forall\, k \in K, h \in H \qquad (6\text{-}12)$$

$$l_i \leq e_k - \tau_{ijkh} + M \cdot \left(1 - X_{ijkh}\right) \qquad \forall (\, i,j) \in A, k \in K, h \in H \qquad (6\text{-}13)$$

$$a_j \geq b_k +\ \tau_{ijkh} - M \cdot \left(1 - X_{ijkh}\right) \qquad \forall (\, i,j) \in A, k \in K, h \in H \qquad (6\text{-}14)$$

$$a_j \geq l_i + \sum_{k=1}^{m}\sum_{h=1}^{q} \tau_{ijkh} - M \cdot \left(1 - X_{ij}\right) \qquad \forall\, (i,j) \in A \qquad (6\text{-}15)$$

$$g_i = \mathrm{t}_{\mathrm{unload}} \cdot \frac{\mathrm{R}_i}{C_h} \qquad \forall\, h \in H \qquad (6\text{-}16)$$

$$a_i + g_i \leq l_i \qquad \forall\, i \in N' \qquad (6\text{-}17)$$

$$a_0 \leq e_m \qquad (6\text{-}18)$$

$$\sum_{j=1}^{n}\sum_{i=0}^{n} R_j \cdot Y_{ijh} \leq C_h \qquad \forall\, h \in H \qquad (6\text{-}19)$$

$$d_{ijkh} = \sum_{n=0}^{B-1} d_{p_{(n)}p_{(n+1)}kh} \qquad \forall\, h \in H, k \in K \qquad (6\text{-}20)$$

$$\tau_{ijkh} = \sum_{n=0}^{B-1} \tau^{(z)}_{p_{(n)}p_{(n+1)}kh} \qquad \forall\, h \in H, k \in K, z \in Z \qquad (6\text{-}21)$$

$$\tau^{(z)}_{p_{(n)}p_{(n+1)}kh} = \frac{d_{p_{(n)}p_{(n+1)}kh}}{V^{(z)}_{p_{(n)}p_{(n+1)}kh}} \qquad \forall (\, p_n, p_{n+1}) \in B, k \in K, h \in H \qquad (6\text{-}22)$$

$$V^{(z)}_{p_{(n)}p_{(n+1)}kh} = \begin{cases} VP^{(z)}_{p_{(n)}p_{(n+1)}kh} & \text{αν } k \text{ δεν ανήκει στα peak hours} \\ VR^{(z)}_{p_{(n)}p_{(n+1)}kh} & \text{αν } k \text{ ανήκει στα peak hours} \end{cases} \qquad \forall \left(p_{(n)}, p_{(n+1)}\right) \in B, k \in K, h \in H \qquad (6\text{-}23)$$

$$Vol_{ijkh} = \sum_{n=0}^{B-1} (a_0 + a_1 V^{(z)}_{p_{(n)}p_{(n+1)}kh} + a_2 \left(V^{(z)}_{p_{(n)}p_{(n+1)}kh}\right)^3 + a_3 \frac{1}{(V^{(z)}_{p_{(n)}p_{(n+1)}kh})^2}) d_{p_{(n)}p_{(n+1)}kh} \qquad (6\text{-}24)$$

$$\sum_{j=0}^{n}\sum_{i=0}^{n}\sum_{k=1}^{m}\sum_{h=1}^{q} Vol_{ijkh} \cdot X_{ijkh} \leq \mathrm{Co2}_{lim} \qquad (6\text{-}25)$$

$$l_0 \geq b_0 + \mathrm{t}_{\mathrm{load}} \cdot (\sum_{j=1}^{n}\sum_{i=0}^{n} R_j \cdot Y_{ijh})/C_h \qquad \forall\, h \in H \qquad (6\text{-}26)$$

6.7 Επεξήγηση Περιορισμών

Ο περιορισμός **(6-3)** απαιτεί το άθροισμα των οχημάτων που ξεκινούν από ένα κόμβο i να είναι ακριβώς 1 για κάθε i διάφορο του μηδενός. Έτσι, για κάθε πελάτη μόνο ένα όχημα πρέπει να ξεκινά από αυτόν, αφού κάθε πελάτης εξυπηρετείται από ακριβώς ένα όχημα.

Ο περιορισμός **(6-4)** απαιτεί το άθροισμα των οχημάτων που καταλήγουν σε ένα κόμβο j να είναι ακριβώς 1 για κάθε j διάφορο του μηδενός. Έτσι, για κάθε πελάτη μόνο ένα όχημα πρέπει να καταλήγει σε αυτόν, αφού κάθε πελάτης εξυπηρετείται από ακριβώς ένα όχημα.

Οι περιορισμοί (6-3) και (6-4) διασφαλίζουν οτι κάθε πελάτη θα επισκεφθεί ακριβώς ένα όχημα.

Ο περιορισμός **(6-5)** δηλώνει οτι κάθε δρομολόγιο ij,εαν επιλεγεί μπορεί να γίνει από ένα και μόνο όχημα. Συγκεκριμένα, πρέπει το άθροισμα των οχημάτων H που εκτελούν το δρομολόγιο ij να είναι 1, αν το δρομολόγιο εκτελείται δηλαδή X_{ij}=1, ή να είναι 0, αν το δρομολόγιο X_{ij}=0.

Ο περιορισμός **(6-6)** υποχρεώνει το σύνολο των δρομολογιών κάθε οχήματος να είναι μεγαλύτερο από το σύνολο των δρομολογίων μιας περιόδου για το συγκεκριμένο όχημα. Με άλλα λόγια κάθε όχημα πραγματοποιεί τα δρομολόγια του ε συγκεκριμένες περιόδους το καθένα. Έτσι, αν ένα δρομολόγιο εκτελείται σε μια συγκεκριμένη περίοδο k από ένα όχημα h(όχι μόνο σε αυτή αναγκαστικά), τότε το δρομολόγιο πρέπει να εκτελείται ολόκληρο από το ίδιο όχημα, ενώ αν δεν εκτελείται το δρομολόγιο σε αυτή την περίοδο k, το δρομολόγιο μπορεί είτε να εκτελείται σε άλλη περίοδο k είτε να μην εκτελείται καθόλου.

Ο περιορισμός **(6-7)** υποχρεώνει το σύνολο των δρομολογίων κάθε οχήματος να είναι μικρότερο από το σύνολο των δρομολογίων για όλες τις περιόδους για το συγκεκριμένο όχημα. Με άλλα λόγια κάθε όχημα πρέπει να αποτελείται από τουλάχιστον ένα δρομολόγιο σε διαφορετικές περιόδους. Έτσι, αν το δρομολόγιο εκτελείται, δηλαδή Y_{ijh}=1, τότε πρέπει το άθροισμα των δρομολογίων που εκτελούνται σε διαφορετικές περιόδους να είναι τουλάχιστον 1, ενώ αντίθετα αν το δρομολόγιο δεν εκτελείται, δηλαδή Y_{ijh}=0, τότε πρέπει το άθροισμα των δρομολογίων που εκτελούνται σε διαφορετικές περιόδους να είναι τουλάχιστον 0.

Οι περιορισμοί (6-6) και (6-7) επιβάλλουν τη συσχέτιση μεταξύ των μεταβλητών Y_{ijh} και X_{ijkh} .

Ο περιορισμός **(6-8)** επιβάλλει την είσοδο και την έξοδο κάθε οχήματος από κάθε κόμβο ίδιο αριθμό φορών. Συγκεκριμένα, αν ένα όχημα εισέλθει σε ένα κόμβο, δηλαδή αν πραγματοποιηθεί το δρομολόγιο, πρέπει το όχημα να εξέλθει από τον κόμβο, δηλαδή να συνεχίσει το δρομολόγιο του είτε προς άλλο πελάτη είτε προς την κεντρική αποθήκη.

Ο περιορισμός **(6-9)** απαιτεί κάθε όχημα να φεύγει από την κεντρική αποθήκη το πολύ μια φορά. Έτσι, αν κάποιο όχημα χρησιμοποιηθεί σε ένα συγκεκριμένο δρομολόγιο, δηλαδή Y0jh=1, τότε το όχημα αυτό όταν επιστρέψει μετά το πέρας του δρομολογίου του στην κεντρική αποθήκη, δεν μπορεί να χρησιμοποιηθεί ξανά για την εκτέλεση διαφορετικού δρομολογίου.

Ο περιορισμός **(6-10)** αναγκάζει τη συνεχή μεταβλητή d_{ijkh}, η οποία εκφράζει τη διανυόμενη απόσταση ενός δρομολογίου ij από ένα όχημα h σε μια χρονική περίοδο k, να είναι μικρότερη από τη συνολική απόσταση του δρομολογίου ij. Έτσι, αν το X_{ijkh} γίνει 0 δηλαδή το δρομολόγιο δεν εκτελεσθεί τη συγκεκριμένη περίοδο, η απόσταση d_{ijkh} που θα εκτελεσθεί εκείνη την περίοδο πρέπει να είναι επίσης 0. Αντίθετα, αν το δρομολόγιο εκτελεσθεί τη συγκεκριμένη περίοδο, η απόσταση d_{ijkh} πρέπει να είναι μικρότερη ή ίση της συνολικής απόστασης D_{ij} , η οποία εκτελείται σε τουλάχιστον μια περίοδο.

Ο περιορισμός **(6-11)** επιβάλλει οτι αν επιλεγεί το δρομολόγιο, δηλαδή X_{ij}=1, πρέπει να διανυθεί ολόκληρη η απόσταση D_{ij}, δηλαδή η συνεχής μεταβλητή d_{ijkh} για κάθε περίοδο και για κάθε όχημα πρέπει να είναι ίση με την απόσταση της ακμής ij, δηλαδή με την D_{ij}. Αν το δρομολόγιο δεν επιλεγεί, δηλαδή X_{ij}=0, τότε και η συνεχής μεταβλητή d_{ijkh} για κάθε περίοδο και για κάθε όχημα πρέπει να είναι ίση με το 0.

Ο περιορισμός **(6-12)** εξασφαλίζει οτι ο χρόνος κατά την εκτέλεση μιας διαδρομής ij από ένα όχημα, σε μια μόνο περίοδο είναι μικρότερος από τη διαφορά της έναρξης της περιόδου από την λήξη της. Με άλλα λόγια, ο χρόνος αυτός είναι μικρότερος από τη χρονική διάρκεια της περιόδου.

Ο περιορισμός **(6-13)** επιβάλλει οτι αν κάποιο δρομολόγιο ij πραγματοποιηθεί την περίοδο k, δηλαδή (X_{ijkh}=1), τότε ο χρόνος αναχώρησης από τον κόμβο i πρέπει να γίνει πριν από το τέλος της περιόδου k μείον το χρόνο εκτέλεσης της διαδρομής ij. Αν το δρομολόγιο δεν εκτελεσθεί, δηλαδή (X_{ijkh}=0), ο περιορισμός συνεχίζει και ισχύει λόγω του μεγάλου αριθμού M που χρησιμοποιείται στην εξίσωση.

Ο περιορισμός **(6-14)** δηλώνει οτι αν κάποιο δρομολόγιο ij πραγματοποιηθεί την περίοδο k, δηλαδή (X_{ijkh}=1), τότε ο χρόνος άφιξης στον κόμβο j πρέπει να είναι μεγαλύτερος από το άθροισμα του χρόνου έναρξης της περιόδου και του χρόνου εκτέλεσης της διαδρομής ij. Αν το δρομολόγιο δεν εκτελεσθεί, δηλαδή (X_{ijkh}=0), ο περιορισμός συνεχίζει και ισχύει λόγω του μεγάλου αριθμού M που χρησιμοποιείται στην εξίσωση.

Ο περιορισμός **(6-15)** είναι ένας διαζευκτικός περιορισμός, ο οποίος απαιτεί η άφιξη στον κόμβο j να είναι μεγαλύτερη από την αναχώρηση από τον κόμβο i συν το άθροισμα των χρόνων εκτέλεσης του δρομολογίου από το όχημα σε κάθε περίοδο, αν επιλεγεί το δρομολόγιο, δηλαδή αν X_{ijkh}=1. Αν το δρομολόγιο δεν επιλεγεί,δηλαδή αν X_{ijkh}=1, τότε ο περιορισμός συνεχίζει και ισχύει λόγω του μεγάλου αριθμου M που χρησιμοποιείται στην εξίσωση.

Ο περιορισμός **(6-16)** υπολογίζει το χρόνο εξυπηρέτησης κάθε πελάτη, ως το γινόμενο του χρόνου εκφόρτωσης για πλήρες φορτίο t_{unload} επί το ποσοστό της ζήτησης του πελάτη σε σχέση με τη χωρητικότητα του οχήματος.

Ο περιορισμός **(6-17)** επιβάλλει οτι ο χρόνος αναχώρησης κάθε οχήματος πρέπει να είναι μεγαλύτερος από το άθροισμα του χρόνου άφιξης στον κόμβο i και του χρόνου εξυπηρέτησης του στον ίδιο κόμβο. Με άλλα λόγια, κάθε όχημα πρέπει να αναχωρήσει μετά από την άφιξη και την εξυπηρέτηση του από κάθε κόμβο i που επισκέπτεται.

Ο περιορισμός **(6-18)** εξασφαλίζει οτι ο χρόνος επιστροφής κάθε οχήματος στην κεντρική αποθήκη είναι μικρότερος από το τέλος της τελευταίας χρονικής περιόδου. Με άλλα λόγια, κάθε όχημα πρέπει να έχει επιστρέψει πίσω στην κεντρική αποθήκη πριν το τέλος της βάρδιας, για να είναι έτοιμο να χρησιμοποιηθεί και την επόμενη μέρα.

Ο περιορισμός **(6-19)** αναγκάζει το άθροισμα των ζητήσεων κάθε πελάτη που εξυπηρετείται από ένα συγκεκριμένο όχημα να είναι μικρότερο από τη συνολική χωρητικότητα του οχήματος αυτού, για κάθε όχημα του στόλου της επιχείρησης.

Ο περιορισμός **(6-20)** υπολογίζει τη συνολική απόσταση d_{ijkh} ως το άθροισμα των στοιχειωδών αποστάσεων που την απαρτίζουν, και στις οποίες η ταχύτητα κίνησης είναι σταθερή.

Ο περιορισμός **(6-21)** αναλογικά με τον προηγούμενο περιορισμό υπολογίζει το συνολικό χρόνο εκτέλεσης του δρομολογίου d_{ijkh}, ως το άθροισμα των στοιχειωδών χρόνων εκτέλεσης των στοιχειωδών αποστάσεων που απαρτίζουν τη συνολική διαδρομή και στις οποίες η ταχύτητα κίνησης είναι σταθερή.

Ο περιορισμός **(6-22)** υπολογίζει το χρόνο εκτέλεσης μιας στοιχειώδους απόστασης, ως το πηλίκο της στοιχειώδους απόστασης προς την ταχύτητα που επικρατεί στην απόσταση αυτή και είναι σταθερή. Όλες οι μονάδες που χρησιμοποιούνται στα μεγέθη βρίσκονται στο κατάλληλο σύστημα μονάδων.

Ο περιορισμός **(6-23)** δηλώνει οτι η ταχύτητα ενός οχήματος για κάθε μια από τις μικρότερες διαδρομές που ορίστηκαν, μπορεί να πάρει δύο τιμές, ανάλογα με το αν η χρονική περίοδος k εκτέλεσης του δρομολογεί ανήκει ή όχι στις ώρες αιχμής(peak hours).

Ο περιορισμός **(6-24)** υπολογίζει το συνολικό όγκο εκπομπών για μια διαδρομή i-j, από ένα όχημα h, μια χρονική περίοδο k, ώς το άθροισμα των εκπομπών για κάθε μια από τις μικρότερες διαδρομές που απαρτίζουν τη συνολική διαδρομή i-j. Ο όγκος των εκπομπών εξαρτάται από την ταχύτητα και τη διανυόμενη απόσταση και οι $\alpha_0,\alpha_1,\alpha_2,\alpha_3$ είναι σταθερές για κάθε είδος οχήματος.

Ο περιορισμός **(6-25)** επιβάλλει το άθροισμα των εκπομπών όλων των οχημάτων, για όλες τις χρονικές περιόδους και για κάθε διαδρομή που έχει επιλεχθεί να μην ξεπερνά το όριο, το οποίο έχει τεθεί από την εταιρεία.

Ο περιορισμός **(6-26)** υποχρεώνει ο νωρίτερος χρόνος αναχώρησης από την κεντρική αποθήκη να είναι μεγαλύτερος από το άθροισμα του χρόνου έναρξης της πρώτης περιόδου και του χρόνου φόρτωσης του εμπορεύματος των πελατών που θα εξυπηρετήσει το συγκεκριμένο όχημα.

7 Προτεινόμενη μέθοδος επίλυσης

7.1 Περιγραφή του συστήματος

Με στόχο την επίλυση του προτεινόμενου προβλήματος δρομολόγησης όπως αυτό ορίστηκε στο κεφάλαιο 6 , αναπτύχθηκε μεταευρετικός αλγόριθμος αποτελούμενος από 2 εμφωλευμένους γενετικούς αλγορίθμους, με τον εξωτερικό να διαχειρίζεται το πρόβλημα ανάθεσης πελατών σε τύπους οχημάτων και τη διαχείρηση των διαδράσεων ενώ ο εσωτερικός την καθ΄αυτό δρομολόγηση των οχημάτων. Με αυτό το σκοπό αναπτύχθηκε ένα σύστημα επίλυσης του προβλήματος δρομολόγησης οχημάτων για την εξυπηρέτηση ενός συνόλου πελατών. Το σύστημα δέχεται ως είσοδο από το χρήστη ή υπό τη μορφή αρχείου:

1. Τα χαρακτηριστικά των πελατών:
    a. Ακριβής γεωγραφική τοποθεσία.
    b. Ζήτηση.
2. Τα χαρακτηριστικά των οχημάτων:
    a. Κατηγορίες οχημάτων.
    b. Σταθερές για τον υπολογισμό των εκπομπών διοξειδίου του άνθρακα.
    c. Αριθμό οχημάτων κάθε κατηγορίας
3. Χαρακτηριστικά των ζωνών κίνησης:
    a. Γεωγραφικά όρια.
    b. Χαρακτηρισμός της ζώνης(υψηλής,μεσαίας,μικρής).
    c. Ώρες αιχμής.
4. Χαρακτηριστικά της περιόδου μελέτης:
    a. Χρόνος έναρξης Βάρδιας.
    b. Χρόνος λήξης βάρδιας.

Το σύστημα έχει ως έξοδο μια λίστα με το δρομολόγιο κάθε οχήματος, δηλαδή τους πελάτες καθώς και τη σειρά με την οποία θα τους εξυπηρετήσει και γραφική αναπαράσταση της λύσης.

Σχήμα 33 Είσοδος-Έξοδος συστήματος

7.2 Προτεινόμενος αλγόριθμος

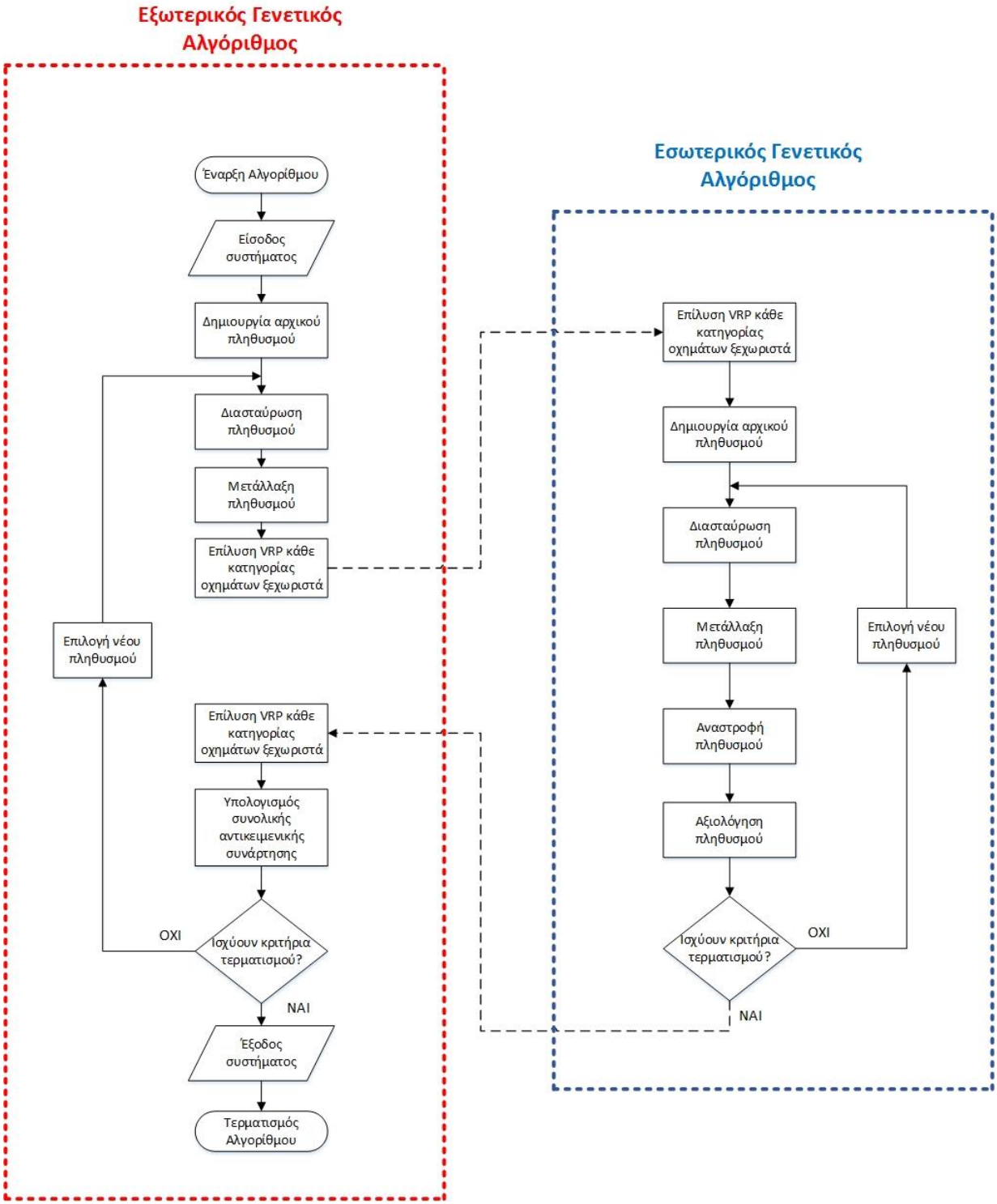
Ο βασικός αλγόριθμος στηρίζεται στη συνεργασία δύο γενετικών αλγορίθμων, έναν εσωτερικό και ένα εξωτερικό, οι οποίοι συνεργάζονται μεταξύ τους. Ο εξωτερικός γενετικός τροφοδοτεί σε κάθε επανάληψη τον εσωτερικό γενετικό με ένα αριθμό συγκεκριμένων πελατών για κάθε κατηγορία οχήματος. Κάθε πελάτης ανήκει αναγκαστικά σε μια και μόνο κατηγορία οχημάτων βάσει των παραδοχών που περιγράφηκαν νωρίτερα. Ο εσωτερικός γενετικός για το δεδομένο σύνολο πελατών που δέχεται κάθε φορά λύνει το πρόβλημα δρομολόγησης οχημάτων κάθε κατηγορίας. Το αποτέλεσμα του εσωτερικού γενετικού για κάθε κατηγορία οχημάτων τροφοδοτείται στον εξωτερικό γενετικό, ο οποίος αξιολογεί συνολικά τη διανυόμενη απόσταση όλου του συστήματος, τις εκπομπές διοξειδίου του άνθρακα και τον αριθμό των χρησιμοποιούμενων οχημάτων και προσπαθεί να βελτιώσει τις παραγόμενες λύσεις ξεκινώντας ένα νέο κύκλο επαναλήψεων, μέχρι τα κριτήρια τερματισμού να ικανοποιηθούν.

Τα βήματα του αλγορίθμου μπορούν να συνοψιστούν ως εξής:

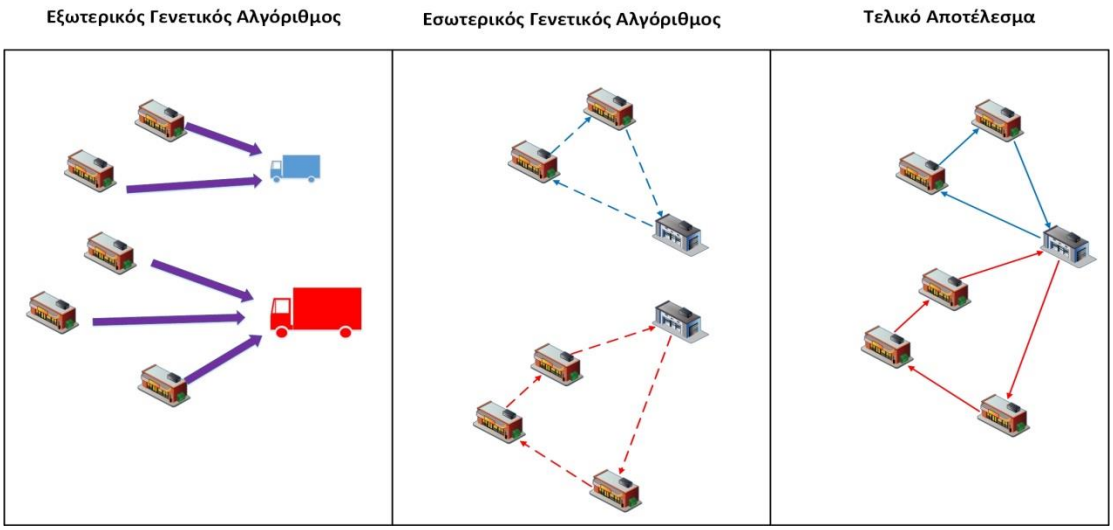
1. Εισάγονται στο σύστημα όλα τα απαιτούμενα στοιχεία.
2. Οι πελάτες χωρίζονται σε κατηγορίες και δημιουργείται ο αρχικός πληθυσμός του εξωτερικού αλγορίθμου.
3. Πραγματοποιείται διασταύρωση στον πληθυσμό του εξωτερικού γενετικού αλγορίθμου.
4. Πραγματοποιείται μετάλλαξη στον πληθυσμό του εξωτερικού γενετικού αλγορίθμου.
5. Κάθε κατηγορία οχήματος εισάγεται στον εσωτερικό γενετικό αλγόριθμο ξεχωριστά και επιλύεται:
    a. Δημιουργείται για το εκάστοτε σύνολο πελατών ένας αρχικός πληθυσμός με βάση ευρετικές μεθόδους.
    b. Πραγματοποιείται διασταύρωση στον πληθυσμό του εσωτερικού γενετικού αλγορίθμου.
    c. Πραγματοποιείται μετάλλαξη στον πληθυσμό του εσωτερικού γενετικού αλγορίθμου.
    d. Πραγματοποιείται αναστροφή στον πληθυσμό του εσωτερικού γενετικού αλγορίθμου.
    e. Ο πληθυσμός του εσωτερικού γενετικού αλγορίθμου αξιολογείται.
    f. Όσο τα κριτήρια τερματισμού δεν ισχύουν, ο αλγόριθμος επιλέγει το νέο αρχικό πληθυσμό και επιστρέφει στο βήμα 5.
    g. Ο αλγόριθμος επιστρέφει την τελική λύση κάθε κατηγορίας και πηγαίνει στο βήμα 6.
6. Ο πληθυσμός του εξωτερικού γενετικού αλγορίθμου αξιολογείται.
7. Όσο τα κριτήρια τερματισμού δεν ισχύουν, ο αλγόριθμος επιλέγει το νέο αρχικό πληθυσμό και επιστρέφει στο βήμα 3.
8. Ο αλγόριθμος δημιουργεί γραφική αναπαράσταση της τελικής λύσης και τερματίζει.

Αλγόριθμος 11 Προτεινόμενος Αλγόριθμος

Ακολουθεί το διάγραμμα ροής του προτεινόμενου αλγορίθμου:



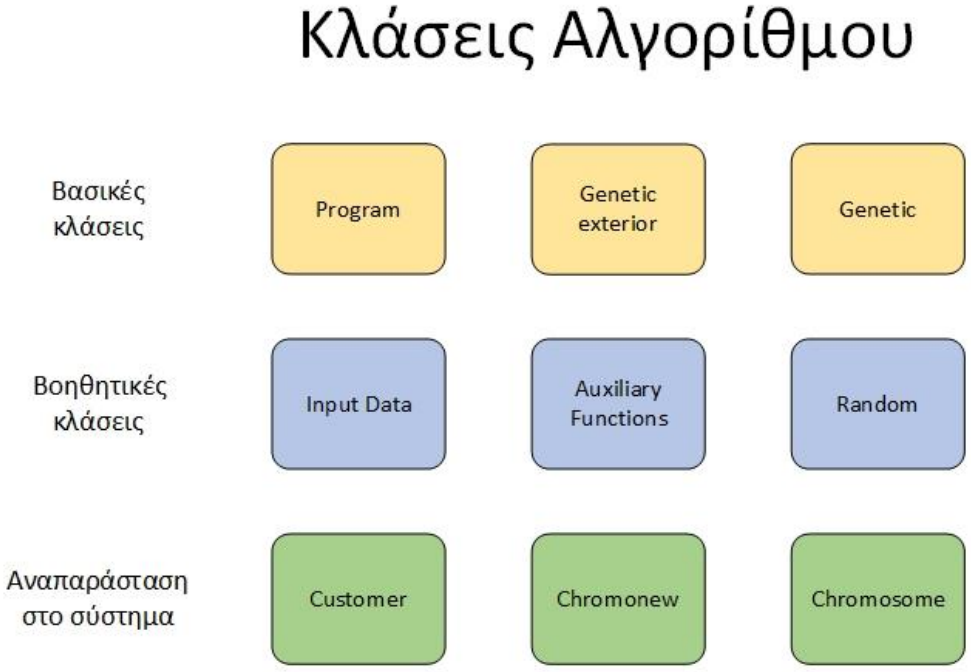
Σχήμα 34 Διάγραμμα ροής προτεινόμενου αλγορίθμου



Σχήμα 35 Παράδειγμα Χρήσης Υβριδικού Αλγορίθμου

7.3 Δομικά Στοιχεία Αλγορίθμου

Για την σωστή υλοποίηση του αλγορίθμου πρώτο βήμα αποτελεί η ανάλυση του συστήματος και ο καθορισμός των απαραίτητων κλάσεων. Κάθε κλάση έχει ιδιαίτερα χαρακτηριστικά και είναι καίριο κομμάτι του συστήματος. Ο συνδυασμός όλων των κλάσεων οδηγεί στην τελική υλοποίηση του αλγορίθμου. Έτσι, κατασκευάσθηκαν οι κάτωθι κλάσεις :



Σχήμα 36 Αναπαράσταση χρησιμοποιούμενων κλάσεων

1. Η κλάση **Program** αποτελεί το βασικό μέρος του κύριου προγράμματος, το οποίο καλεί και χρησιμοποιεί όλες τις υπόλοιπες κλάσεις με το σωστό τρόπο και τη σωστή σειρά.

2. Η κλάση **Genetic exterior** αποτελεί την υλοποίηση του εξωτερικού γενετικού αλγορίθμου. Δέχεται ως είσοδο ένα πίνακα πελατών και τα βασικά δεδομένα του προβλήματος και τους χωρίζει βάσει κάποιων μεθόδων σε κατηγορίες εξυπηρέτησης από τα οχήματα και προσπαθεί να βελτιώσει τη συνολική λύση. Ο εξωτερικός γενετικός αλγόριθμος καλεί τον εσωτερικό γενετικό για κάθε μια κατηγορία οχημάτων ξεχωριστά, όπως έχει περιγραφεί νωρίτερα.

3. Η κλάση **Genetic** αποτελεί την υλοποίηση του εσωτερικού γενετικού αλγορίθμου. Δέχεται ως είσοδο ένα υποσύνολο των συνολικών πελατών, το οποίο ορίζεται από τον εξωτερικό γενετικό αλγόριθμο, και επιλύει με συγκεκριμένες μεθόδους το πρόβλημα δρομολόγησης οχημάτων των πελατών αυτών.

4. Η κλάση **Input Data** είναι μια κλάση εισόδου δεδομένων στο σύστημα, είτε με τη μορφή ενός αρχείου δεδομένων, αντίστοιχο των προβλημάτων αναφοράς, είτε με τη μορφή εισόδου από το πληκτρολόγιο. Τα δεδομένα περιλαμβάνουν τα χαρακτηριστικά των πελατών, τα χαρακτηριστικά των οχημάτων, τα χαρακτηριστικά των ζωνών κίνησης και τα χαρακτηριστικά κάθε βάρδιας.

5. Η κλάση **Auxiliary Functions** είναι μια βοηθητική κλάση, η οποία περιέχει εργαλεία, τα οποία βοηθούν στην επίλυση του προβλήματος, όπως οι διάφορες μέθοδοι διασταύρωσης, ο υπολογισμός αποστάσεων των πελατών, ο υπολογισμός των πολικών γονιών, η κωδικοποίηση και αποκωδικοποίηση δεδομένων κλπ.

6. Η κλάση **Random** είναι μια κλάση παραγωγής τυχαίων αριθμών. Η χρησιμοποίηση ξεχωριστής κλάσης για την δημιουργία τυχαίων αριθμών βοηθάει στη χρήση διαφορετικών αριθμών σε κάθε επανάληψη, γεγονός απαραίτητο για τη σωστή λειτουργία των γενετικών αλγορίθμων.

7. Η κλάση **Customer** αναπαριστά έναν πελάτη ή την κεντρική αποθήκη και περιλαμβάνει τα χαρακτηριστικά του/της, όπως τις συντεταγμένες του και τη ζήτησή του.

8. Η κλάση **Chromosome** αναπαριστά το χρωμόσωμα του εξωτερικού γενετικού αλγορίθμου, δηλαδή μια κωδικοποιημένη λύση σε μορφή χρωμοσώματος.

9. Η κλάση **Chromonew** αναπαριστά το χρωμόσωμα του εσωτερικού γενετικού αλγορίθμου, δηλαδή μια κωδικοποιημένη λύση σε μορφή χρωμοσώματος.

8 Αλγόριθμος που αναπτύχθηκε

8.1 Προγραμματιστικό Περιβάλλον

Η υλοποίηση του προτεινόμενου αλγορίθμου έγινε στο λογισμικό **Visual Studio 2013** της εταιρείας Microsoft, στην αντικειμενοστραφή γλώσσα προγραμματισμού C#.NET έκδοσης framework 4.5, καθώς η συγκεκριμένη γλώσσα προγραμματισμού συνδυάζει την υπολογιστική δύναμη της C++ και την ευκολία προγραμματισμού της Visual Basic. Κατά την υλοποίηση του συστήματος δόθηκε ιδιαίτερη έμφαση στην συντηρησιμότητα, την επεκτασιμότητα και την ευανάγνωστη γραφή του παραγόμενου κώδικα, μέσω της προσεκτικής σχεδίασης των κλάσεων, της ξεκάθαρης λογικής συνεπαγωγής του κώδικα αλλά και της προσθήκης των απαραίτητων σχολίων και της συστηματικής ονοματολογίας των μεταβλητών που χρησιμοποιήθηκαν.

8.2 Παράμετροι προτεινόμενου αλγορίθμου

Παρουσιάζονται οι βασικοί παράμετροι των δύο γενετικών αλγορίθμων, οι οποίοι επηρεάζουν την ταχύτητα σύγκλισης, το χρόνο εκτέλεσης και την ποιότητα της λύσης που δημιουργείται.

8.2.1 Παράμετροι Εσωτερικού Γενετικού Αλγορίθμου

Παρουσιάζονται οι βασικές παράμετροι του εσωτερικού γενετικού αλγορίθμου. Η πιθανότητα μετάλλαξης αφορά την πιθανότητα με την οποία ένα χρωμόσωμα ανταλλάσσει δύο τυχαία στοιχεία του. Αντίστοιχα, η πιθανότητα διασταύρωσης και αναστροφής είναι η πιθανότητα με την οποία ένα χρωμόσωμα διασταυρώνεται με ένα άλλο και η πιθανότητα με την οποία ένα χρωμόσωμα αλλάζει τη σειρά ορισμένων στοιχείων του αντίστοιχα. Ο αριθμός χρωμοσωμάτων είναι ο αριθμός των λύσεων που χρησιμοποιούνται σαν πληθυσμός στον εσωτερικό γενετικό αλγόριθμο. Τέλος, ο αριθμός των επαναλήψεων αφορά τις εσωτερικές επαναλήψεις του γενετικού αλγορίθμου μέχρι να τροφοδοτήσει τον εξωτερικό γενετικό αλγόριθμο.

Pmutation	Πιθανότητα Μετάλλαξης
Pcrossover	Πιθανότητα Διασταύρωσης
Pinversion	Πιθανότητα Αναστροφής
NoChromonew	Αριθμός χρωμοσωμάτων
NoIter	Αριθμός Επαναλήψεων

Πινακας 5 Παράμετροι εσωτερικού γενετικού αλγορίθμου

8.2.2 Παράμετροι Εξωτερικού Γενετικού Αλγορίθμου

Κατά αντιστοιχία με τις παραμέτρους του εσωτερικού γενετικού αλγορίθμου, οι πιθανότητες διασταύρωσης και μετάλλαξης είναι η πιθανότητα με τις οποίες ένα χρωμόσωμα του εξωτερικού γενετικού αλγορίθμου διασταυρώνεται με ένα άλλο ή μεταλλάσσεται αντίστοιχα. Η παράμετρος λ είναι μια παράμετρος ποινής, η οποία αλλάζει την ποιότητα της παραγόμενης λύσης, εαν αυτή δεν ικανοποιεί κάποιο περιορισμό. Τέλος, ο αριθμός χρωμοσωμάτων και ο αριθμός των επαναλήψεων έχουν ακριβώς το ίδιο νόημα για τον εξωτερικό γενετικό, όπως είχαν και για τον εσωτερικό.

λ	Παράμετρος Ποινής
Pcrossover	Πιθανότητα Διασταύρωσης
Pmutation	Πιθανότητα Μετάλλαξης
NoChromosome	Αριθμος χρωμοσωμάτων
NoIterations	Αριθμός Επαναλήψεων

Πινακας 6 Παράμετροι εξωτερικού γενετικού αλγορίθμου

8.3 Βασικοί μέθοδοι που υλοποιήθηκαν

8.3.1 Μέθοδοι Εσωτερικού Γενετικού

Ο εσωτερικός γενετικός αλγόριθμος είναι το ένα από τα δύο κύρια μέρη του αλγορίθμου που υλοποιήθηκε. Ο αλγόριθμος αυτός δέχεται ως είσοδο από τον εξωτερικό γενετικό αλγόριθμο ένα σύνολο πελατών και τα χαρακτηριστικά μιας κατηγορίας οχημάτων. Το αποτέλεσμα του είναι όλα τα δρομολόγια της εκάστοτε κατηγορίας που εξυπηρετούν τους δοθέντες πελάτες, η επίλυση δηλαδή ενός μικρότερου προβλήματος δρομολόγησης οχημάτων. Ο αλγόριθμος ξεκινά δημιουργώντας ένα σύνολο αρχικών λύσεων ίσο με τον αριθμό των χρωμοσωμάτων. Οι λύσεις αυτές προέρχονται από διάφορες μεθόδους που περιγράφονται στη συνέχεια. Στη συνέχεια βάσει των παραμέτρων που ορίστηκαν νωρίτερα υπολογίζεται ο νέος πληθυσμός, ο οποίος είναι ίσος σε μέγεθος με τον αρχικό πληθυσμό. Το άθροισμα των δύο πληθυσμών αποτελεί το συνολικό πληθυσμό. Ο συνολικός πληθυσμός αφού δημιουργηθεί αξιολογείται με βάση τη διανυόμενη απόσταση και επιλέγεται ο νέος αρχικός πληθυσμός με συγκεκριμένο μέθοδο. Ο νέος αρχικός πληθυσμός ακολουθεί την ίδια διαδικασία μέχρι έναν ορισμένο αριθμό επαναλήψεων. Το αποτέλεσμα αυτής της διαδικασίας διοχετεύεται στον εξωτερικό γενετικό αλγόριθμο, όπου και συνεχίζει τις επαναλήψεις.

8.3.1.1 Δημιουργία Αρχικού πληθυσμού

Για τη δημιουργία του αρχικού πληθυσμού χρησιμοποιούνται τέσσερις βασικές κατασκευαστικές μέθοδοι, τρείς ευρετικές και μια τυχαίας κατασκευής. Οι μέθοδοι αυτοί περιγράφονται στη συνέχεια.

8.3.1.1.1 Μέθοδος Savings

Η μέθοδος Savings που υλοποιείται είναι η σειριακή εκδοχή του αλγορίθμου των Clark και Wright, ο οποίος παρουσιάστηκε σε προηγούμενη ενότητα. Η μέθοδος υπολογίζει αρχικά, το κέρδος από την συνένωση δύο διαδρομών για οποιεσδήποτε διαδρομές είναι δυνατές και δημιουργεί τον αντίστοιχο πίνακα. Στη συνέχεια δημιουργεί σειριακά διαδρομές,ακολουθώντας τις διαδρομές του πίνακα που έχουν ιεραρχηθεί σε φθίνουσα σειρά, με τον ακόλουθο τρόπο:

1. Αν κανένας πελάτης από τους δύο δεν ανήκει στη διαδρομή δημιουργείται νέα διαδρομή με αυτούς τους δύο πελάτες.
2. Αν ένας από τους δύο πελάτες ανήκει σε διαδρομή και είναι εξωτερικός στην αλυσίδα, δηλαδή δεν παρεμβάλλεται κάποιος άλλος πελάτης μεταξύ αυτού και της κεντρικής αποθήκης είτε κατά τη φορά εξυπηρέτησης είτε κατά την αντίθετη φορά΄, τότε ο πελάτης που δεν ανήκει σε διαδρομή εισέρχεται στην ήδη υπάρχουσα.

Για την εισαγωγή ενός πελάτη σε μια διαδρομή, απαραίτητη προϋπόθεση αποτελεί η μη υπέρβαση της εναπομένουσας χωρητικότητας του οχήματος από την προσθήκη του επιπλέον πελάτη. Σε κάθε διαδρομή εκτός από το σύνολο των πελατών που δρομολογούνται, προστίθεται η κεντρική αποθήκη πριν από τον πρώτο και μετά από τον τελευταίο πελάτη.

8.3.1.1.2 Μέθοδος Sweep

Η μέθοδος Sweep που υλοποιείται είναι ο αλγόριθμος των Gilet και Miller που περιγράφηκε σε προηγούμενη ενότητα. Η μέθοδος υπολογίζει τις πολικές γωνίες των πελατών ως προς την κεντρική αποθήκη και τις αποθηκεύει σε κατάλληλο πίνακα. Στη συνέχεια, αρχίζει και δημιουργεί διαδρομές προσθέτοντας τους πελάτες με τις μικρότερες γωνίες βάσει του πίνακα που δημιουργήθηκε. Όταν κάποια διαδρομή δεν μπορεί να παραλάβει άλλο πελάτη γιατί η εναπομένουσα χωρητικότητα του οχήματος δεν επαρκεί, η μέθοδος δημιουργεί νέα διαδρομή και συνεχίζει από τον τελευταίο πελάτη, ο οποίος δεν δρομολογήθηκε. Η διαδικασία συνεχίζεται μέχρι κάθε πελάτης να τοποθετηθεί σε ένα δρομολόγιο. Σε κάθε διαδρομή εκτός από το σύνολο των πελατών που δρομολογούνται, προστίθεται η κεντρική αποθήκη πριν από τον πρώτο και μετά από τον τελευταίο πελάτη.

8.3.1.1.3 Μέθοδος Moles & Jameson

Η μέθοδος αυτή αποτελείται υλοποίηση του αλγορίθμου των Moles και Jameson, που περιγράφηκε νωρίτερα. Συγκεκριμένα, η μέθοδος αρχικά επιλέγεται ένας τυχαίος πελάτης, ο οποίος δεν έχει ήδη δρομολογηθεί. Ο πελάτης αυτός εισάγεται σε μια νέα διαδρομή και ξεκινάει μια επαναληπτική διαδικασία. Αρχικά, υπολογίζεται το κόστος εισαγωγής ενός νέου πελάτη στη διαδρομή για κάθε δυνατή θέση εισαγωγής του και αναζητείται το ελάχιστο κόστος εισαγωγής κάθε πελάτη. Εν συνεχεία, για κάθε πελάτη υπολογίζεται το κέρδος εισαγωγής του στην διαδρομή, ως η διαφορά του κόστους σύνδεσης του πελάτη με την κεντρική αποθήκη και του κόστους εισαγωγής στη νέα διαδρομή. Ο πελάτης με το μεγαλύτερο κέρδος εισαγωγής εισάγεται στη νέα διαδρομή, στη θέση που υποδεικνύεται από το ελάχιστο κόστος εισαγωγής, όπως αυτό υπολογίστηκε. Η διαδικασία επαναλαμβάνεται μέχρι το όχημα να μην διαθέτει αρκετή χωρητικότητα για να παραλάβει άλλο πελάτη. Μόλις συμβεί αυτό, η διαδρομή ολοκληρώνεται και η διαδικασία ξεκινάει από την αρχή για άλλο πελάτη. Σε κάθε διαδρομή εκτός από το σύνολο των πελατών που δρομολογούνται, προστίθεται η κεντρική αποθήκη πριν από τον πρώτο και μετά από τον τελευταίο πελάτη.

8.3.1.1.4 Μέθοδος Randomizing

Η μέθοδος αυτή είναι μια μέθοδος κατασκευής μιας αρχικής λύσης, η οποία όμως δεν βασίζεται σε κάποια ευρετική τεχνική. Σκοπός της είναι η εμφάνιση νέων χαρακτηριστικών στον αρχικό πληθυσμό, τα οποία θα οδηγήσουν σε διαφοροποίηση των χρωμοσωμάτων, προκειμένου να βρεθούν διαφορετικές καλύτερες λύσεις. Η μέθοδος χρησιμοποιείται επαναληπτικά. Ξεκινώντας από ένα πελάτη, η μέθοδος συνεχίζει και προσθέτει πελάτες στην υπάρχουσα διαδρομή, μέχρι να μην χωράει κανένας άλλος πελάτης, λόγω αδυναμίας υπέρβασης της χωρητικότητας του φορτηγού. Οι πελάτες δεν επιλέγονται βάσει κάποιου κριτηρίου, αλλά βάσει τυχαίων αριθμών. Όταν μια διαδρομή ολοκληρωθεί, δημιουργείται μια νέα, μέχρι όλοι οι πελάτες να δρομολογηθούν. Σε κάθε διαδρομή εκτός από το σύνολο των πελατών που δρομολογούνται, προστίθεται η κεντρική αποθήκη πριν από τον πρώτο και μετά από τον τελευταίο πελάτη.

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Δημιούργησε νέα διαδρομή.
2. Όσο υπάρχουν μη δρομολογημένοι πελάτες επανέλαβε:
    a. Επέλεξε πελάτη που δεν έχει δρομολογηθεί και χωράει στην παρούσα διαδρομή και προσέθεσε τον.
    b. Μείωσε την εναπομένουσα χωρητικότητα σύμφωνα με τη ζήτηση του πελάτη.
    c. Πήγαινε στο βήμα 2.
3. Τέλος Αλγορίθμου.

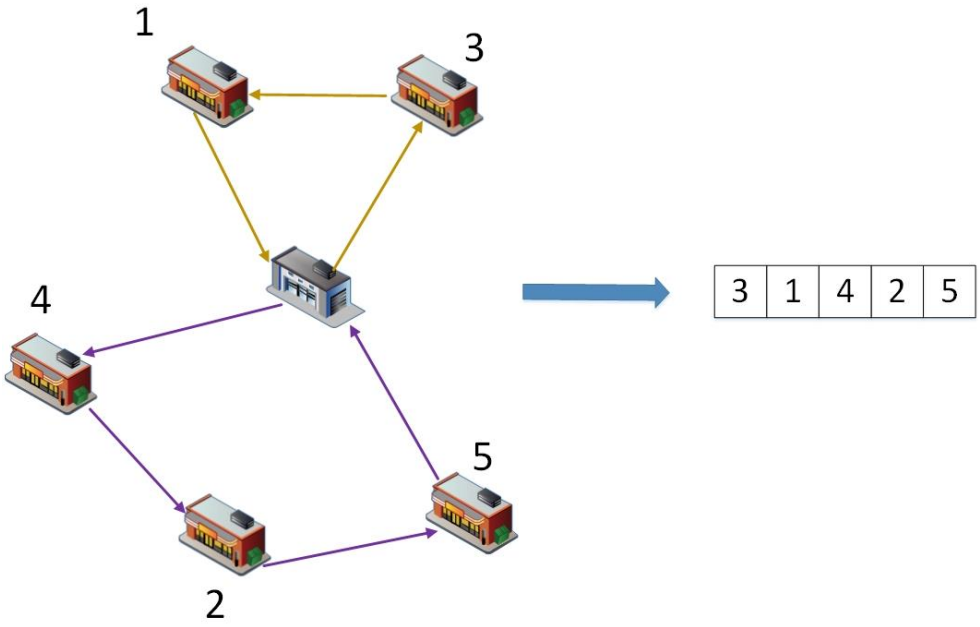
Αλγόριθμος 12 Μέθοδος Randomizing

8.3.1.2 Κωδικοποίηση/Αποκωδικοποίηση λύσεων

Η κωδικοποίηση και αποκωδικοποίηση των λύσεων είναι απαραίτητες για τη λειτουργία του γενετικού αλγορίθμου και βασίζονται στον τρόπο αναπαράστασης των λύσεων. Για τον εσωτερικό γενετικό αλγόριθμο επιλέχθηκε η αναπαράσταση λύσεων που ακολουθεί τους παρακάτω κανόνες:

1. Κάθε χρωμόσωμα έχει τόσες θέσεις όσοι και οι πελάτες του προβλήματος που επιλύεται(υποσύνολο των συνολικών πελατών).
2. Κάθε όχημα λαμβάνει πελάτες με τη σειρά ‘’σαρώνοντας’’ από τα αριστερά προς τα δεξία με τη σειρά που εμφανίζονται μέχρι η εναπομείνουσα χωρητικότητα του να μην επαρκεί για την είσοδο του επόμενου πελάτη στο δρομολόγιο.
3. Η κεντρική αποθήκη δεν περιλαμβάνεται σε κανένα χρωμόσωμα και προστίθεται στην αποκωδικοποίηση.

Στο παρακάτω σχήμα φαίνεται η αναπαράσταση μιας λύσης ενός προβλήματος δρομολόγησης με 5 πελάτες και 2 οχήματα:



Σχήμα 37 Παράδειγμα Αναπαράστασης Εσωτερικού Γενετικού

Η κωδικοποίηση και η αποκωδικοποίηση είναι η διαδικασία ‘’μετάφρασης’’ της διαδρομής/λύσης σε χρωμόσωμα και του χρωμοσώματος σε διαδρομή/λύση αντίστοιχα. Παρακάτω παρουσιάζονται οι δύο μέθοδοι που χρησιμοποιήθηκαν.

8.3.1.2.1 Μέθοδος Route Encoding

Η μέθοδος αυτή αποτελεί τη μέθοδο της κωδικοποίησης μιας λύσης/διαδρομής σε χρωμόσωμα. Δέχεται ως είσοδο μια λύση σε μορφή λίστας και δίνει ως έξοδο ένα χρωμόσωμα, σύμφωνα με την αναπαράσταση που περιγράφηκε στην προηγούμενη ενότητα. Αρχικά, επιλέγεται ένα όχημα και οι πελάτες της διαδρομής του τοποθετούνται στο χρωμόσωμα, με τη σειρά με την οποία εξυπηρετούνται. Στη συνέχεια, επιλέγεται το επόμενο όχημα και οι πελάτες της διαδρομής του τοποθετούνται στο χρωμόσωμα, μετά τον τελευταίο πελάτη του προηγούμενου οχήματος που τοποθετήθηκε. Η διαδικασία επαναλαμβάνεται για όλα τα οχήματα.

Ακολουθεί ο ψευδοκώδικας της κωδικοποίησης:

1. Δημιούργησε νέο χρωμόσωμα.
2. Όσο υπάρχουν οχήματα που δεν έχουν χρησιμοποιηθεί επανέλαβε.
    a. Επέλεξε ένα όχημα που δεν έχει χρησιμοποιηθεί.
    b. Τοποθέτησε τους πελάτες του οχήματος με τη σειρά στο χρωμόσωμα.
    c. Πήγαινε στο βήμα 2.
3. Τέλος αλγορίθμου.

Αλγόριθμος 13 Μέθοδος Coding

8.3.1.2.2 Μέθοδος Route Decoding

Η μέθοδος αυτή αποτελεί τη μέθοδο της αποκωδικοποίησης ενός χρωμοσώματος σε λύση/διαδρομή. Δέχεται ως είσοδο ένα χρωμόσωμα και δίνει ως έξοδο μια λύση σε μορφή λίστας. Πρόκειται για την αντίστροφη διαδικασία της κωδικοποίησης. Αρχικά, επιλέγεται ένα όχημα και λαμβάνει τους πελάτες του χρωμοσώματος σαρώνοντας από αριστερά προς τα δεξιά, μέχρι η ζήτηση ενός πελάτη να υπερβαίνει την εναπομένουσα χωρητικότητα του οχήματος, οπότε και ολοκληρώνεται το παρόν δρομολόγιο. Στο τέλος και στην αρχή κάθε δρομολογίου προστίθεται η κεντρική αποθήκη. Η διαδικασία επαναλαμβάνεται για όλους τους πελάτες του χρωμοσώματος και συμπληρώνονται τα αντίστοιχα οχήματα.Ακολουθεί ο ψευδοκώδικας της αποκωδικοποίησης:

1. Όσο υπάρχουν πελάτες που δεν έχουν σαρωθεί στο χρωμόσωμα επανέλαβε:
    a. Δημιούργησε νέο δρομολόγιο.
    b. Όσο η εναπομένουσα χωρητικότητα του οχήματος είναι μεγαλύτερη ή ίση από τη ζήτηση του επόμενου πελάτη επανέλαβε:
        i. Τοποθέτησε τον πελάτη στη διαδρομή και ανανέωσε την εναπομένουσα χωρητικότητα του οχήματος.
        ii. Πήγαινε στο βήμα b.
    c. Πήγαινε στο βήμα 1.
2. Τέλος Αλγορίθμου

Αλγόριθμος 14 Μέθοδος RouteDecoding

8.3.1.3 Μέθοδος 2-opt

Η μέθοδος αυτή είναι η υλοποίηση του αλγορίθμου 2-opt που περιγράφηκε σε προηγούμενη ενότητα. Δέχεται ως είσοδο μια λύση/διαδρομή σε μορφή λίστας και δίνει ως έξοδο μια καλύτερη λύση διαδρομή με τη χρήση του αλγορίθμου. Ο αλγόριθμος ξεκινά από τον πρώτη διαδρομή στη λίστα και επαναλαμβάνει τη διαδικασία για κάθε δρομολόγιο ξεχωριστά. Αρχικά επιλέγει τον πρώτο πελάτη και αρχίζει να αλλάζει τη σειρά με την οποία εξυπηρετείται στο δρομολόγιο. Αν βρεθεί κάποια νέα θέση στην οποία το συνολικό κόστος της διαδρομής είναι μικρότερο από πριν, τότε η λύση αλλάζει και αποθηκεύεται η νέα διαδρομή. Η διαδικασία αυτή επαναλαμβάνεται σειριακά για όλους τους πελάτες. Η μέθοδος 2-opt επεμβαίνει σε κάθε διαδρομή ξεχωριστά και δεν αλλάζει τους πελάτες που εξυπηρετούνται από κάθε όχημα. Η χρησιμοποίηση της, συγκριτικά με μια μέθοδο 3-opt επιλέχθηκε λόγω της απλότητας υλοποίησης της και της ανεπαίσθητης διαφοράς στα αποτελέσματα των δύο μεθόδων.

Ακολουθεί ο ψευδοκώδικας της 2-opt όπως αυτή υλοποιήθηκε:

1. Όσο υπάρχουν διαδρομές που δεν έχουν ελεγχθεί επανέλαβε:
    a. Επέλεξε την πρώτη μη ελεγμένη διαδρομή.
    b. Όσο υπάρχουν πελάτες που δεν έχουν ελεγχθεί επανέλαβε:
        i. Επέλεξε τον πρώτο πελάτη που δεν έχει ελεγχθεί.
        ii. Μετακίνησε τον πελάτη σε διαδοχικές θέσεις.
        iii. Αν η νέα διαδρομή είναι καλύτερη από την προηγούμενη, αποθήκευσε την.
        iv. Πήγαινε στο βήμα b.
    c. Πήγαινε στο βήμα a.
2. Τέλος Αλγορίθμου.

Αλγόριθμος 15 Μέθοδος 2 opt

8.3.1.4 Μέθοδος PMX Crossover

Η μέθοδος αυτή αποτελεί τη μέθοδο διασταύρωσης του εσωτερικού γενετικού αλγορίθμου. Η χρήση της είναι αναγκαία σε σχέση με την απλή διασταύρωση, εφόσον οι λύσεις που δημιουργεί είναι πάντοτε εφικτές. Η απλή διασταύρωση, στη γενική περίπτωση θα δημιουργούσε προβλήματα, επειδή οι κατασκευασμένες λύσεις της, θα περιείχαν κάποιους πελάτες δύο φορές και δεν θα περιείχαν κάποιους άλλους καθόλου. Έτσι, θα παραβιαζόταν ο περιορισμός του προβλήματος δρομολόγησης οχημάτων, κάθε πελάτης να επισκέπτεται ακριβώς μια φορά. Η Pmx διασταύρωση επιλέγει ένα τμήμα του ενός χρωμοσώματος και το αντικαθιστά στο άλλο στην αντίστοιχη θέση. Οι πελάτες που δεν επηρεάζονται από το νέο τμήμα διατηρούνται από το δεύτερο χρωμόσωμα. Αντίθετα, όσοι πελάτες επηρεάζονται από το νέο τμήμα, λόγω ύπαρξης τους δύο φορές συμπληρώνονται σύμφωνα με τη μέθοδο Pmx αυτόματα, όπως περιγράφηκε σε προηγούμενη ενότητα. Έτσι, οι λύσεις που παράγονται δεν παρουσιάζουν προβλήματα και ο αλγόριθμος λειτουργεί ομαλά.

Ακολουθεί ο ψευδοκώδικας της μεθόδου PMX:

1. Επέλεξε ένα τυχαίο τμήμα του πρώτου χρωμοσώματος.
2. Τοποθέτησε το στην αντίστοιχη θέση του δεύτερο χρωμοσώματος.
3. Συμπλήρωσε όσους πελάτες του δεύτερου χρωμοσώματος δεν περιλαμβάνονται στο τμήμα που συμπληρώθηκε.
4. Συμπλήρωσε τους υπόλοιπους πελάτες σύμφωνα με τις αλλαγές Pmx, ούτως ώστε κάθε πελάτης να περιλαμβάνεται ακριβώς μια φορά.
5. Επανέλαβε τη διαδικασία για το άλλο χρωμόσωμα.

Αλγόριθμος 16 Μέθοδος PMX διασταύρωση

8.3.1.5 Μέθοδος Mutation

Η μέθοδος Mutation είναι η υλοποιούμενη μέθοδος μετάλλαξης του εσωτερικού γενετικού αλγορίθμου. Στόχος της είναι η εμφάνιση νέων χαρακτηριστικών στον πληθυσμό, προκειμένου να ερευνηθεί μεγαλύτερο φάσμα λύσεων με σκοπό την εύρεση της βέλτιστης. Η λειτουργία της βασίζεται στην ανταλλαγή δύο πελατών σε ένα χρωμόσωμα. Έτσι, η μέθοδος αυτή επηρεάζει τόσο τη σειρά με την οποία εξυπηρετούνται κάποιοι πελάτες, όσο και το όχημα από το οποίο εξυπηρετούνται.

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Επέλεξε δύο τυχαίους πελάτες.
2. Άλλαξε τις θέσεις τους, χωρίς καμιά άλλη μεταβολή στο υπόλοιπο χρωμόσωμα.
3. Τέλος Αλγορίθμου.

Αλγόριθμος 17 Μέθοδος Mutation

8.3.1.6 Μέθοδος Inversion

Η μέθοδος Inversion είναι η υλοποιημένη μέθοδος αναστροφής του εσωτερικού γενετικού αλγορίθμου. Λειτουργεί σαν ένας συνδυασμός τοπικής αναζήτησης και μετάλλαξης. Συγκεκριμένα, η μέθοδος αυτή επιλέγει δύο πελάτες και αναστρέφει τη σειρά όλων όσων περιλαμβάνονται μεταξύ τους. Η αναστροφή αποθηκεύεται μόνο αν η νέα λύση είναι καλύτερη από την προηγούμενη.Ακολυθεί ο ψευδοκώδικας της μεθόδου:

1. Επέλεξε δύο τυχαίους πελάτες.
2. Ανέστρεψε τη σειρά όλων των πελατών που περιλαμβάνονται μεταξύ τους.
3. Αν η λύση που παράγεται είναι καλύτερη, αποθήκευσε τη.
4. Τέλος Αλγορίθμου.

Αλγόριθμος 18 Μέθοδος Inversion

8.3.1.7 Μέθοδος Fitness

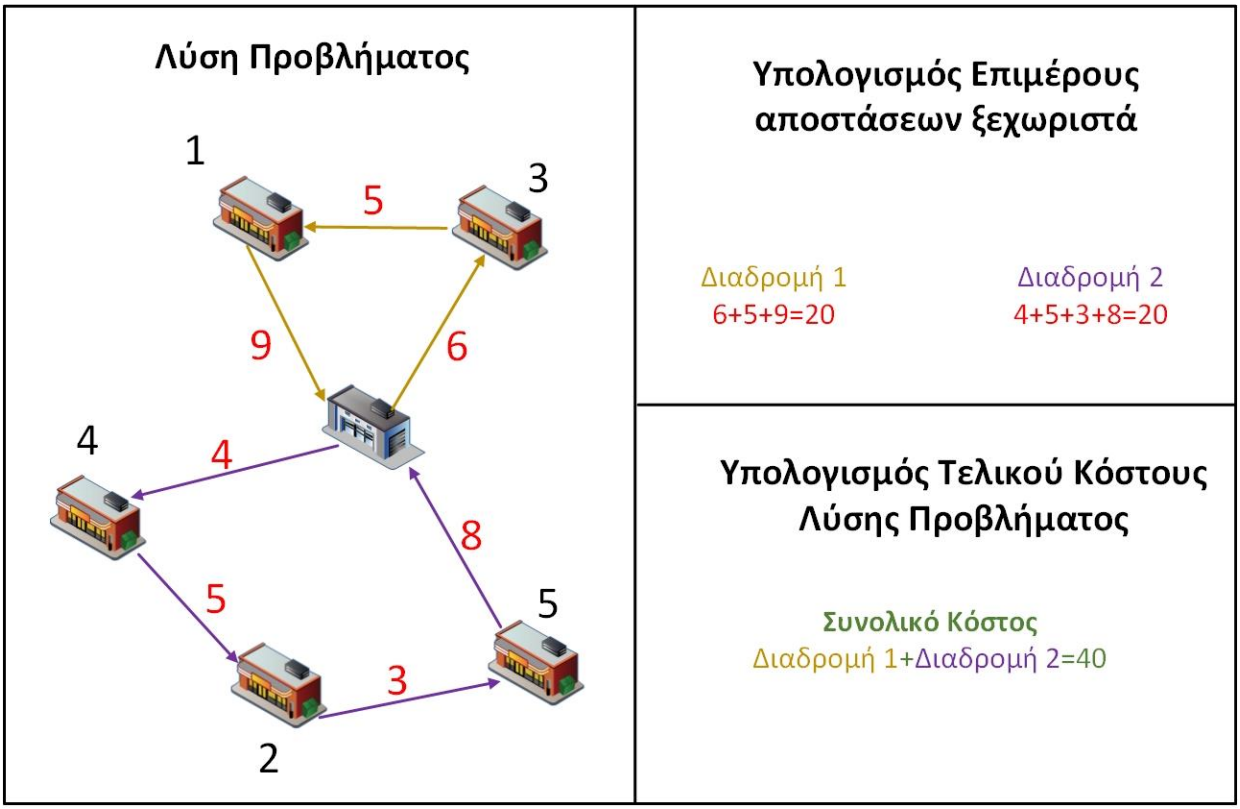
Η μέθοδος αυτή αποτελεί την μέθοδο αξιολόγησης του πληθυσμού σε κάθε επανάληψη του εσωτερικού γενετικού αλγορίθμου. Δέχεται ως είσοδο μια λύση/διαδρομή και δίνει ως έξοδο το κόστος της λύσης αυτής, το οποίο μεταφράζεται σε συνολικά διανυόμενη απόσταση. Η μέθοδος υπολογίζει για κάθε δρομολόγιο που απαρτίζεται τη συνολική διαδρομή το κόστος του. Στη συνέχεια, προσθέτει όλα τα κόστη μαζί, βρίσκοντας το τελικό κόστος της λύσης.

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Όσο υπάρχουν διαδρομές που δεν υπολογίστηκαν επανέλαβε:
    a. Επέλεξε μια διαδρομή που δεν έχει υπολογιστεί.
    b. Υπολόγισε το κόστος της ως το άθροισμα των αποστάσεων μεταξύ των πελατών που την απαρτίζουν.
    c. Πήγαινε στο βήμα 1.
2. Τέλος Αλγορίθμου.

Αλγόριθμος 19 Μέθοδος Fitness

Στο σχήμα που ακολουθεί παρουσιάζεται η διαδικασία υπολογισμού μιας τελικής λύσης ενός προβλήματος δρομολόγησης. Το κόστος κάθε διαδρομής φαίνεται πάνω από την αντίστοιχη ακμή.



Σχήμα 38 Μέθοδος Fitness

8.3.1.8 Μέθοδος Selection

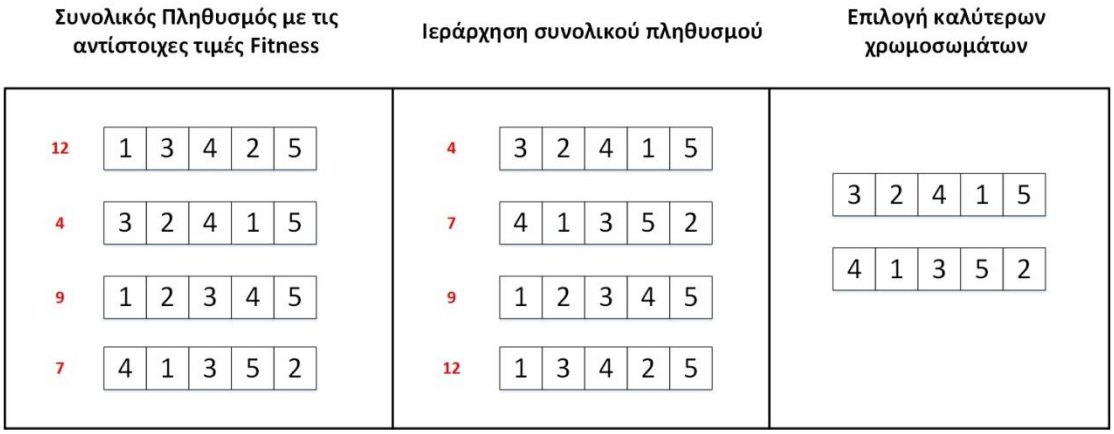
Η μέθοδος Selection είναι η μέθοδος που υλοποιεί την επιλογή νέου πληθυσμού στο τέλος κάθε επανάληψης του εσωτερικού γενετικού αλγορίθμου. Δέχεται ως είσοδο ένα σύνολο χρωμοσωμάτων διπλάσιο σε μέγεθος από τον αρχικό πληθυσμό, το οποίο περιλαμβάνει τόσο τον παλαιό πληθυσμό, όσο και τον νέο, και δίνει ως έξοδο το νέο πληθυσμό που θα χρησιμοποιηθεί στη νέα επανάληψη. Η μέθοδος ξεκινά χρησιμοποιώντας τις τιμές της αντικειμενικές συνάρτησης που υπολόγισε η μέθοδος Fitness. Τις τιμές αυτές τις κατατάσσει κατά αύξουσα σειρά, τοποθετώντας έτσι τις καλύτερες λύσεις στην αρχή. Στη συνέχεια επιλέγει τα πρώτα χρωμοσώματα της λίστας, ίσα σε αριθμό με τον πληθυσμό που πρέπει να επιλεγεί. Η προσέγγιση αυτή, αν και δεν αφήνει τον αλγόριθμο να κρατήσει χειρότερες λύσεις που πιθανώς τον οδηγήσουν στη βέλτιστη λύση, κρίθηκε σκόπιμο να χρησιμοποιηθεί, λόγω της εύκολης υλοποίησης της και των ιδιαίτερα καλών αποτελεσμάτων που απέφερε, μετά από τη δοκιμή της σε προβλήματα αναφοράς.

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Πάρε τον πίνακα των τιμών της μεθόδου Fitness.
2. Ιεράρχησε τες κατά αύξουσα σειρά.
3. Επέλεξε τις N πρώτες, όπου N το μέγεθος του αρχικού πληθυσμού.
4. Θέσε τα N χρωμοσώματα που αντιστοιχούν σε αυτές τις τιμές ως νέο αρχικό πληθυσμό.
5. Τέλος Αλγορίθμου.

Αλγόριθμος 20 Μέθοδος Selection

Στο σχήμα που ακολουθεί παρουσιάζεται η μέθοδος Selection σε πλήθος 4 χρωμοσωμάτων, από τα οποία πρέπει να επιλεγούν 2. Δίπλα σε κάθε χρωμόσωμα βρίσκεται η τιμή της μεθόδου Fitness.



Σχήμα 39 Μέθοδος Selection

8.3.2 Μέθοδοι Εξωτερικού Γενετικού Αλγορίθμου

Ο εξωτερικός γενετικός αλγόριθμος αποτελεί το δεύτερο κύριο συστατικό του μεταευρετικού αλγορίθμου. Δέχεται ως είσοδο τα χαρακτηριστικά όλων των πελατών και οχημάτων, τις ζώνες κίνησης και τα χαρακτηριστικά τους, την αρχή και τη λήξη της βάρδιας και δίνει ως έξοδο τα δρομολόγια κάθε οχήματος, κάθε κατηγορίας. Ο αλγόριθμος ξεκινά δημιουργώντας ένα τυχαίο αρχικό πληθυσμό με τη μέθοδο Randomexterior. Στη συνέχεια, διασταυρώνει τα χρωμοσώματα με τη μέθοδο CrossoverExterior και τα μεταλλάσσει με τη μέθοδο MutationExterior. Αφού δημιουργήσει το συνολικό πληθυσμό, ο εξωτερικός γενετικός αλγόριθμος καλεί τον εσωτερικό γενετικό αλγόριθμο, ο οποίος επιλύει το πρόβλημα δρομολόγησης, όπως αναλύθηκε στην προηγούμενη ενότητα. Τα αποτελέσματα του εσωτερικού γενετικού τροφοδοτούνται εκ νέου στον εξωτερικό αλγόριθμο, όπου συλλέγονται και αξιολογούνται ως προς το συνολικό κόστος. Η αξιολόγηση γίνεται με τη μέθοδο FitnessExterior. Το αποτέλεσμα της αξιολόγησης χρησιμοποιείται από τη μέθοδο SelectionExterior, όπου κάνει την τελική επιλογή των χρωμοσωμάτων της επόμενης επανάληψης. Η διαδικασία αυτή επαναλαμβάνεται για τον ορισθέντα αριθμό επαναλήψεων.

8.3.2.1 Δημιουργία Αρχικού Πληθυσμού

Για την δημιουργία του αρχικού πληθυσμού του εξωτερικού αλγορίθμου χρησιμοποιείται μια απλή μέθοδο αντιστοίχησης πελάτη σε κάθε κατηγορία οχήματος βάσει πιθανότητας. Συγκεκριμένα, ανάλογα με τον αριθμό των κατηγοριών των διαφορετικών οχημάτων δημιουργούνται κλάσεις πιθανοτήτων συγκεκριμένου εύρους. Το εύρος δίνεται από την εξίσωση:

$$F_i = \frac{1}{n} \qquad (8\text{-}1)$$

όπου n είναι το πλήθος των διαφορετικών κατηγοριών οχημάτων. Για κάθε πελάτη επιλέγεται ένας τυχαίος αριθμός μεταξύ του 0 και του 1. Η κλάση στην οποία ανήκει ο αριθμός υποδηλώνει την κατηγορία του οχήματος που θα εξυπηρετήσει τον πελάτη.

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Όσο υπάρχουν πελάτες που δεν ανήκουν σε κατηγορία οχημάτων επανέλαβε:
    a. Επέλεξε ένα πελάτη που δεν ανήκει σε κατηγορία οχημάτων.
    b. Επέλεξε ένα τυχαίο αριθμό μεταξύ 0 και 1.
    c. Τοποθέτησε τον πελάτη στην αντίστοιχη κατηγορία οχημάτων που υπαγορεύεται από τον αριθμό.
    d. Πήγαινε στο βήμα 1.
2. Τέλος Αλγορίθμου.

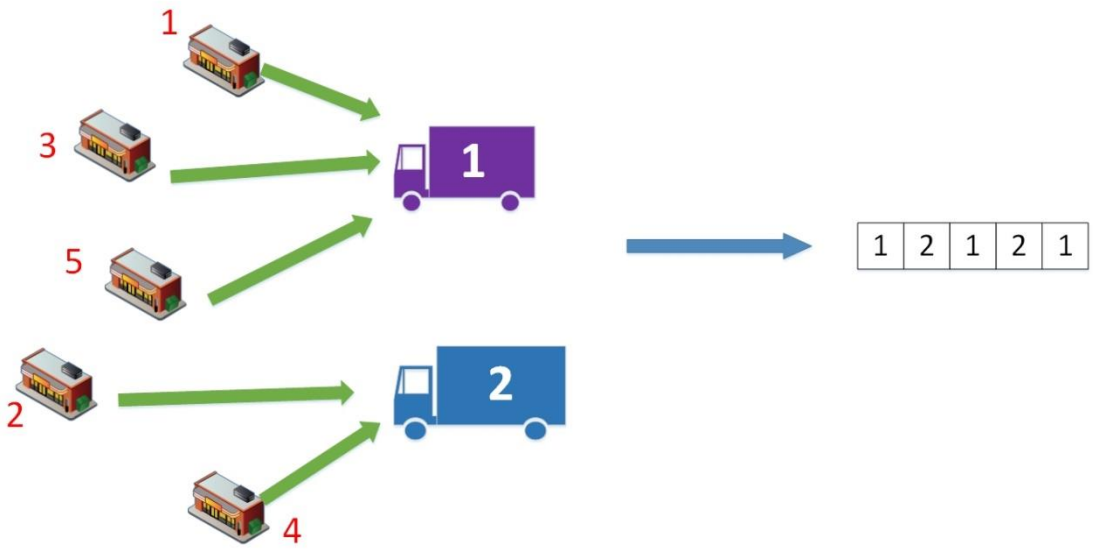
Αλγόριθμος 21 Μέθοδος RandomExterior

8.3.2.2 Κωδικοποίηση/Αποκωδικοποίηση λύσεων

Η κωδικοποίηση και αποκωδικοποίηση των λύσεων είναι απαραίτητες για τη λειτουργία του γενετικού αλγορίθμου και βασίζονται στον τρόπο αναπαράστασης των λύσεων. Για τον εξωτερικό γενετικό αλγόριθμο επιλέχθηκε η αναπαράσταση λύσεων που ακολουθεί τους παρακάτω κανόνες:

1. Κάθε χρωμόσωμα έχει τόσες θέσεις όσοι και οι πελάτες του προβλήματος.
2. Κάθε πελάτης αντιστοιχίζεται σε μια κατηγορία οχήματος, σύμφωνα με την προηγούμενη μέθοδο αντιστοίχησης.

Στο παρακάτω σχήμα φαίνεται η αναπαράσταση μιας λύσης ενός προβλήματος δρομολόγησης με 5 πελάτες και 2 κατηγορίες οχημάτων:



Σχήμα 40 Αναπαράσταση εξωτερικού Γενετικού

8.3.2.2.1 Μέθοδος Κωδικοποίησης

Η μέθοδος αυτή δέχεται ως είσοδο ένα σύνολο πελατών και κάποιες κατηγορίες οχημάτων και δίνει ως έξοδο ένα χρωμόσωμα. Η διαδικασία που ακολουθείται είναι αρκετά απλή και βασίζεται στην αναπαράσταση που περιγράφηκε νωρίτερα. Για κάθε κατηγορία οχημάτων, αν ο πελάτης ανήκει στην κατηγορία, βάσει της διαδικασίας αρχικοποίησης, η αντίστοιχη θέση του χρωμοσώματος συμπληρώνεται με τον κωδικό αριθμό της εκάστοτε κατηγορίας. Με άλλα λόγια, αν ο πρώτος πελάτης ανήκει στην τρίτη κατηγορία οχημάτων, η πρώτη θέση του χρωμοσώματος συμπληρώνεται με τον αριθμό τρία.

Ακολουθεί ο ψευδοκώδικας της μεθόδου CodingExterior:

1. Δημιούργησε ένα χρωμόσωμα.
2. Όσο υπάρχουν κενές θέσεις επανέλαβε:
    a. Επέλεξε την πρώτη κενή θέση του χρωμοσώματος.
    b. Βρες από την μέθοδο RandomExterior σε ποια κατηγορία ανήκει ο συγκεκριμένος πελάτης.
    c. Τοποθέτησε τον κωδικό αριθμό της κατηγορίας στην θέση αυτή του χρωμοσώματος.
    d. Πήγαινε στο βήμα 2.
3. Τέλος Αλγορίθμου.

Αλγόριθμος 22 Μέθοδος CodingExterior

8.3.2.2.2 Μέθοδος Αποκωδικοποίησης

Η μέθοδος αυτή δέχεται ως είσοδο ένα χρωμόσωμα και δίνει ως έξοδο ένα σύνολο πελατών που εξυπηρετείται από συγκεκριμένες κατηγορίες οχημάτων. Η αποκωδικοποίηση βασίζεται, όπως και η κωδικοποίηση, στην αναπαράσταση του χρωμοσώματος που περιγράφηκε νωρίτερα. Η διαδικασία είναι η αντίστροφη της CodingExterior και ονομάζεται **DecodingExterior**. Αρχικά, επιλέγεται το χρωμόσωμα που θα αποκωδικοποιηθεί. Στη συνέχεια, ξεκινώντας από την πρώτη θέση του χρωμοσώματος και σαρώνοντας από αριστερά προς τα δεξιά, κάθε πελάτης ανάλογα με τον αριθμό της αντίστοιχης θέσης του στο χρωμόσωμα, αντιστοιχίζεται σε μια κατηγορία οχημάτων. Έτσι, η θέση 1 του χρωμοσώματος που περιλαμβάνει τον αριθμό 3, αντιστοιχίζει τον πρώτο πελάτη με την τρίτη κατηγορία οχημάτων.

Ακολουθεί ο ψευδοκώδικας της μεθόδου DecodingExterior:

1. Επέλεξε ένα χρωμόσωμα.
2. Όσο υπάρχουν θέσεις στο χρωμόσωμα που δεν έχεις αντιστοιχίσει επανέλαβε:
    a. Επέλεξε την πρώτη από τα αριστερά θέση που δεν έχεις αντιστοιχίσει.
    b. Αντιστοίχισε τον πελάτη αυτής της θέσης με την κατηγορία που υποδεικνύει ο αριθμός που περιέχεται στη θέση αυτή.
    c. Πήγαινε στο βήμα 2.
3. Τέλος Αλγορίθμου

Αλγόριθμος 23 Μέθοδος DecodingExterior

8.3.2.3 Μέθοδος ExteriorCrossover

Η μέθοδος **ExteriorCrossover** αποτελεί την υλοποιήση της διασταύρωσης στον εξωτερικό γενετικό αλγόριθμο. Επιλέχθηκε να υλοποιηθεί απλή διασταύρωση στον εξωτερικό γενετικό αλγόριθμο, η οποία δεν θα επηρεάσει τις παραγόμενες λύσεις, ως προς την εφικτότητα τους. Συγκεκριμένα, δύο χρωμοσώματα που υφίστανται απλή διασταύρωση δημιουργούν δύο απογόνους. Για τη δημιουργία τους επιλέγεται μια τυχαία θέση του χρωμοσώματος βάσει ενός τυχαίου αριθμού. Ο πρώτος απόγονος λαμβάνει το τμήμα του πρώτου χρωμοσώματος πριν από τη θέση που επιλέχθηκε και το τμήμα του δεύτερου χρωμοσώματος μετά τη θέση που επιλέχθηκε, ενώ αντίθετα ο δεύτερος απόγονος λαμβάνει το τμήμα του δεύτερου χρωμοσώματος πριν από τη θέση που επιλέχθηκε και το τμήμα του πρώτου χρωμοσώματος μετά από τη θέση που επιλέχθηκε. Έτσι, τα δύο αρχικά χρωμοσώματα διασταυρώνονται μεταξύ τους, δημιουργώντας δύο νέους απογόνους.

Ακολουθεί ο ψευδοκώδικας της μεθόδου ExteriorCrossover:

1. Επέλεξε δύο χρωμοσώματα.
2. Επέλεξε μια τυχαία θέση βάσει ενός τυχαίου αριθμού.
3. Δημιούργησε τον πρώτο απόγονο με το πρώτο συνθετικό του πρώτου χρωμοσώματος και το δεύτερο συνθετικό του δεύτερου χρωμοσώματος.
4. Δημιούργησε το πρώτο συνθετικό του δεύτερου χρωμοσώματος και το δεύτερο απόγονο με το δεύτερο συνθετικό του πρώτου χρωμοσώματος.
5. Τέλος Αλγορίθμου.

Αλγόριθμος 24 Μέθοδος ExteriorCrossover

8.3.2.4 Μέθοδος ExteriorMutation

Η μέθοδος **ExteriorMutation** είναι η υλοποίηση της μετάλλαξης στον εξωτερικό γενετικό αλγόριθμο. Στόχος της είναι η εμφάνιση νέων χαρακτηριστικών στον πληθυσμό, προκειμένου να ερευνηθεί μεγαλύτερο φάσμα λύσεων με σκοπό την εύρεση της βέλτιστης. Η λειτουργία της βασίζεται στην ανταλλαγή της κατηγορίας του οχήματος εξυπηρέτησης δύο πελατών. Η ανταλλαγή αυτή μπορεί να μην προκαλέσει καμιά αλλαγή στο χρωμόσωμα, εφόσον δύο πελάτες μπορούν να εξυπηρετούνται από την ίδια κατηγορία οχήματος και για αυτό η πιθανότητα με την οποία μεταλλάσσονται τα στοιχεία του εξωτερικού γενετικού αποφασίστηκε να είναι πολύ μεγαλύτερη από αυτή του εσωτερικού γενετικού.

Ακολουθεί ο ψευδοκώδικας της μεθόδου:

1. Επέλεξε δύο τυχαίους πελάτες.
2. Αντάλλαξε το περιεχόμενο των θέσεων των δύο πελατών χωρίς οποιαδήποτε άλλη μεταβολή στο χρωμόσωμα.
3. Τέλος Αλγορίθμου.

Αλγόριθμος 25 Μέθοδος ExteriorMutation

8.3.2.5 Μέθοδος Evaluation

Η μέθοδος Evaluation αποτελεί την μέθοδο αξιολόγησης των λύσεων του εξωτερικού γενετικού αλγορίθμου, αντίστοιχη της Fitness του εσωτερικού γενετικού. Δέχεται ως είσοδο μια λύση του εξωτερικού γενετικού αλγορίθμου και τις επιμέρους λύσεις του εσωτερικού γενετικού αλγορίθμου και δίνει ως έξοδο το τελικό κόστος της λύσης του εξωτερικού γενετικού. Η μέθοδος δεν λαμβάνει υπόψη μόνο τη συνολικά διανυόμενη απόσταση αλλά και το κόστος υπεργολαβίας εαν αυτό απαιτείται σε μια λύση, την έλλειψη δηλαδή επαρκούς στόλου για την κάλυψη των αναγκών της επιχείρησης. Τέλος, η μέθοδος αυτή περιλαμβάνει και την υπέρβαση του ορίου εκπομπής διοξειδίου του άνθρακα σαν ποινή, μιας και δεν μπορεί να συμπεριληφθεί διαφορετικά στο πρόβλημα ο περιορισμός αυτός.

Ακολουθεί ο ψευδοκώδικας της μεθόδου Evaluation:

1. Επέλεξε ένα χρωμόσωμα.
2. Υπολόγισε το κόστος κάθε υποκατηγορίας οχήματος.
3. Υπολόγισε το κόστος υπεργολαβίας, αν υπάρχει.
4. Υπολόγισε την ποινή για την υπέρβαση ρύπων αν υπάρχει.
5. Υπολόγισε το συνολικό κόστος.
6. Τέλος Αλγορίθμου.

Αλγόριθμος 26 Μέθοδος Evaluation

8.3.2.6 Μέθοδος ExteriorSelection

Η μέθοδος αυτή αποτελεί την υλοποίηση της επιλογής του νέου πληθυσμού στον εξωτερικό γενετικό αλγόριθμο. Δέχεται ως είσοδο το συνολικό πληθυσμό της προηγούμενης επανάληψης καθώς και την τιμή της αντικειμενικής συνάρτησης κάθε χρωμοσώματος και δίνει ως έξοδο το νέο πληθυσμό της επόμενης επανάληψης. Αντίστοιχα με τον εσωτερικό γενετικό αλγόριθμο αποφασίσθηκε και αυτή η μέθοδος να επιλέγει τα καλύτερα χρωμοσώματα σε κάθε επανάληψη, όσον αφορά το συνολικό κόστος της παραγόμενης λύσης.

Ακολουθεί ο κώδικας της μεθόδου ExteriorSelection:

1. Πάρε τους πίνακες Fitness των επιμέρους κατηγοριών για κάθε χρωμόσωμα.
2. Πάρε τα κόστη υπεργολαβίας των επιμέρους κατηγοριών για κάθε χρωμόσωμα.
3. Πάρε την ποινή υπέρβασης ορίου εκπομπών για κάθε χρωμόσωμα αν υπάρχει.
4. Πρόσθεσε τα κόστη για κάθε χρωμόσωμα.
5. Ιεράρχησε τα κόστη σε αύξουσα σειρά.
6. Επέλεξε από την αρχή της λίστας τόσα χρωμοσώματα, όσα απαιτεί ο νέος πληθυσμός.
7. Τέλος Αλγορίθμου.

Αλγόριθμος 27 Μέθοδος ExteriorSelection

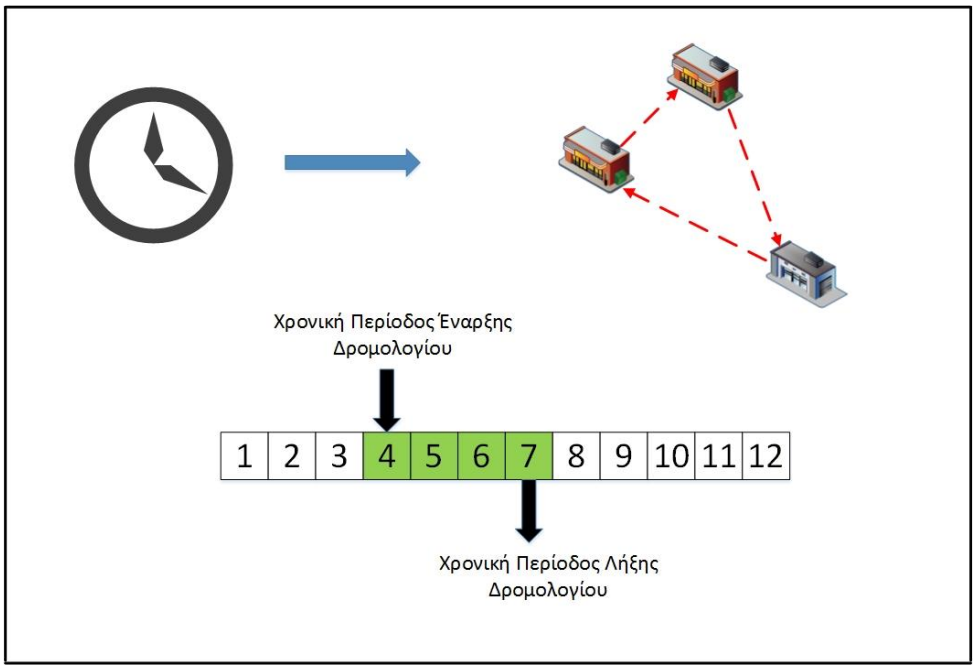
8.3.3 Άλλες Μέθοδοι

8.3.3.1 Μέθοδος GreenOptimization

Η μέθοδος αυτή χρησιμοποιείται για την βελτιστοποίηση των εκπομπών διοξειδίου του άνθρακα για μια δεδομένη διαδρομή. Συγκεκριμένα, γνωρίζοντας το ακριβές δρομολόγιο ενός οχήματος, η μέθοδος σύμφωνα με τις ζώνες κίνησης και τις ταχύτητες που επικρατούν σε αυτές αναζητεί σειριακά το καλύτερο χρόνο δρομολόγησης μέσα στη μέρα, για τον οποίο οι εκπομπές διοξειδίου του άνθρακα ελαχιστοποιούνται. Η μέθοδος αυτή δεν αποτελεί βασικό μέρος του αλγορίθμου και μπορεί να εξαιρεθεί, αν κάποια επιχείρηση είτε αποφασίσει να την αγνοήσει, είτε υπάρχει κάποια εταιρική πολιτική που την απαγορεύει. Ακολουθεί ο ψευδοκώδικας της μεθόδου GreenOptimization :

1. Επέλεξε μια λύση του προβλήματος.
2. Όσο υπάρχουν δρομολόγια που δεν έχουν προγραμματιστεί μέσα στη μέρα επανέλαβε:
    a. Επέλεξε ένα δρομολόγιο που δεν έχει ήδη προγραμματιστεί.
    b. Για κάθε χρονική περίοδο μέσα στη μέρα υπολόγισε το συνολικό αριθμό εκπομπών διοξειδίου του άνθρακα.
    c. Επέλεξε τη χρονική περίοδο έναρξης του δρομολογίου για την οποία ο συνολικός αριθμός εκπομπών διοξειδίου του άνθρακα γίνεται ελάχιστος.
    d. Πήγαινε στο βήμα 2.
3. Τέλος Αλγορίθμου.

Αλγόριθμος 28 Μέθοδος GreenOptimization



Σχήμα 41 Παράδειγμα Green Optimization

9 Αποτελέσματα και αξιολόγηση προβλημάτων βιβλιογραφίας

9.1 Παραμετροποίηση Αλγορίθμου

Οι παράμετροι του εσωτερικού γενετικού αλγορίθμου μελετήθηκαν βάσει της βιβλιογραφίας πάνω στα προβλήματα αναφοράς. Πραγματοποιήθηκαν μια σειρά από «τρεξίματα» του υλοποιημένου αλγορίθμου. Σύμφωνα με τα αποτελέσματα επιλέχθηκαν οι παρακάτω τιμές για τις παραμέτρους του αλγορίθμου:

Pmutation	7,5%
Pcrossover	98%
Pinversion	7,5%
NoChromonew	40
NoIter	25

Πινακας 7 Υλοποιημένοι Παράμετροι Εσωτερικού Γενετικού Αλγορίθμου

9.2 Παρουσίαση Προβλημάτων Αναφοράς

Τα προβλήματα αναφοράς(benchmark problems) που χρησιμοποιήθηκαν για τον έλεγχο σωστής λειτουργίας, αποδοτικότητας και αποτελεσματικότητας του προτεινόμενου αλγορίθμου ανήκουν σε τέσσερις μεγάλες κατηγορίες προβλημάτων. Οι κατηγορίες προβλημάτων είναι:

1. **Christofides, Mingozzi, Toth**: Περιλαμβάνει 14 προβλήματα με 50 εως 200 πελάτες.
2. **Taillard:** Περιλαμβάνει 13 προβλήματα με 75 εως 385 πελάτες.
3. **Golden, Wasil, Kelly, Chao:** Περιλαμβάνει 20 προβλήματα με 200 εως και 480 πελάτες.
4. **Li:** Περιλαμβάνει 12 προβλήματα με 560 εως και 1200 πελάτες.

Συγκεντρωτικά, τα χαρακτηριστικά των πελατών που δίνονται από τα προβλήματα αυτά για την απλή εκδοχή του είναι

- Ένας αύξοντας αριθμός-ταυτότητα του πελάτη, όπου θεωρείται ότι ο πρώτος πελάτης αντιπροσωπεύει την κεντρική αποθήκη
- Τις συντεταγμένες του πελάτη, από τις οποίες προκύπτει ο πίνακας κόστους μετάβασης από κάθε πελάτη (ή την κεντρική αποθήκη) προς κάθε άλλον (ή την κεντρική αποθήκη)
- Τη ζήτηση του κάθε πελάτη

9.3 Στατική Ανάλυση και αξιολόγηση

Ο αλγόριθμος που υλοποιήθηκε χρησιμοποιήθηκε σε καθένα από τα παρακάτω προβλήματα. Από τα μεγέθη του πίνακα:

1. **Cust.** (Customers) είναι ο αριθμός των πελατών του προβλήματος.
2. **BKV** (Best Known Value) είναι η καλύτερη λύση που έχει βρεθεί στη βιβλιογραφία.
3. **BV** (Best Value) είναι η καλύτερη λύση που βρήκε ο αλγόριθμος.
4. **m** (Mean Value) είναι η μέση τιμή των λύσεων του αλγορίθμου.
5. **σ** (Standard Deviation) είναι η τυπική απόκλιση των λύσεων του αλγορίθμου.
6. **Average m** (Average Mean Value) είναι η μέση τιμή της απόκλισης της λύσης του προβλήματος από τη καλύτερη λύση της βιβλιογραφίας.
7. **Average σ** (Average Standard Deviation) είναι η τυπική απόκλιση της απόκλισης της λύσης του προβλήματος από την καλύτερη λύση της βιβλιογραφίας.

Καθένα από τα 59 προβλήματα που μελετήθηκαν έτρεξε με τις ίδιες παραμέτρους του εσωτερικού γενετικού αλγορίθμου για 25 γενιές χρωμοσωμάτων και 40 χρωμοσώματα σε κάθε γενιά. Τα αποτελέσματα φαίνονται στους παρακάτω πίνακες ανά κατηγορία προβλημάτων.

Τα αρχεία για τα δεδομένα όλων των προβλημάτων προήλθαν από το http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/. Ο επεξεργαστής του υπολογιστή που χρησιμοποιήθηκε είναι Inter Core i7 2GHz.

Ακολουθεί ο πίνακας αποτελεσμάτων των προβλημάτων Christofides:

Πρόβλημα	Cust.	BKV	BV	M	σ	Average m (%)	Average σ (%)
Christofides 01	50	524,61	576,81	590,94	9,33	12,64%	1,78%
Christofides 02	75	835,26	922,86	950,20	14,07	13,76%	1,68%
Christofides 03	100	826,14	965,52	986,10	12,09	19,36%	1,46%
Christofides 04	150	1028,42	1245,41	1264,07	11,82	22,91%	1,15%
Christofides 05	199	1291,29	1551,35	1596,88	21,47	23,67%	1,66%
Christofides 06	50	555,43	579,51	596,47	10,96	7,39%	1,97%
Christofides 07	75	909,68	920,69	942,13	13,75	3,57%	1,51%
Christofides 08	100	865,94	951,30	981,13	15,56	13,30%	1,80%
Christofides 09	150	1162,55	1248,54	1266,33	12,39	8,93%	1,07%
Christofides 10	199	1395,85	1553,34	1580,05	18,66	13,20%	1,34%
Christofides 11	120	1041,11	1122,25	1180,00	33,33	13,34%	3,20%
Christofides 12	100	819,56	940,87	953,59	9,46	16,35%	1,15%
Christofides 13	120	1541,14	1135,01	1183,97	35,55	-23,18%	2,31%
Christofides 14	100	866,37	935,01	955,12	12,26	10,24%	1,41%

Πινακας 8 Πίνακας Αποτελεσμάτων προβλημάτων Christofides

Ακολουθεί ο πίνακας των αποτελεσμάτων των προβλημάτων Golden:

Πρόβλημα	Cust.	BKV	BV	M	σ	Average m (%)	Average σ (%)
Golden 01	240	5623,47	5745,25	5833,76	51,52	3,74%	0,92%
Golden 02	320	8435	9234,79	9285,74	33,25	10,09%	0,39%
Golden 03	400	11036,22	12869,34	12961,98	47,84	17,45%	0,43%
Golden 04	480	13624,52	17033,39	17148,12	113,30	25,86%	0,83%
Golden 05	200	6460,98	9254,77	9321,13	52,94	44,27%	0,82%
Golden 06	280	8412,8	10707,52	10860,91	60,33	29,10%	0,72%
Golden 07	360	10181,75	12245,07	12296,74	41,93	20,77%	0,41%
Golden 08	440	11649,89	13256,77	13351,746	46,28	14,61%	0,38%
Golden 09	255	579,71	691,07	698,165	4,21	20,43%	0,69%
Golden 10	323	737,28	879,32	889,046	5,63	20,58%	0,73%
Golden 11	399	913,35	1103,61	1110,945	4,94	21,63%	0,52%
Golden 12	483	1102,76	1341,13	1354,718	7,23	22,85%	0,63%
Golden 13	252	857,19	994,68	1006,522	6,53	17,42%	0,73%
Golden 14	320	1080,55	1257,89	1278,839	9,20	18,35%	0,81%
Golden 15	396	1338	1577,35	1587,82	8,92	18,67%	0,67%
Golden 16	480	1613,66	1923,11	1935,776	8,46	19,96%	0,52%
Golden 17	240	707,76	818,09	825,758	3,81	16,67%	0,54%
Golden 18	300	995,13	1171,18	1178,939	4,35	18,47%	0,44%
Golden 19	360	1365,32	1593,32	1606,177	8,29	17,64%	0,61%
Golden 20	420	1818,25	2129,39	2149,092	8,30	18,20%	0,46%

Πινακας 9 Πίνακας Αποτελεσμάτων προβλημάτων Golden

Ακολουθεί ο πίνακας αποτελεσμάτων των προβλημάτων Taillard:

Πρόβλημα	Cust.	BKV	BV	m	σ	Average m (%)	Average σ (%)
Taillard 75A	75	1618,36	1865,28	1971,53	60,64	21,82%	3,75%
Taillard 75B	75	1344,64	1554,54	1602,47	23,22	19,17%	1,73%
Taillard 75C	75	1291,01	1480,88	1519,63	22,46	17,71%	1,74%
Taillard 75D	75	1365,42	1543,11	1657,68	58,09	21,40%	4,25%
Taillard 100A	100	2041,34	2363,29	2467,47	61,42	20,87%	3,01%
Taillard 100B	100	1939,9	2300,95	2393,24	51,58	23,37%	2,66%
Taillard 100C	100	1406,2	1672,28	1757,98	41,31	25,02%	2,94%
Taillard 100D	100	1581,25	1904,28	1939,77	23,80	22,67%	1,51%
Taillard 150A	150	3055,23	3655,17	3792,41	73,42	24,13%	2,40%
Taillard 150B	150	2727,2	3303,86	3428,12	90,01	25,70%	3,30%
Taillard 150C	150	2341,84	2905,61	2987,78	85,04	27,58%	3,63%
Taillard 150D	150	2645,4	3128,83	3233,66	47,23	22,24%	1,79%
Taillard 385	385	24366,69	29074,85	29714,01	249,20	21,95%	1,02%

Πινακας 10 Πίνακας Αποτελεσμάτων προβλημάτων Taillard

Ακολουθεί ο πίνακας αποτελεσμάτων των προβλημάτων Li:

Πρόβλημα	Cust.	BKV	BV	m	σ	Average m (%)	Average σ (%)
Li 21	560	16212,74	20989,17	21378,78	185,29	31,86%	1,14%
Li 22	600	14584,42	14932,21	15116,91	78,45	3,65%	0,54%
Li 23	640	18801,12	25021,06	25755,49	337,95	36,99%	1,80%
Li 24	720	21389,33	30312,21	30751,82	296,86	43,77%	1,39%
Li 25	760	16763,72	16320,29	16584,20	134,14	-1,07%	0,80%
Li 26	800	23971,74	36171,12	36477,58	154,94	52,17%	0,65%
Li 27	840	17433,69	17174,94	17405,33	163,93	-0,16%	0,94%
Li 28	880	25656,92	40613,95	41246,88	367,95	60,76%	1,43%
Li 29	960	29154,34	47171,67	47835,77	415,77	64,08%	1,43%
Li 30	1040	31742,51	53292,87	54101,93	490,04	70,44%	1,54%
Li 31	1120	34330,84	56995,17	58423,03	655,92	70,18%	1,91%
Li 32	1200	36919,24	64787,66	65484,66	447,22	77,37%	1,21%

Πινακας 11 Πίνακας Αποτελεσμάτων προβλημάτων Li

Από τα παραπάνω προβλήματα, το εύρος λύσεων κυμάνθηκε από -23% μέχρι 78% ως προς τη βέλτιστη της βιβλιογραφίας. Σε 3 προβλήματα ο αλγόριθμος κατάφερε να βρεί καλύτερη λύση από την εως τώρα βέλτιστη(Christofides 13, Li 25, Li 27). Όπως γίνεται φανερό, ο αλγόριθμος είχε καλύτερα αποτελέσματα σε μερικά προβλήματα και λίγο χειρότερα στα υπόλοιπα, αν και τα γενικότερα αποτελέσματα του αλγορίθμου είναι ενθαρρυντικά.. Αυτό αφενός οφείλεται στις πολύ λίγες γενιές χρωμοσωμάτων που χρησιμοποιήθηκαν και αφετέρου στην χρησιμοποίηση των ίδιων παραμέτρων για κάθε πρόβλημα. Επιλέχθηκε να χρησιμοποιηθούν οι ίδιοι παράμετροι σε κάθε πρόβλημα και να μην γίνει προσπάθεια βελτιστοποίησης του εσωτερικού γενετικού αλγορίθμου, αφού σκοπός της παρούσας διπλωματικής δεν είναι η επίλυση ενός απλού προβλήματος δρομολόγησης αλλά ενός σύνθετου προβλήματος, κοντά σε πραγματικές συνθήκες λειτουργίας, όπως αυτό που θα περιγραφεί στην επόμενη ενότητα.

10 Επίδειξη χρήσης του αλγορίθμου σε Μελέτη Περίπτωσης

Ο αλγόριθμος που υλοποιήθηκε ως στόχο έχει την δημιουργία ενός εργαλείου, με δυνατότητα άμεσης εφαρμογής σε πραγματικές επιχειρήσεις δραστηριοποιούμενες σε αστικά περιβάλλοντα που θέλουν να μειώσουν τα κόστη τους και να υιοθετήσουν μια πράσινη διάσταση στις μεταφορές τους. Για αυτό το λόγο, η παρούσα διπλωματική παρουσιάζει μια μελέτη περίπτωσης(case study), με χρήση συγκεκριμένων στοιχείων για την εξαγωγή των τελικών δρομολογίων μιας τυπικής μέρας στην Αθήνα.

10.1 Παρουσίαση δεδομένων και τρόπος συλλογής

Για την επίδειξη του αλγορίθμου γίνονται οι εξής παραδοχές:

- Μια **κεντρική Αποθήκη depot (id=0,i=0 or j=0)**
- **Δύο είδη καταστημάτων-πελάτες** (Συνολικά N+1) τα οποία διαφοροποιούνται ως προς τη ζήτηση σε :
    a. Μεσαία
    b. Μεγάλα

    Τα **καταστήματα μεσαίου μεγέθους** απαρτίζουν το σύνολο **C_m**, ενώ τα **καταστήματα μεγάλου μεγέθους** το σύνολο **C_b**. Προφανώς, ισχύουν :

    1. $C_m \cup C_u = C$, όπου C={1,2…,N+1}
    2. $C_m \cap C_u = \emptyset$

    Η ζήτηση κάθε πελάτη σχετίζεται με το μέγεθος του:
    - $0 \leq R_i \leq R_m, \forall\, i \in C_m$
    - $R_m < R_i \leq R_b, \forall\, i \in C_u$

    Όπου **R_m η μέγιστη δυνατή ζήτηση μεσαίου καταστήματος** και ταυτόχρονα η ελάχιστη ζήτηση μεγάλου καταστήματος,ενώ **R_b η μέγιστη δυνατή ζήτηση μεγάλου καταστήματος**.

    Κάθε κατάστημα ανήκει σε μια ζώνη κίνησης (traffic zone) z:

    Z= {high (1), medium (2), low (3)}

    Συμβολίζεται iz το κατάστημα I που ανήκει στη ζώνη z.

Κάθε ζώνη έχει διαφορετικά peak hours, τα οποία επηρεάζουν την ταχύτητα κίνησης μέσα στη ζώνη:

1. High {8.00-10.00 & 16.00-18.00}
2. Medium{9.00-11.00 & 16.00-18.00}
3. Low{Ø}

- **Έχουμε 2 τύπους οχημάτων** V={small,medium}={1,2}

1. $v \in V_s$, t_{load}=20minutes,t_{unload}=10minutes Capacity=20boxes
2. $v \in V_m$, t_{load}=60minutes,t_{unload}=30minutes Capacity=64boxes

Το μικρό όχημα έχει σταθερές για τον υπολογισμό των εκπομπών :

α_o	1,576
α_1	-17,6
α_2	0,00117
α_3	36,067

Πινακας 12 Σταθερές εκπομπών μικρού οχήματος

Το μεγάλο όχημα θεωρείται οτι έχει σταθερές 30% μεγαλύτερες από αυτές του μικρού οχήματος. Έτσι:

α_o	2,049
α_1	-22,9
α_2	0,00152
α_3	46,887

Πινακας 13 Σταθερές εκπομπών μεγάλου οχήματος

Αποφασίσθηκε να χρησιμοποιηθεί μια κεντρική αποθήκη εκτός της Αθήνας, όπως ακριβώς συμβαίνει και στην πραγματικότητα, η οποία βρίσκεται στο Σχηματάρι. Ταυτόχρονα, οι πελάτες που πρέπει να εξυπηρετηθούν είναι 13 και βρίσκονται σε διάφορα σημεία της Αττικής σύμφωνα με τον παρακάτω πίνακα:

Κωδικός	Περιοχή	Μέγεθος	Ζήτηση
Depot	Σχηματάρι	Αποθήκη	-
Customer 1	Εξάρχεια	Μεσαίο	6
Customer 2	Σύνταγμα	Μεγάλο	16
Customer 3	Ταύρος	Μεσαίο	5
Customer 4	Αγία Παρασκευή	Μεσαίο	10
Customer 5	Μαρούσι	Μεγάλο	15
Customer 6	Μεταμόρφωση	Μεσαίο	9
Customer 7	Κηφισιά	Μεγάλο	18
Customer 8	Νέο ψυχικό	Μεσαίο	8
Customer 9	Αργυρούπολη	Μεσαίο	10
Customer 10	Καλλιθεα	Μεσαίο	8
Customer 11	Περιστέρι	Μεσαίο	10
Customer 12	Πειραιάς	Μεγάλο	17
Customer 13	Γλυφάδα	Μεγάλο	15

Πινακας 14 Περιοχές Πελατών και Κεντρικής Αποθήκης

Τα οχήματα της εταιρείας που ασχολούνται με την εξυπηρέτηση αυτών των πελατών είναι:

1. 1 όχημα των 20 κιβωτίων.
2. 3 οχήματα των 64 κιβωτίων.

Τα οχήματα των 64 κιβωτίων έχουν τη δυνατότητα να φύγουν με μισό φορτίο(half truck load).

Για την εύρεση των αποστάσεων μεταξύ όλων των πελατών χρησιμοποιήθηκαν οι γεωγραφικές συντεταγμένες μέσω της εφαρμογής Bing Maps. Οι συντεταγμένες παρουσιάζονται στον παρακάτω πίνακα:

Name	Address	Latitude (y)	Longitude (x)
Depot	Schimatari 320 09	38,3096313	23,5809116
Customer 1	Stournari 24, Αθήνα 106 82	37,9876096	23,7307036
Customer 2	Voulis 38, Αθήνα 105 57	37,9743041	23,7319983
Customer 3	Tavros	37,9692001	23,6935005
Customer 4	Leoforos Mesogeion 512, Agia Paraskevi 153 42	38,0118645	23,8373072
Customer 5	Andrea Papandreou 35, Marousi 151 22	38,0449755	23,7918353
Customer 6	Favierou 5, Metamorfosi 144 52	38,0682364	23,7656699
Customer 7	Leoforos Kifisias 287, Kifisia 145 61	38,0750598	23,8120365
Customer 8	Omirou 5, Neo Psichiko 154 51	38,0052485	23,7768982
Customer 9	Leoforos Vouliagmenis 595, Argiroupoli 164 52	37,9073600	23,7454500
Customer 10	Davaki 33, Alimos 174 55	37,9182100	23,7090600
Customer 11	Ethnikis Antistaseos 42, Peristeri 121 34	38,0133307	23,6909226
Customer 12	Ethnikis Antistaseos 21, Dafni 172 37	37,9539305	23,7381886
Customer 13	Glyfada	37,8580017	23,7577991

Πινακας 15 Συντεταγμένες πελατών μέσω Bing Maps

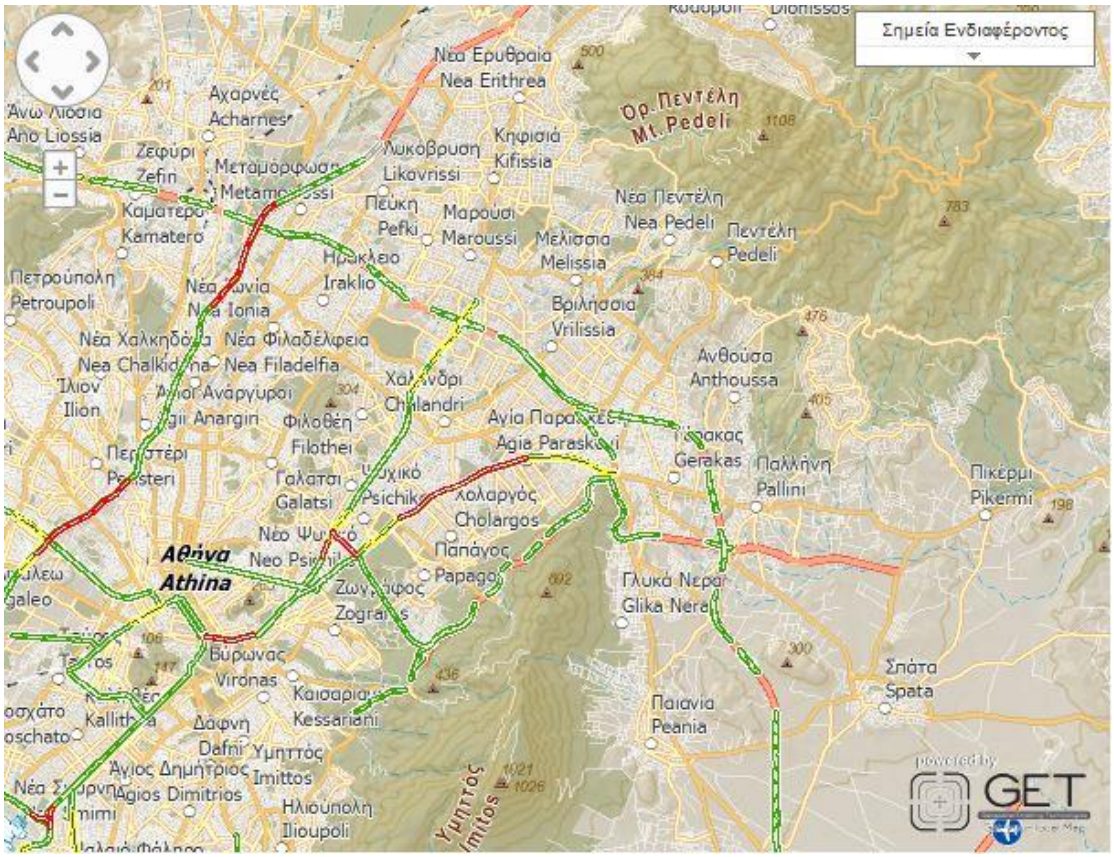
Με βάση τις παραπάνω συντεταγμένες συμπληρώνεται ο πίνακας των αποστάσεων:

	Depot	Cust. 1	Cust. 2	Cust. 3	Cust. 4	Cust. 5	Cust. 6	Cust. 7	Cust. 8	Cust. 9	Cust. 10	Cust. 11	Cust. 12	Cust. 13
Depot		66,27	67,23	66,24	60,92	53,30	49,64	48,10	58,39	80,65	73,59	62,05	76,94	83,53
Cust. 1	61,98		1,85	4,90	11,32	10,18	12,28	18,65	5,96	10,97	9,97	6,78	4,40	15,81
Cust. 2	66,48	2,47		5,10	11,22	12,43	16,78	15,96	6,34	9,36	8,35	8,13	2,79	14,19
Cust. 3	66,34	5,27	5,60		15,42	17,61	16,66	23,03	10,54	11,67	9,01	7,98	5,10	16,51
Cust. 4	61,52	12,00	12,13	16,80		8,67	12,79	13,15	7,62	20,46	20,05	19,00	13,89	26,15
Cust. 5	53,64	11,46	11,61	16,27	8,24		4,59	6,20	5,99	19,94	19,52	15,56	13,37	24,78
Cust. 6	51,25	12,46	16,95	17,09	12,31	5,65		7,95	9,99	30,10	24,45	12,90	19,50	33,25
Cust. 7	49,10	14,63	14,78	22,32	11,41	5,19	5,46		9,16	23,11	22,69	18,13	16,54	27,95
Cust. 8	57,49	6,21	7,31	11,97	5,96	6,49	10,29	10,02		15,64	15,22	14,17	9,07	20,48
Cust. 9	78,69	10,83	9,21	14,66	18,63	19,16	28,09	22,69	13,08		4,98	19,40	5,47	7,66
Cust. 10	73,58	9,78	8,16	9,56	18,40	19,61	23,88	30,25	13,52	6,35		15,20	7,79	9,23
Cust. 11	61,85	7,35	8,31	8,45	17,93	13,10	12,15	18,52	10,80	21,45	15,80		10,86	24,61
Cust. 12	70,56	5,66	4,03	5,45	12,78	13,99	19,96	17,52	7,90	6,79	6,14	11,31		11,63
Cust. 13	82,52	17,31	15,69	18,49	24,44	26,95	31,91	38,28	19,56	6,48	8,81	23,23	11,95	

Πινακας 16 Αποστάσεις μεταξύ πελατών

Το κόστος υπεργολαβίας για τα φορτηγά οποιασδήποτε κατηγορίας λήφθηκε ως 40% υψηλότερο του κόστους των ιδιόκτητων φορτηγών της επιχείρησης.

Για την εκτίμηση των ζωνών κίνησης των διαδρομών μεταξύ των πελατών χρησιμοποιήθηκε ο διαχωρισμός βάσει του παρακάτω χάρτη για μια τυπική μέρα στην Αθήνα:



Οι κόκκινες ζώνες αναπαριστούν τις ζώνες high traffic, οι κίτρινες τις medium traffic, ενώ οι πράσινες τις low traffic zones. Οι ταχύτητες κάθε ζώνης(km/h) εντός και εκτός ωρών αιχμής φαίνονται στον παρακάτω πίνακα:

	Peak Hours	Non Peak Hours
High Traffic	11	17
Medium Traffic	15	22
Low Traffic	30	30

Πινακας 17 Ταχύτητες Ζωνών Κίνησης

Τέλος, θεωρείται οτι η έναρξη της βάρδιας είναι 6.30 π.μ. και η λήξη της 12μ.μ. ενώ η βάρδια χωρίζεται σε 35 μικρότερες περιόδους 30λεπτών η καθεμία.

10.2 Παρουσίαση Αποτελεσμάτων

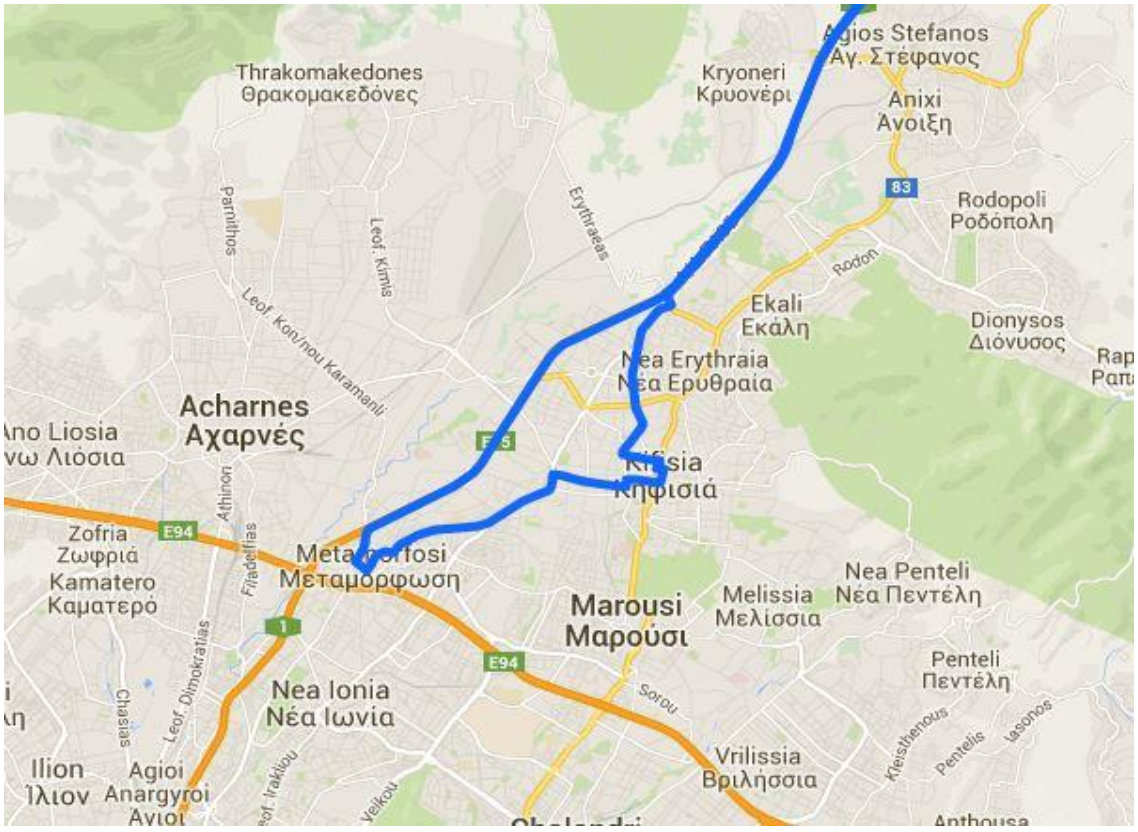
Ο αλγόριθμος εκτελέστηκε για 100 επαναλήψεις του εξωτερικού γενετικού με 20 χρωμοσώματα και 10 επαναλήψεις του εσωτερικού γενετικού με 10 χρωμοσώματα η καθεμία.

Τα αποτελέσματα των δρομολογίων είναι:

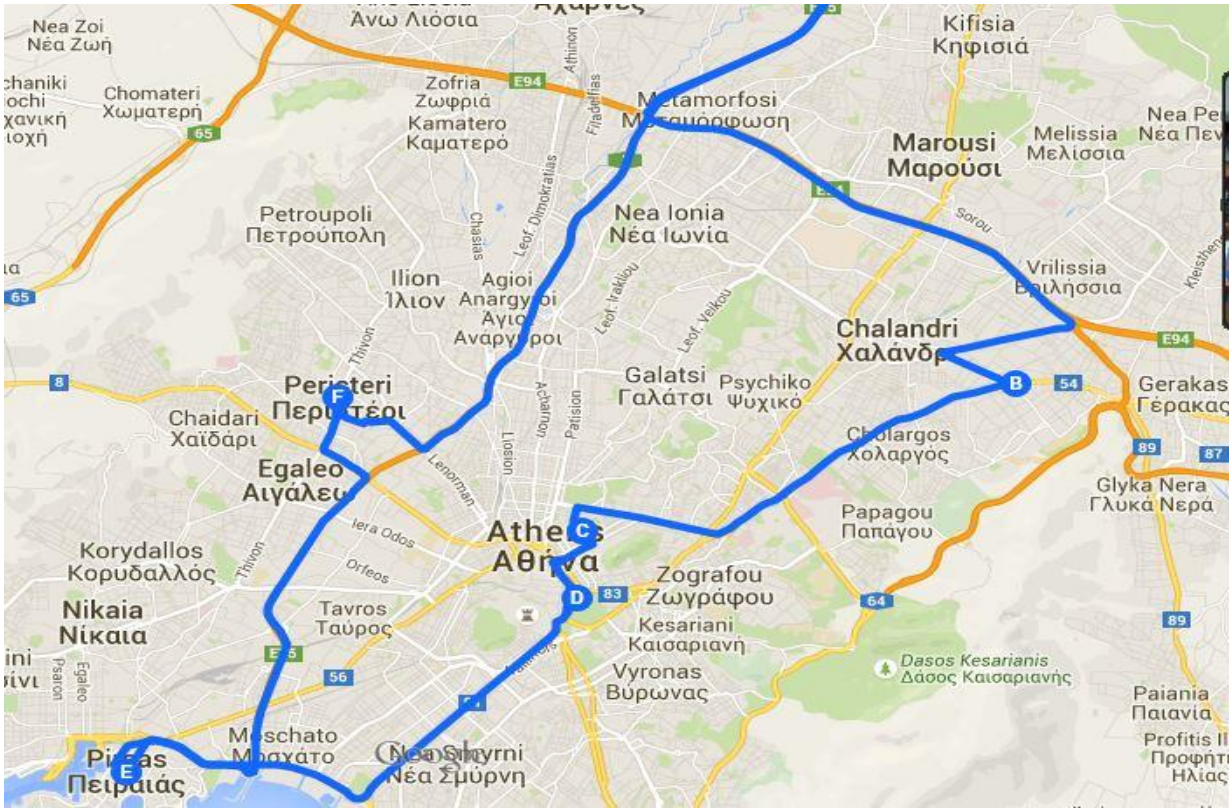
1. Όχημα 1 χωρητικότητας 64 boxes: Κεντρική Αποθήκη - Μεταμόρφωση – Κηφισιά με 27 κιβώτια(half truck load).
2. Όχημα 2 χωρητικότητας 64 boxes: Κεντρική Αποθήκη – Αγία Παρασκευή – Εξάρχεια – Σύνταγμα – Πειραιάς – Περιστέρι – Κεντρική Αποθήκη με 59 κιβώτια.
3. Όχημα 3 χωρητικότητας 64 boxes: Κεντρική Αποθήκη – Ταύρος – Καλλιθέα- Γλυφάδα- Αργυρούπολη – Νέο Ψυχικό – Μαρούσι – Κεντρική Αποθήκη με 61 κιβώτια.
4. Όχημα 4 χωρητικότητας 20 boxes: Δεν χρησιμοποιείται στη λύση.

10.3 Γραφική Απεικόνιση Αποτελεσμάτων

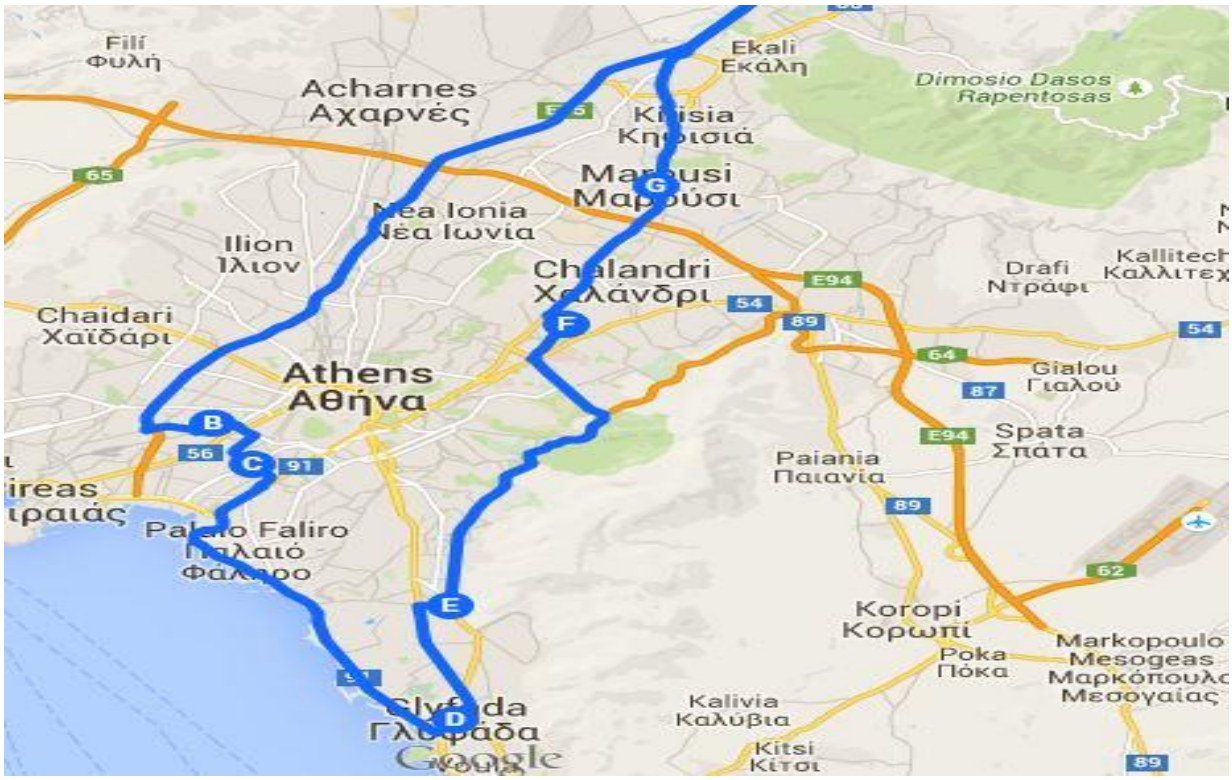
Τα δρομολόγια που δημιουργήθηκαν από τον αλγόριθμο φαίνονται στα παρακάτω σχήματα:



Σχήμα 42 Πρώτο δρομολόγιο



Σχήμα 43 Δεύτερο Δρομολόγιο



Σχήμα 44 Τρίτο Δρομολόγιο

11 Συμπεράσματα

Η δρομολόγηση οχημάτων αφορά την δημιουργία ενός σχεδίου, βάσει του οποίου ένας στόλος οχημάτων θα εξυπηρετήσει ένα σύνολο πελατών, με σκοπό την επίτευξη του μεγαλύτερου κέρδους ή του μικρότερου κόστους. Το σχέδιο καθορίζει τους πελάτες που θα εξυπηρετηθούν από κάθε όχημα, αλλά και τη σειρά με την οποία θα εξυπηρετηθούν προκειμένου να επιτευχθεί το μεγαλύτερο δυνατό όφελος, ικανοποιώντας ταυτόχρονα τους εκάστοτε περιορισμούς του προβλήματος.

Τα προβλήματα δρομολόγησης μελετώνται εκτενώς τα τελευταία χρόνια, λόγω του μεγάλου ποσοστού του κόστους των προϊόντων για το οποίο ευθύνονται οι μεταφορές. Μικρές βελτιώσεις του κόστους αυτού ενδέχεται να προκαλέσουν σημαντική μείωση του τελικού κόστους των προϊόντων, γεγονός που οδηγεί στην ανάπτυξη νέων αποδοτικότερων τεχνικών δρομολόγησης.

Στην παρούσα διπλωματική εργασία, μετά τη μελέτη των προβλημάτων της βιβλιογραφίας, αποφασίστηκε να αναπτυχθεί ένα νέο μοντέλο προβλήματος δρομολόγησης, εμπνευσμένο από το αστικό περιβάλλον της Αθήνας. Το μοντέλο που αναπτύχθηκε λαμβάνει υπόψη του περιορισμούς κίνησης, ωρών αιχμής αλλά και την περιβαλλοντική διάσταση της δρομολόγησης οχημάτων, υιοθετώντας ένα ανώτατο όριο για τις εκπομπές διοξειδίου του άνθρακα του στόλου των οχημάτων μιας εταιρείας. Τέλος, λήφθηκε υπόψη η δυνατότητα υπεργολαβίας(outsourcing) οχημάτων από εταιρείες logistics, όταν ο στόλος οχημάτων δεν επαρκεί.

Το μοντέλο αυτό μπορεί να βρει άμεσα εφαρμογή σε πολλές επιχειρήσεις, που δραστηριοποιούνται σε αστικά περιβάλλοντα. Για την επίλυση του μετά τη μελέτη της βιβλιογραφίας, αποφασίστηκε η υλοποίηση ενός συνδυαστικού υβριδικού μεταευρετικού αλγορίθμου, αποτελούμενου από ένα εσωτερικό και ένα εξωτερικό γενετικό αλγόριθμο, οι οποίοι συνεργάζονται μεταξύ τους και δημιουργούν πολύ καλής ποιότητας λύσεις.

Για την επαλήθευση του αλγορίθμου που υλοποιήθηκε χρησιμοποιήθηκαν προβλήματα αναφοράς(benchmark problems) από τη βιβλιογραφία, στα οποία ο αλγόριθμος κατάφερε να βρει πολύ καλές λύσεις πολύ κοντά στις βέλτιστες. Τα αποτελέσματα του συνδυασμένου αλγορίθμου είναι ενθαρρυντικά, για την περαιτέρω μελέτη του συνδυασμού τέτοιων αλγορίθμων.

Το πρόβλημα που επιλύθηκε και οι αλγόριθμοι που χρησιμοποιήθηκαν μπορούν να βελτιωθούν με περαιτέρω έρευνα. Ενδεικτικά δίνονται ορισμένες κατευθύνσεις που μπορούν να ακολουθηθούν. Αρχικά, το πρόβλημα μπορεί να προσομοιώσει ακόμα καλύτερα πραγματικά σενάρια λειτουργίας, με χρήση κατανάλωσης καυσίμου και σταθμών ανεφοδιασμού σε διάφορα σημεία μέσα στην πόλη. Επιπλέον, η υιοθέτηση αυστηρών παραθύρων χρόνου για τους πελάτες ενδέχεται να βελτιώσουν ακόμη περισσότερο το πρόβλημα, αφού τα περισσότερα καταστήματα-πελάτες δέχονται παραλαβές, μόνο συγκεκριμένες ώρες μέσα στη μέρα.

Οι γενετικοί αλγόριθμοι, από την άλλη, μπορούν να συνδυασθούν με τον αλγόριθμο προσομοίωσης ανόπτησης ή ακόμη και με κάποιο άλλο μεταευρετικό αλγόριθμο για ακόμη καλύτερα αποτελέσματα . Επιπρόσθετα, η χρήση διαφορετικών μεθόδων κατασκευής αρχικού πληθυσμού μπορεί να βελτίωσει σημαντικά την ποιότητα των παραγόμενων λύσεων. Τέλος, η χρησιμοποίηση παράλληλων γενιών στους γενετικούς αλγορίθμους είναι μια νέα μέθοδος που μπορεί να χρησιμοποιηθεί στο υπάρχον πρόβλημα.

Τέλος, η δυνατότητα χρήσης νέων μεθόδων συλλογής δεδομένων, όπως η χρήση ζωντανών δεδομένων κίνησης στην πόλη μπορούν να τροφοδοτήσουν το πρόβλημα με ακόμη περισσότερα στοιχεία και τα αποτελέσματα να βελτιωθούν ακόμη περισσότερο. Επιπρόσθετα κρίνεται αναγκαία η δημιουργία διεπαφής(interface) χρήστη μηχανής για τη εισαγωγή των δεδομένων στο σύστημα. Έτσι, το πράσινο πρόβλημα δρομολόγησης με στόλο ετερογενών οχημάτων δραστηριοποιούμενο σε αστικό περιβάλλον μπορεί να βρει άμεσα εφαρμογή σε πραγματικές επιχειρήσεις που θέλουν να χρησιμοποιήσουν σύγχρονες μεθόδους δρομολόγησης για τις μεταφορές τους.

12 Βιβλιογραφία

Agarwal, Y., K. Mathur and H. M. Salkin (1989). "A set-partitioning-based exact algorithm for the vehicle routing problem." Networks **19**(7): 731-749.
Altınel, İ. K. and T. Öncan (2005). "A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem." Journal of the Operational Research Society **56**(8): 954-961.
Alvarenga, G. B., R. M. De Abreu Silva and G. R. Mateus (2005). A hybrid approach for the dynamic vehicle routing problem with time windows. Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on, IEEE.
Amberg, A., W. Domschke and S. Voß (2000). "Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees." European Journal of Operational Research **124**(2): 360-376.
Archetti, C., M. W. Savelsbergh and M. G. Speranza (2006). "Worst-case analysis for split delivery vehicle routing problems." Transportation Science **40**(2): 226-234.
Archetti, C. and M. Speranza (2012). "Vehicle routing problems with split deliveries." International transactions in operational research **19**(1-2): 3-22.
Archetti, C., M. G. Speranza and A. Hertz (2006). "A tabu search algorithm for the split delivery vehicle routing problem." Transportation Science **40**(1): 64-73.
Bäck, T. (1992). The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm. PPSN.
Baker, B. M. and M. Ayechew (2003). "A genetic algorithm for the vehicle routing problem." Computers & Operations Research **30**(5): 787-800.
Baldacci, R. and A. Mingozzi (2009). "A unified exact method for solving different classes of vehicle routing problems." Mathematical Programming **120**(2): 347-380.
Baldacci, R., A. Mingozzi and R. Roberti (2012). "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints." European Journal of Operational Research **218**(1): 1-6.
Baldacci, R., P. Toth and D. Vigo (2007). "Recent advances in vehicle routing exact algorithms." 4OR **5**(4): 269-298.
Baowen, C., S. Shen-min, C. Xinglin and S. Zhizhong (2006). A Multi-Ant Colony System for Vehicle Routing Problems. Control Conference, 2006. CCC 2006. Chinese, IEEE.
Baptista, S., R. C. Oliveira and E. Zúquete (2002). "A period vehicle routing case study." European Journal of Operational Research **139**(2): 220-229.
Bard, J. F., L. Huang, M. Dror and P. Jaillet (1998). "A branch and cut algorithm for the VRP with satellite facilities." IIE transactions **30**(9): 821-834.
Bard, J. F., L. Huang, P. Jaillet and M. Dror (1998). "A decomposition approach to the inventory routing problem with satellite facilities." Transportation science **32**(2): 189-203.
Battarra, M., J.-F. Cordeau and M. Iori (2014). "Pickup-and-Delivery Problems for Goods Transportation." Vehicle Routing: Problems, Methods, and Applications **18**: 161.
Belmecheri, F., C. Prins, F. Yalaoui and L. Amodeo (2010). Particle swarm optimization to solve the vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, IEEE.
Belmecheri, F., C. Prins, F. Yalaoui and L. Amodeo (2013). "Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows." Journal of intelligent manufacturing **24**(4): 775-789.
Berger, J. and M. Barkaoui (2003). A hybrid genetic algorithm for the capacitated vehicle routing problem. Genetic and Evolutionary Computation—GECCO 2003, Springer.
Bertsimas, D. J. (1992). "A vehicle routing problem with stochastic demand." Operations Research **40**(3): 574-585.
Bland, R. G. and D. F. Shallcross (1989). "Large travelling salesman problems arising from experiments in X-ray crystallography: a preliminary report on computation." Operations Research Letters **8**(3): 125-128.
Brandão, J. (2011). "A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem." Computers & Operations Research **38**(1): 140-151.
Brandao, J. and A. Mercer (1997). "A tabu search algorithm for the multi-trip vehicle routing and scheduling problem." European journal of operational research **100**(1): 180-191.
Bräysy, O. and M. Gendreau (2005). "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms." Transportation science **39**(1): 104-118.
Campbell, A. M. and M. Savelsbergh (2004). "Efficient insertion heuristics for vehicle routing and scheduling problems." Transportation science **38**(3): 369-378.
Cattaruzza, D., N. Absi, D. Feillet, O. Guyon and X. Libeaut (2014). The multi-trip vehicle routing problem with time windows and release dates. 10th Metaheuristics International Conference (MIC 2013).
Chen, A.-l., G.-k. Yang and Z.-m. Wu (2006). "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem." Journal of Zhejiang University Science A **7**(4): 607-614.
Chen, J., C. Chen, B. Chen, M. Lay, L. Liu, B. Lee, S. Huang, W. Chang and D. Huang (2000). Application of Vehicle Routing Problem with Hard Time Window Constraints. The 29th International Conference Computers and Industrial Engineering.

Chopra, S. and P. Meindl (2007). Supply chain management. Strategy, planning & operation, Springer.
Christiansen, C. H. and J. Lysgaard (2007). "A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands." Operations Research Letters **35**(6): 773-781.
Christofides, N. and J. E. Beasley (1984). "The period routing problem." Networks **14**(2): 237-256.
Cordeau, J.-F. and G. d. é. e. d. r. e. a. d. décisions (2000). The VRP with time windows, Montréal: Groupe d'études et de recherche en analyse des décisions.
Cordeau, J.-F., M. Gendreau, A. Hertz, G. Laporte and J.-S. Sormany (2005). New heuristics for the vehicle routing problem, Springer.
Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin and F. Semet (2002). "A guide to vehicle routing heuristics." Journal of the Operational Research society: 512-522.
Cordeau, J.-F., G. Laporte and A. Mercier (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." Journal of the Operational research society **52**(8): 928-936.
Cordeau, J.-F., G. Laporte, M. W. Savelsbergh and D. Vigo (2006). "Vehicle routing." Transportation, handbooks in operations research and management science **14**: 367-428.
Crainic, T. G., G. Perboli, S. Mancini and R. Tadei (2010). "Two-echelon vehicle routing problem: a satellite location analysis." Procedia-Social and Behavioral Sciences **2**(3): 5944-5955.
Dal, R. v. (1992). Special cases of the traveling salesman problem, Rijksuniversiteit Groningen.
Dantzig, G. B. and J. H. Ramser (1959). "The truck dispatching problem." Management science **6**(1): 80-91.
de Magalhães, J. M. and J. P. De Sousa (2006). "Dynamic VRP in pharmaceutical distribution—a case study." Central European Journal of Operations Research **14**(2): 177-192.
Dominguez, O., A. A. Juan, B. Barrios, J. Faulin and A. Agustin (2014). "Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet." Annals of Operations Research: 1-22.
Dong, L. W. and C. T. Xiang (2006). Ant colony optimization for VRP and mail delivery problems. Industrial Informatics, 2006 IEEE International Conference on, IEEE.
Dorigo, M. and T. Stützle (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. Handbook of metaheuristics, Springer**:** 250-285.
Dorigo, M. and T. Stützle (2010). Ant colony optimization: overview and recent advances. Handbook of metaheuristics, Springer**:** 227-263.
Dror, M., G. Laporte and P. Trudeau (1989). "Vehicle routing with stochastic demands: Properties and solution frameworks." Transportation science **23**(3): 166-176.
Dueck, G. and T. Scheuer (1990). "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing." Journal of computational physics **90**(1): 161-175.
Eksioglu, B., A. V. Vural and A. Reisman (2009). "The vehicle routing problem: A taxonomic review." Computers & Industrial Engineering **57**(4): 1472-1483.
Elhamifar, E. and R. Vidal (2013). "Sparse subspace clustering: Algorithm, theory, and applications." Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(11): 2765-2781.
Erdoğan, S. and E. Miller-Hooks (2012). "A green vehicle routing problem." Transportation Research Part E: Logistics and Transportation Review **48**(1): 100-114.
Escobar, J. W., R. Linfati, P. Toth and M. G. Baldoquin (2014). "A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem." Journal of Heuristics **20**(5): 483-509.
Ferrucci, F., S. Bock and M. Gendreau (2013). "A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods." European Journal of Operational Research **225**(1): 130-141.
Fisher, M. L. (1994). "Optimal solution of vehicle routing problems using minimum k-trees." Operations research **42**(4): 626-642.
Fleszar, K., I. H. Osman and K. S. Hindi (2009). "A variable neighbourhood search algorithm for the open vehicle routing problem." European Journal of Operational Research **195**(3): 803-809.
Francis, P. M., K. R. Smilowitz and M. Tzur (2008). The period vehicle routing problem and its extensions. The vehicle routing problem: latest advances and new challenges, Springer**:** 73-102.
Fu, Z., R. Eglese and L. Li (2006). "A new tabu search heuristic for the open vehicle routing problem." Journal of the Operational Research Society **57**(8): 1018-1018.
Fu, Z., R. Eglese and L. Y. Li (2005). "A new tabu search heuristic for the open vehicle routing problem." Journal of the operational Research Society **56**(3): 267-274.
Geem, Z. W., J. H. Kim and G. Loganathan (2001). "A new heuristic optimization algorithm: harmony search." Simulation **76**(2): 60-68.
Gendreau, M., F. Guertin, J.-Y. Potvin and E. Taillard (1999). "Parallel tabu search for real-time vehicle routing and dispatching." Transportation science **33**(4): 381-390.
Gendreau, M., A. Hertz and G. Laporte (1992). "New insertion and postoptimization procedures for the traveling salesman problem." Operations Research **40**(6): 1086-1094.
Gendreau, M., G. Laporte, C. Musaraganyi and É. D. Taillard (1999). "A tabu search heuristic for the heterogeneous fleet vehicle routing problem." Computers & Operations Research **26**(12): 1153-1173.

Gendreau, M., G. Laporte and R. Séguin (1996). "Stochastic vehicle routing." European Journal of Operational Research **88**(1): 3-12.
Geoffrion, A. M. (1974). Lagrangean relaxation for integer programming, Springer.
Gillett, B. E. and J. G. Johnson (1976). "Multi-terminal vehicle-dispatch algorithm." Omega **4**(6): 711-718.
Glover, F. (1989). "Tabu search-part I." ORSA Journal on computing **1**(3): 190-206.
Glover, F. and M. Laguna (2013). Tabu Search∗, Springer.
Gong, Y.-J., J. Zhang, O. Liu, R.-Z. Huang, H. S.-H. Chung and Y.-H. Shi (2012). "Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **42**(2): 254-267.
Graham, R. L., E. L. Lawler, J. K. Lenstra and A. R. Kan (1979). "Optimization and approximation in deterministic sequencing and scheduling: a survey." Annals of discrete mathematics **5**: 287-326.
Grefenstette, J., R. Gopal, B. Rosmaita and D. Van Gucht (1985). Genetic algorithms for the traveling salesman problem. Proceedings of the first International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum, New Jersey (160-168).
Grötschel, M., M. Jünger and G. Reinelt (1991). "Optimal control of plotting and drilling machines: a case study." Mathematical Methods of Operations Research **35**(1): 61-84.
Hadjiconstantinou, E., N. Christofides and A. Mingozzi (1995). "A new exact algorithm for the vehicle routing problem based onq-paths andk-shortest paths relaxations." Annals of Operations Research **61**(1): 21-43.
Ho, W., G. T. Ho, P. Ji and H. C. Lau (2008). "A hybrid genetic algorithm for the multi-depot vehicle routing problem." Engineering Applications of Artificial Intelligence **21**(4): 548-557.
Holland, J. H. (1975). "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence."
Hong, B. and C. Liu (2009). A two-stage hybrid heuristic for vehical routing problem with pickups and deliveries. Information Management, Innovation Management and Industrial Engineering, 2009 International Conference on, IEEE.
Hwang, H.-S. (2002). "An improved model for vehicle routing problem with time constraint based on genetic algorithm." Computers & Industrial Engineering **42**(2): 361-369.
Ichoua, S., M. Gendreau and J.-Y. Potvin (2003). "Vehicle dispatching with time-dependent travel times." European journal of operational research **144**(2): 379-396.
Jepsen, M., B. Petersen, S. Spoorendonk and D. Pisinger (2008). "Subset-row inequalities applied to the vehicle-routing problem with time windows." Operations Research **56**(2): 497-511.
Johnson, D. (1996). Asymptotic experimental analysis for the Held–Karp traveling salesman bound DS Johnson* LA McGeochf EE Rothberg. Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms, SIAM.
Jung, S. and B. R. Moon (2002). A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows. GECCO.
Kenyon, A. S. and D. P. Morton (2003). "Stochastic vehicle routing with random travel times." Transportation Science **37**(1): 69-82.
Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi (1983). "Optimization by simulated annealing." science **220**(4598): 671-680.
Koskosidis, Y. A., W. B. Powell and M. M. Solomon (1992). "An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints." Transportation science **26**(2): 69-85.
Kumar, N. (2012). "KARAMBIR; KUMAR, R. A comparative analysis of pmx, cx and ox crossover operators for solving travelling salesman problem." International Journal of Latest Research in Science and Technology, MNK Publication **1**(2): 98-101.
Kuo, Y. and C.-C. Wang (2012). "A variable neighborhood search for the multi-depot vehicle routing problem with loading cost." Expert Systems with Applications **39**(8): 6949-6954.
Labadi, N., C. Prins and M. Reghioui (2008). "A memetic algorithm for the vehicle routing problem with time windows." RAIRO-Operations research **42**(03): 415-431.
Labadie, N. and C. Prins (2012). Vehicle routing nowadays: Compact review and emerging problems. Production systems and supply chain management in emerging countries: best practices, Springer**:** 141-166.
Laporte, G. (1992). "The traveling salesman problem: An overview of exact and approximate algorithms." European Journal of Operational Research **59**(2): 231-247.
Laporte, G. (1992). "The vehicle routing problem: An overview of exact and approximate algorithms." European Journal of Operational Research **59**(3): 345-358.
Laporte, G., M. Gendreau, J. Y. Potvin and F. Semet (2000). "Classical and modern heuristics for the vehicle routing problem." International transactions in operational research **7**(4-5): 285-300.
Laporte, G., F. Louveaux and H. Mercure (1992). "The vehicle routing problem with stochastic travel times." Transportation science **26**(3): 161-170.
Laporte, G., F. V. Louveaux and L. Van Hamme (2002). "An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands." Operations Research **50**(3): 415-423.
Lenstra, J. K. and A. Kan (1981). "Complexity of vehicle routing and scheduling problems." Networks **11**(2): 221-227.

Lenstra, J. K. and A. R. Kan (1975). "Some simple applications of the travelling salesman problem." Operational Research Quarterly: 717-733.
Letchford, A. N., J. Lysgaard and R. W. Eglese (2007). "A branch-and-cut algorithm for the capacitated open vehicle routing problem." Journal of the Operational Research Society **58**(12): 1642-1651.
Lima, C. d. R., M. C. Goldbarg and E. F. G. Goldbarg (2004). "A memetic algorithm for the heterogeneous fleet vehicle routing problem." Electronic Notes in Discrete Mathematics **18**: 171-176.
Lin, S.-W., Z.-J. Lee, K.-C. Ying and C.-Y. Lee (2009). "Applying hybrid meta-heuristics for capacitated vehicle routing problem." Expert Systems with Applications **36**(2): 1505-1512.
Lin, S. (1965). "Computer solutions of the traveling salesman problem." Bell System Technical Journal, The **44**(10): 2245-2269.
Little, J. D., K. G. Murty, D. W. Sweeney and C. Karel (1963). "An algorithm for the traveling salesman problem." Operations research **11**(6): 972-989.
Lourenço, H. R., O. C. Martin and T. Stützle (2003). Iterated local search, Springer.
Maffioli, F. (2003). "The vehicle routing problem: A book review." Quarterly Journal of the Belgian, French and Italian Operations Research Societies **1**(2): 149-153.
Maredia, A. (2010). History, Analysis, and Implementation of Traveling Salesman Problem (TSP) and Related Problems, University of Houston-Downtown.
Marinakis, Y., M. Marinaki and G. Dounias (2010). "A hybrid particle swarm optimization algorithm for the vehicle routing problem." Engineering Applications of Artificial Intelligence **23**(4): 463-472.
Martinhon, C., A. Lucena and N. Maculan (2004). "Stronger k-tree relaxations for the vehicle routing problem." European Journal of Operational Research **158**(1): 56-71.
Matai, R., M. L. Mittal and S. Singh (2010). Traveling salesman problem: An overview of applications, formulations, and solution approaches, INTECH Open Access Publisher.
Mendoza, J. E., B. Castanier, C. Guéret, A. L. Medaglia and N. Velasco (2010). "A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands." Computers & Operations Research **37**(11): 1886-1898.
Miller, B. L. and D. E. Goldberg (1995). "Genetic algorithms, tournament selection, and the effects of noise." Complex Systems **9**(3): 193-212.
Miller, D. L. (1995). "A matching based exact algorithm for capacitated vehicle routing problems." ORSA Journal on Computing **7**(1): 1-9.
Moghaddam, B. F., R. Ruiz and S. J. Sadjadi (2012). "Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm." Computers & Industrial Engineering **62**(1): 306-317.
Mole, R. and S. Jameson (1976). "A sequential route-building algorithm employing a generalised savings criterion." Operational Research Quarterly: 503-511.
Montemanni, R., L. M. Gambardella, A. E. Rizzoli and A. V. Donati (2005). "Ant colony system for a dynamic vehicle routing problem." Journal of Combinatorial Optimization **10**(4): 327-343.
Nagata, Y., O. Bräysy and W. Dullaert (2010). "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows." Computers & Operations Research **37**(4): 724-737.
Novoa, C. and R. Storer (2009). "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands." European Journal of Operational Research **196**(2): 509-515.
Ombuki-Berman, B. and F. T. Hanshar (2009). Using genetic algorithms for multi-depot vehicle routing. Bio-inspired algorithms for the vehicle routing problem, Springer**:** 77-99.
Ombuki, B., M. Nakamura and O. Maeda (2002). A hybrid search based on genetic algorithms and tabu search for vehicle routing. 6th IASTED Intl. Conf. On Artificial Intelligence and Soft Computing (ASC 2002).
Ombuki, B., B. J. Ross and F. Hanshar (2006). "Multi-objective genetic algorithms for vehicle routing problem with time windows." Applied Intelligence **24**(1): 17-30.
Paessens, H. (1988). "The savings algorithm for the vehicle routing problem." European Journal of Operational Research **34**(3): 336-344.
Park, Y. and S. Hong (2003). "A performance evaluation of vehicle routing heuristics in a stochastic environment." INTERNATIONAL JOURNAL OF INDUSTRIAL ENGINEERING-THEORY APPLICATIONS AND PRACTICE **10**(4): 435-441.
Penna, P. H. V., A. Subramanian and L. S. Ochi (2013). "An iterated local search heuristic for the heterogeneous fleet vehicle routing problem." Journal of Heuristics **19**(2): 201-232.
Petch, R. J. and S. Salhi (2003). "A multi-phase constructive heuristic for the vehicle routing problem with multiple trips." Discrete Applied Mathematics **133**(1): 69-92.
Pillac, V., M. Gendreau, C. Guéret and A. L. Medaglia (2013). "A review of dynamic vehicle routing problems." European Journal of Operational Research **225**(1): 1-11.
Potvin, J.-Y. and S. Bengio (1996). "The vehicle routing problem with time windows part II: genetic search." INFORMS journal on Computing **8**(2): 165-172.
Prins, C. (2004). "A simple and effective evolutionary algorithm for the vehicle routing problem." Computers & Operations Research **31**(12): 1985-2002.
Psaraftis, H. N. (1995). "Dynamic vehicle routing: Status and prospects." annals of Operations Research **61**(1): 143-164.

Ratliff, H. D. and A. S. Rosenthal (1983). "Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem." Operations Research **31**(3): 507-521.
Renaud, J. and F. F. Boctor (2002). "A sweep-based algorithm for the fleet size and mix vehicle routing problem." European Journal of Operational Research **140**(3): 618-628.
Repoussis, P. P., C. D. Tarantilis, O. Bräysy and G. Ioannou (2010). "A hybrid evolution strategy for the open vehicle routing problem." Computers & Operations Research **37**(3): 443-455.
Roberts, D. and E. Hadjiconstantinou (1998). Algorithmic developments in stochastic vehicle routing. Operations Research Proceedings 1997, Springer**:** 156-161.
Rochat, Y. and É. D. Taillard (1995). "Probabilistic diversification and intensification in local search for vehicle routing." Journal of heuristics **1**(1): 147-167.
Rosenkrantz, D. J., R. E. Stearns and I. Lewis, Philip M (1977). "An analysis of several heuristics for the traveling salesman problem." SIAM journal on computing **6**(3): 563-581.
Salhi, S. and R. Petch (2007). "A GA based heuristic for the vehicle routing problem with multiple trips." Journal of Mathematical Modelling and Algorithms **6**(4): 591-613.
Schneider, M., A. Stenger and D. Goeke (2014). "The electric vehicle-routing problem with time windows and recharging stations." Transportation Science **48**(4): 500-520.
Şen, A. and K. Bülbül (2008). "A survey on multi trip vehicle routing problem."
Shen, Z., F. Ordónez and M. M. Dessouky (2009). The stochastic vehicle routing problem for minimum unmet demand. Optimization and logistics challenges in the enterprise, Springer**:** 349-371.
Solomon, M. M. (1984). Vehicle routing and scheduling with time window constraints: Models and algorithms.
Solomon, M. M. (1987). "Algorithms for the vehicle routing and scheduling problems with time window constraints." Operations research **35**(2): 254-265.
Soonpracha, K., A. Mungwattana, G. K. Janssens and T. Manisri (2014). "Heterogeneous VRP review and conceptual framework."
Srinivas, M. and L. M. Patnaik (1994). "Adaptive probabilities of crossover and mutation in genetic algorithms." Systems, Man and Cybernetics, IEEE Transactions on **24**(4): 656-667.
Subramanian, A., L. M. d. A. Drummond, C. Bentes, L. S. Ochi and R. Farias (2010). "A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery." Computers & Operations Research **37**(11): 1899-1911.
Subramanian, A., P. H. V. Penna, E. Uchoa and L. S. Ochi (2012). "A hybrid algorithm for the heterogeneous fleet vehicle routing problem." European Journal of Operational Research **221**(2): 285-295.
Suman, B. and P. Kumar (2006). "A survey of simulated annealing as a tool for single and multiobjective optimization." Journal of the operational research society **57**(10): 1143-1160.
Sungur, I., F. Ordónez and M. Dessouky (2008). "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty." IIE Transactions **40**(5): 509-523.
Syswerda, G. (1991). "A study of reproduction in generational and steady state genetic algorithms." Foundations of genetic algorithms **2**: 94-101.
Tan, K. C., L. H. Lee, Q. Zhu and K. Ou (2001). "Heuristic methods for vehicle routing problem with time windows." Artificial intelligence in Engineering **15**(3): 281-295.
Tao, G. and Z. Michalewicz (1998). Inver-over operator for the TSP. Parallel Problem Solving from Nature—PPSN V, Springer.
Thangiah, S. R. (1993). Vehicle routing with time windows using genetic algorithms, Citeseer.
Thangiah, S. R., I. H. Osman, R. Vinayagamoorthy and T. Sun (1993). "Algorithms for the vehicle routing problems with time deadlines." American Journal of Mathematical and Management Sciences **13**(3-4): 323-355.
Thangiah, S. R., R. Vinayagamoorty and A. V. Gubbi (1993). Vehicle routing and time deadlines using genetic and local algorithms. Proceedings of the 5th international Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc.
Thompson, P. M. and H. N. Psaraftis (1993). "Cyclic transfer algorithm for multivehicle routing and scheduling problems." Operations research **41**(5): 935-946.
Tillman, F. A. and T. M. Cain (1972). "An upperbound algorithm for the single and multiple terminal delivery problem." Management Science **18**(11): 664-682.
Toth, P. and D. Vigo (2002). "Models, relaxations and exact approaches for the capacitated vehicle routing problem." Discrete Applied Mathematics **123**(1): 487-512.
Üçoluk, G. (1997). "Genetic algorithm solution of the tsp avoiding special crossover and mutation."
Üçoluk, G. (2002). "Genetic algorithm solution of the TSP avoiding special crossover and mutation." Intelligent Automation & Soft Computing **8**(3): 265-272.
Valenzuela, C. L. and A. J. Jones (1997). "Estimating the Held-Karp lower bound for the geometric TSP." European journal of operational research **102**(1): 157-175.
Van Breedam, A. (1994). An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints, RUCA.
van Hemert, J. I. and J. A. La Poutré (2004). Dynamic routing problems with fruitful regions: Models and evolutionary computation. Parallel Problem Solving from Nature-PPSN VIII, Springer.

Verweij, B., S. Ahmed, A. J. Kleywegt, G. Nemhauser and A. Shapiro (2003). "The sample average approximation method applied to stochastic routing problems: a computational study." Computational Optimization and Applications **24**(2-3): 289-333.
Wang, W.-L., H.-P. Huang, Y.-W. Zhao and J.-L. Zhang (2011). "Dynamic customer demand VRP with soft time windows based on vehicle sharing." Computer Integrated Manufacturing Systems **17**(5): 1056-1063.
Wang, W., B. Wu, Y. Zhao and D. Feng (2006). Particle swarm optimization for open vehicle routing problem. Computational Intelligence, Springer**:** 999-1007.
Watterson, G., W. J. Ewens, T. Hall and A. Morgan (1982). "The chromosome inversion problem." Journal of Theoretical Biology **99**(1): 1-7.
Whitley, D. (1994). "A genetic algorithm tutorial." Statistics and computing **4**(2): 65-85.
Wren, A. and A. Holliday (1972). "Computer scheduling of vehicles from one or more depots to a number of delivery points." Operational Research Quarterly: 333-344.
Xiao, Z. and W. Jiang-qing (2012). "Hybrid Ant Algorithm and Applications for Vehicle Routing Problem." Physics Procedia **25**: 1892-1899.
Yang, X.-S. (2010). Nature-inspired metaheuristic algorithms, Luniver press.
Yeun, L. C., W. Ismail, K. Omar and M. Zirour (2008). "Vehicle routing problem: models and solutions." Journal of Quality Measurement and Analysis **4**(1): 205-218.
Yu, B. and Z. Z. Yang (2011). "An ant colony optimization model: The period vehicle routing problem with time windows." Transportation Research Part E: Logistics and Transportation Review **47**(2): 166-181.
Zachariadis, E. E. and C. T. Kiranoudis (2010). "An open vehicle routing problem metaheuristic for examining wide solution neighborhoods." Computers & Operations Research **37**(4): 712-723.
Zachariadis, E. E. and C. T. Kiranoudis (2011). "A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries." Expert Systems with Applications **38**(3): 2717-2726.
Zeng, L., H. Ong and K. Ng (2007). "A generalized crossing local search method for solving vehicle routing problems." Journal of the Operational Research Society **58**(4): 528-532.

Παράρτημα

Υλοποιημένος Κώδικας

Μέθοδοι εξωτερικού γενετικού

GeneticExterior

```
public List<List<List<Int32>>> GeneticExtStart(customer[] allcustomers,inputdata
data1,int number,int iterations)
        {
            List<List<List<Int32>>> FinalSolution= new List<List<List<Int32>>>();
            numberofCategories = data1.GetNumberofCat();
            numberofChromo = number;
            auxiliary aux=new auxiliary();
            chromosome[] InitialPopulation = new chromosome[numberofChromo];
            chromosome[] Allpopulation = new chromosome[2 *
(InitialPopulation.Length)];       //synolikos pli8ismos
            chromosome[] NewPopulation = new chromosome[InitialPopulation.Length];
            double[] Costing = new double[Allpopulation.Length];
            double[] CapacityVehicles = new double[numberofCategories];
            int[] NumberVehicles = new int[numberofCategories];
            CapacityVehicles = data1.CapacityReturn();
            NumberVehicles = data1.NumberingReturn();
            int numberofChromonew = data1.Getnochromonew();
            int Pcrossover=100;
            int Pmutation=75;
            for (o = 0; o < 20;o++ )
            {
                InitialPopulation[o] =
RandomExterior(allcustomers,numberofCategories);
            }
            int dimension = InitialPopulation[0].genes.Length;
            for (q = 0; q < iterations;q++ )
            {
                Console.WriteLine(q);
                for (s=0;s<InitialPopulation.Length;s=s+2)
                {
                    int spot1 = randomizer111.rInt2(0, dimension);
                    int spot3 = randomizer111.rInt2(0, dimension);
                    int spot4 = randomizer111.rInt2(0, dimension);
                    if (spot4 < spot3)
                    {
                        int temp2 = spot4;
                        spot4 = spot3;
                        spot3 = temp2;
                    }
                    int probability1 = randomizer111.rInt2(0, 100);
//gia crossover
                    int probability2 = randomizer111.rInt2(0, 100);
//gia mutation
```

```
                    int probability3 = randomizer111.rInt2(0, 100);
//gia mutation2

                    if (probability1<Pcrossover)
                    {
                        NewPopulation[s] =
aux.ExteriorCrossover(InitialPopulation[s], InitialPopulation[s + 1], spot1);
                        NewPopulation[s + 1] =
aux.ExteriorCrossover(InitialPopulation[s + 1], InitialPopulation[s], spot1);
                    }
                    if (probability2<Pmutation)
                    {
                        NewPopulation[s] = aux.ExteriorMutation(NewPopulation[s],
spot3, spot4);
                    }
                    if (probability3<Pmutation)
                    {
                        NewPopulation[s + 1] = aux.ExteriorMutation(NewPopulation[s
+ 1], spot3, spot4);
                    }
                }
                for (i=0;i<InitialPopulation.Length;i++)
                {
                    Allpopulation[i] = InitialPopulation[i];
                }
                for (p = InitialPopulation.Length; p < Allpopulation.Length; p++)
                {
                    Allpopulation[p] = NewPopulation[p - InitialPopulation.Length];
                }
                for (i=0;i<Allpopulation.Length; i++)
                {
                    temporary = 0;
                    for (kount=0;kount<numberofCategories;kount++)
                    {
                        List<List<Int32>> beta = new List<List<Int32>>();
                        Cost = 0;
                        List<customer> pelates = new List<customer>();
                        for (count = 0; count <
Allpopulation[i].genes.Length;count++ )
                        {
                            if (Allpopulation[i].genes[count] == kount)
                            {
                                pelates.Add(allcustomers[count+1]);
                            }
                        }
                        customer[] pelates1=pelates.ToArray();
                        GA genetic = new GA();

                            beta = genetic.GeneticStart(numberofChromonew,
pelates1, data1, CapacityVehicles[kount],allcustomers[0]);
                            Cost = aux.TotCostfinder(beta, pelates1,
data1,allcustomers[0]);
                            if (beta.Count>NumberVehicles[kount])
                            {
                                Cost = Cost * 100;
                            }
                            temporary = temporary + Cost;
```

```
                    }
                    Costing[i] = temporary;
                }
                Array.Sort(Costing, Allpopulation);
                InitialPopulation = SelectionExterior(Allpopulation,
numberofChromo);
            }

            List<List<customer>> abud = new List<List<customer>>();
            for (kount = 0; kount < numberofCategories; kount++)
            {
                List<List<Int32>> beta = new List<List<Int32>>();
                List<customer> pelates = new List<customer>();
                for (count = 0; count < Allpopulation[0].genes.Length; count++)
                {
                    if (Allpopulation[0].genes[count] == kount)
                    {
                        pelates.Add(allcustomers[count + 1]);
                    }
                }
                abud.Add(pelates);
                customer[] pelates1 = pelates.ToArray();
                GA genetic = new GA();
                beta = genetic.GeneticStart(numberofChromonew, pelates1, data1,
CapacityVehicles[kount],allcustomers[0]);
                FinalSolution.Add(beta);
            }
```

RandomExterior

```
public chromosome RandomExterior(customer[] allcustomers,int kapa)
        {
            chromosome chromosolv = new chromosome(allcustomers.Length - 1);
            for (j = 0; j < chromosolv.genes.Length; j++)
            {
                int rnd = randomizer111.rInt2(0, kapa);
                chromosolv.genes[j] = rnd;
            }

            return chromosolv;
        }
```

Selection

```
public chromosome[] SelectionExterior(chromosome[] Allpop,int spot6)
        {
            chromosome[] FinalPop = new chromosome[spot6];
            for (z = 0; z < spot6;z++ )
            {
                FinalPop[z] = Allpop[z];
            }
                return FinalPop;
        }
```

Crossover

```
  public chromosome ExteriorCrossover(chromosome chromo1, chromosome chromo2, int
spot)
        {
            int dim = chromo1.genes.Length;
            chromosome chromo3 = new chromosome(dim);
            for (i = 0; i < spot;i++)
            {
                chromo3.genes[i] = chromo1.genes[i];
            }
            for (i = spot; i < dim;i++ )
            {
                chromo3.genes[i] = chromo2.genes[i];
            }
                return chromo3;
        }
```

13 Mutation

```
public chromosome ExteriorMutation(chromosome chromo, int spot1,int spot2)
        {
            int temp = chromo.genes[spot1];
            chromo.genes[spot1] = chromo.genes[spot2];
            chromo.genes[spot2] = temp;
            return chromo;
        }
```

Μέθοδοι Εσωτερικού γενετικού αλγορίθμου

Genetic

```
public List<List<Int32>> GeneticStart(int numberOfChromonew, customer[]
allcustomers, inputdata data10,double Capacity,customer customerApothiki)
        {
            chromonew[] InitialPop = new chromonew[numberOfChromonew];
            InitialPop[0] = MolesJames(allcustomers, data10,
Capacity,customerApothiki);
           // InitialPop[0] = Savings(allcustomers, data10,Capacity);
            InitialPop[1] = MolesJames(allcustomers,
data10,Capacity,customerApothiki);
            for (int o=2;o<numberOfChromonew;o++)
            {
                InitialPop[o] = MolesJames(allcustomers,
data10,Capacity,customerApothiki);
            }
            int dimension2 = InitialPop[0].gonidia.Length;

            double[] Cost = new double[2 * (InitialPop.Length)];
//pinakas Cost gia olo to pop
            //List<List<Int32>> a = new List<List<Int32>>();
//epistrefomeno route
            auxiliary auxi = new auxiliary();
            chromonew[] Allpopulation = new chromonew[2 * (InitialPop.Length)];
//synolikos pli8ismos
            chromonew[] NewPopulation = new chromonew[InitialPop.Length];
//neos pli8ismos ka8e epanalipsis
            List<List<Int32>> Routing = new List<List<Int32>>();
//lista gia decode
            ProbMutation = 75;
            VehicleCapacity = Capacity;
        //    TotalProblemDemand = auxi.TotalProblemDemand(allcustomers);

           // Console.WriteLine("Number of Iterations");
            int noIter = 25;
            //Convert.ToInt32(Console.ReadLine());
            for (z = 0; z < noIter; z++)
            {
                Random r = new Random(DateTime.Now.Millisecond);
                for (i = 0; i < InitialPop.Length; i = i + 2)
                {

                        int spot1 = randomizer111.rInt2(0, dimension2);
                        int spot2 = randomizer111.rInt2(0, dimension2);
                        if (spot2 < spot1)
                        {
                            int temp = spot2;
                            spot2 = spot1;
                            spot1 = temp;
                        }
                        int spot3 = randomizer111.rInt2(0, dimension2);
```

```
                        int spot4 = randomizer111.rInt2(0, dimension2);
                        if (spot4 < spot3)
                        {
                            int temp2 = spot4;
                            spot4 = spot3;
                            spot3 = temp2;
                        }
                        int spot5 = randomizer111.rInt2(0, dimension2);
                        int spot6 = randomizer111.rInt2(0, dimension2);
                        if (spot6 < spot5)
                        {
                            int temp3 = spot6;
                            spot6 = spot5;
                            spot5 = temp3;
                        }
                        if (spot2 != spot1)
                        {
                            NewPopulation[i] = auxi.PMXcrossover(InitialPop[i],
InitialPop[i + 1], spot1, spot2);

                        }
                        if (NewPopulation[i] == null)
                        {

                            NewPopulation[i] = InitialPop[i + 1];
                        }
                        int Probability = randomizer111.rInt2(0, 100);
                        if (Probability < ProbMutation)
                        {
                            if (spot3 != spot4)
                            {
                                NewPopulation[i] = auxi.Mutating(NewPopulation[i],
spot3, spot4);
                            }
                            if (spot6 != spot5)
                            {
                                   //    NewPopulation[i] =
auxi.Inversing(allcustomers, NewPopulation[i], spot5, spot6, data10);
                            }
                        }
                        if (spot2 != spot1)
                        {
                            NewPopulation[i + 1] = auxi.PMXcrossover(InitialPop[i +
1], InitialPop[i], spot1, spot2);
                        }
                    if (NewPopulation[i+1]==null)
                    {

                        NewPopulation[i + 1] = InitialPop[i + 1];
                    }
                        if (Probability < ProbMutation)
                        {
                            if (spot3 != spot4)
                            {
                                NewPopulation[i + 1] =
auxi.Mutating(NewPopulation[i + 1], spot3, spot4);
                            }
```

```
                            if (spot6 != spot5)
                            {
                        //      NewPopulation[i + 1] = auxi.Inversing(allcustomers,
NewPopulation[i + 1], spot5, spot6, data10);
                            }
                        }
                }
                for (p = 0; p < InitialPop.Length; p++)
                {
                    Allpopulation[p] = InitialPop[p];
                }
                for (q = InitialPop.Length; q < Allpopulation.Length; q++)
                {
                    Allpopulation[q] = NewPopulation[q - InitialPop.Length];
                }
                MinCost = 10000000;
                for (u = 0; u < Allpopulation.Length; u++)
                {
                    Cost[u] = 0;
                }
                for (i = 0; i < Allpopulation.Length; i++)
                {
                    if (Allpopulation[i] != null)
                    {
                        Routing = auxi.TwoOpting(Allpopulation[i], allcustomers,
VehicleCapacity,customerApothiki);
                        for (j = 0; j < Routing.Count; j++)
                        {
                            Cost[i] = Cost[i] + auxi.CostFinder(Routing[j],
allcustomers, data10,customerApothiki);
                        }
                    }
                    else
                    {
                        Cost[i] = 100000000;
                    }

                }
                Array.Sort(Cost, Allpopulation);
                for (s = 0; s < InitialPop.Length; s++)
                {
                    InitialPop[s] = Allpopulation[s];
                }

            }

            List<List<Int32>> FinalSolution = auxi.TwoOpting(InitialPop[0],
allcustomers, VehicleCapacity,customerApothiki);
            return FinalSolution;
        }
```

Savings

```
public chromonew Savings(customer[] allCustomers, inputdata data9,double Capacity)
        {
            dim = allCustomers.Length;   //diastasi aplwn pinakwn
            dimMax=dim*(dim-1)/2;
            double[] Savings = new double[dimMax];
            int[] iCustomer = new int[dimMax];
            int[] jCustomer = new int[dimMax];
            double[] dist = new double[dim];
            auxiliary aux = new auxiliary();
            VehicleCapacity = Capacity;


            for (i = 1; i < dim; i++)
            {
                for (j = 1; j < dim; j++)
                {
                    if (i > j)
                    {
                        iCustomer[kount] = allCustomers[i].Getidcustomer();
                        jCustomer[kount] = allCustomers[j].Getidcustomer();
                        Savings[kount] = aux.DistanceCustomers(allCustomers[i],
allCustomers[0],data9) + aux.DistanceCustomers(allCustomers[j],
allCustomers[0],data9) - aux.DistanceCustomers(allCustomers[i],
allCustomers[j],data9);
                        kount++;
                    }
                }
            }
            double[] SavingsClone = (double[])Savings.Clone();
            Array.Sort(Savings, iCustomer);
            Array.Sort(SavingsClone, jCustomer);
            Array.Reverse(Savings);
            Array.Reverse(iCustomer);
            Array.Reverse(jCustomer);

            List<Int32> used = new List<Int32>();
            List<List<Int32>> routes = new List<List<Int32>>();
            while (A == -1)
            {
                A = 0;
                List<Int32> a = new List<Int32>();
                CapacityRemaining = VehicleCapacity;
                for (z = 0; z < dimMax; z++)
                {
                    if (used.Contains(iCustomer[z]) && (a.FirstOrDefault() ==
iCustomer[z]))
                    {
                        if (used.Contains(jCustomer[z]) == false)
                        {
                            if (allCustomers[jCustomer[z] - 1].Getdemcustomer() <
CapacityRemaining)
                            {
```

```
                                a.Insert(0, jCustomer[z]);
                                CapacityRemaining = CapacityRemaining -
allCustomers[jCustomer[z] - 1].Getdemcustomer();
                                A = -1;
                                used.Add(jCustomer[z]);
                            }
                        }
                    }
                    else if ((used.Contains(iCustomer[z])) && (a.LastOrDefault() ==
iCustomer[z]))
                    {
                        if (used.Contains(jCustomer[z]) == false)
                        {
                            if (allCustomers[jCustomer[z] - 1].Getdemcustomer() <
CapacityRemaining)
                            {
                                a.Add(jCustomer[z]);
                                CapacityRemaining = CapacityRemaining -
allCustomers[jCustomer[z] - 1].Getdemcustomer();
                                A = -1;
                                used.Add(jCustomer[z]);
                            }
                        }
                    }
                    if ((used.Contains(jCustomer[z])) && ((a.FirstOrDefault() ==
jCustomer[z])))
                    {
                        if (used.Contains(iCustomer[z]) == false)
                        {
                            if (allCustomers[iCustomer[z] - 1].Getdemcustomer() <
CapacityRemaining)
                            {
                                a.Insert(0, iCustomer[z]);
                                CapacityRemaining = CapacityRemaining -
allCustomers[iCustomer[z] - 1].Getdemcustomer();
                                A = -1;
                                used.Add(iCustomer[z]);
                            }
                        }
                    }
                    else if ((used.Contains(jCustomer[z])) && (a.LastOrDefault() ==
jCustomer[z]))
                    {
                        if (used.Contains(iCustomer[z]) == false)
                        {
                            if (allCustomers[iCustomer[z] - 1].Getdemcustomer() <
CapacityRemaining)
                            {
                                a.Add(iCustomer[z]);
                                CapacityRemaining = CapacityRemaining -
allCustomers[iCustomer[z] - 1].Getdemcustomer();
                                A = -1;
                                used.Add(iCustomer[z]);
                            }
                        }
                    }
```

```
                    if ((used.Contains(iCustomer[z]) == false) &&
(used.Contains(jCustomer[z]) == false))
                    {
                        if (allCustomers[iCustomer[z] - 1].Getdemcustomer() +
allCustomers[jCustomer[z] - 1].Getdemcustomer() < CapacityRemaining)
                        {
                            a.Add(iCustomer[z]);
                            a.Add(jCustomer[z]);
                            CapacityRemaining = CapacityRemaining -
(allCustomers[iCustomer[z] - 1].Getdemcustomer() + allCustomers[jCustomer[z] -
1].Getdemcustomer());
                            A = -1;
                            used.Add(jCustomer[z]);
                            used.Add(iCustomer[z]);
                        }
                    }
                }
                if (A == -1)
                {
                    routes.Add(a);
                }
                count++;
            }
            for (i = 2; i < dim + 1; i++)
            {
                if (used.Contains(i) == false)
                {
                    List<Int32> b = new List<Int32>();
                    b.Add(i);
                    used.Add(i);
                    routes.Add(b);
                }
            }
            chromonew chromo19 = new chromonew(dim - 1);
            chromo19 = aux.Encoding(routes, dim - 1);
            return chromo19;
        }
```

14 Randomizing

```
public chromonew Randomizing(customer[] allcustomers)
        {
            List<Int32> a = new List<Int32>();
            chromonew chromosolv = new chromonew(allcustomers.Length - 1);
            Random r = new Random();
            for (i = 1; i < allcustomers.Length; i++)
            {
                a.Add(i + 1);
            }
            int dimensioning = a.Count;
            for (j = 0; j < dimensioning; j++)
            {
                int rnd = r.Next(a.Count);
                chromosolv.gonidia[j] = a[rnd];
                a.RemoveAt(rnd);
            }
            return chromosolv;
        }
```

```
        public chromonew MolesJames(customer[] AllCustomers, inputdata data7,double
Capacity,customer Customer0)
        {
            customer[] allCustomers=new customer[AllCustomers.Length+1];
            dim = allCustomers.Length-1;
            auxiliary aux = new auxiliary();
            VehicleCapacity = Capacity;
            CapacityRemaining = VehicleCapacity;
            List<Int32> unused = new List<Int32>();

            for (int use = 1; use < dim + 1;use++ )
            {
                allCustomers[use] = AllCustomers[use-1];
            }
            allCustomers[0] = Customer0;
            for (i = 1; i < dim + 1; i++)
            {
                    unused.Add(i + 1);
            }
            List<List<Int32>> b = new List<List<Int32>>();
            while (unused.Count != 0)
            {

                List<Int32> a = new List<Int32>();
                CapacityRemaining = Capacity;
                int spot = randomizer111.rInt2(0, unused.Count);
                a.Add(1);
                a.Add(unused[spot]);
                a.Add(1);
                CapacityRemaining = CapacityRemaining - allCustomers[unused[spot] -
1].Getdemcustomer();
                unused.RemoveAt(spot);
                A = -1;
                while (A == -1)
                {
                    maxb = 0;
                    besti = 0;
                    bestj = 0;
                    bestk = 0;

                    for (j = 0; j < unused.Count; j++)
                    {
                        for (z = 1; z < a.Count; z++)
                        {
                            if (CapacityRemaining > allCustomers[unused[j] -
2].Getdemcustomer())
                            {
                                aijk = aux.DistanceCustomers(allCustomers[a[z - 1]
- 1], allCustomers[unused[j] - 1],data7) + aux.DistanceCustomers(allCustomers[a[z]
- 1], allCustomers[unused[j] - 1],data7) - lamda *
(aux.DistanceCustomers(allCustomers[a[z - 1] - 1], allCustomers[a[z] - 1],data7));
                                bijk = mpar * (aux.DistanceCustomers(Customer0,
allCustomers[unused[j] - 1],data7)) - aijk;
                                if (bijk > maxb)
```

```
                                {
                                    besti = z - 1;
                                    bestj = z;
                                    mina = j;
                                    bestk = unused[j];
                                    maxb = bijk;
                                }
                            }
                        }
                    }
                    A = 0;
                    if (maxb != 0)
                    {
                        a.Insert(bestj, bestk);
                        unused.RemoveAt(mina);
                        CapacityRemaining = CapacityRemaining - allCustomers[bestk
- 2].Getdemcustomer();
                        A = -1;

                    }
                }
                a.RemoveAt(0);
                a.RemoveAt(a.Count - 1);
                b.Add(a);
            }

            chromonew chromosolv = new chromonew(dim);
            List<List<Int32>> c = new List<List<Int32>>();
            for (int counting = 0; counting < b.Count;counting++ )
            {
                List<Int32> abc = new List<Int32>();
                for (int countter = 0; countter < b[counting].Count;countter++ )
                {
                    abc.Add(allCustomers[b[counting].ElementAt(countter)-
1].Getidcustomer());

                }
                    c.Add(abc);
            }

            chromosolv = aux.Encoding(c, dim );
            return chromosolv;
        }
```

Sweep Method

```
        public chromonew SweepMethod(customer[] allCustomers, inputdata
data8,double Capacity)
        {
            dim = allCustomers.Length;
            chromonew chromoSolv = new chromonew(dim - 1);
            auxiliary aux = new auxiliary();
            inputdata data3 = new inputdata();
            double[] Angles = new double[dim - 1];
            int[] CustomerId = new int[dim - 1];

            for (i = 1; (i < dim); i++)
            {
                Angles[i - 1] = aux.PolarAngle(allCustomers[i]);
                if (Angles[i - 1] < 0)
                {
                    Angles[i - 1] = 360 + Angles[i - 1]; ;
                }
                CustomerId[i - 1] = i + 1;
            }
            Array.Sort(Angles, CustomerId);
            List<List<Int32>> Routes = new List<List<Int32>>();
            VehicleCapacity = Capacity;
            CapacityRemaining = VehicleCapacity;
            List<Int32> a = new List<Int32>();

            for (i = 0; i < dim - 1; i++)
            {
                if (allCustomers[CustomerId[i] - 1].Getdemcustomer() <
CapacityRemaining)
                {
                    a.Add(CustomerId[i]);
                    CapacityRemaining = CapacityRemaining -
allCustomers[CustomerId[i] - 1].Getdemcustomer();
                }
                else
                {
                    Routes.Add(a);
                    CapacityRemaining = VehicleCapacity;
                    a = new List<Int32>();
                    a.Add(CustomerId[i]);
                    CapacityRemaining = CapacityRemaining -
allCustomers[CustomerId[i] - 1].Getdemcustomer();
                }
            }
            if (a.Count == 1)
            {
                for (q = 0; q < Routes.Count; q++)
                {
                    if (aux.TotDemand(Routes[q], allCustomers) +
allCustomers[CustomerId[dim - 2] - 1].Getdemcustomer() <= VehicleCapacity)
                    {
                        Routes[q].Add(CustomerId[dim - 2]);
                        VehicleCapacity = -100;
                    }
                }
            }
            if (VehicleCapacity != -100)
            {
                Routes.Add(a);
            }
            chromoSolv = aux.Encoding(Routes, dim - 1);
            return chromoSolv;
```

```
        }
```

15 Encoding

```
public chromonew Encoding(List<List<Int32>> a, int k)
        {
            inputdata dataux = new inputdata();
            chromonew chromo3 = new chromonew(k);

            z = 0;
            for (i = 0; i < a.Count; i++)
            {
                for (j = 0; j < a[i].Count; j++)
                {

                    chromo3.gonidia[j + z] = a[i].ElementAt(j);

                }
                z = z + a[i].Count;
            }

            return chromo3;
        }
```

16 Decoding

```
public List<List<Int32>> TwoOpting(chromonew chromo1,customer[] AllCustomers,double
VehicleCapacity,customer customer0)
        {
            List<List<Int32>> Route = new List<List<Int32>>();
            List<Int32> a = new List<Int32>();
            List<Int32> b = new List<Int32>();
            customer[] allcustomers=new customer[AllCustomers.Length+1];
            int dim = allcustomers.Length-1;

            for (int use = 1; use < dim + 1; use++)
            {
                allcustomers[use] = AllCustomers[use - 1];
            }
            allcustomers[0] = customer0;
            CapacityRem = VehicleCapacity;

            int[] Pinakas=new int[allcustomers.Length];

            for (int use = 0; use < allcustomers.Length;use++ )
            {
                Pinakas[use] = allcustomers[use].Getidcustomer();
            }


                for (i = 0; i < chromo1.gonidia.Length; i++)
                {
                    index1 = Array.IndexOf(Pinakas, chromo1.gonidia[i]);
                    if (CapacityRem > allcustomers[index1].Getdemcustomer())
                    {
                        a.Add(chromo1.gonidia[i]);
                        CapacityRem = CapacityRem -
allcustomers[index1].Getdemcustomer();
                    }
                    else
                    {
                        a.Insert(0, 1);
                        a.Add(1);
                        Route.Add(a);
                        CapacityRem = VehicleCapacity;
                        a = new List<Int32>();
```

```
                        b = new List<Int32>();
                        a.Add(chromo1.gonidia[i]);
                        CapacityRem = CapacityRem -
allcustomers[index1].Getdemcustomer();

                    }

                }
            a.Insert(0, 1);
            a.Add(1);
            Route.Add(a);


            List<List<Int32>> BestRoute=new List<List<Int32>>();
            for (z = 0; z < Route.Count; z++)
            {
                a = new List<Int32>();
                b = new List<Int32>();
                for (p = 0; p < Route[z].Count; p++)
                {
                    //a.Add(Route[z].ElementAt(p));
                }

                BestCost = CostFinder(a, allcustomers) +
DistanceCustomers(allcustomers[a[0] - 1], allcustomers[0]) +
DistanceCustomers(allcustomers[a[a.Count - 1] - 1], allcustomers[0]);
                for (i=0;i<a.Count;i++)
                {
                    b.Add(a[i]);
                }

                for (j = 0; j < a.Count-1;j++ )
                {
                    for (o=j+1;o<a.Count;o++)
                    {
                        for (k=0;k<a.Count;k++)
                        {
                            a[k] = b[k];
                        }
                        int temp = a[j];
                        a[j] = a[o];
                        a[o] = temp;

                        NewCost = CostFinder(a, allcustomers) +
DistanceCustomers(allcustomers[a[0] - 1], allcustomers[0]) +
DistanceCustomers(allcustomers[a[a.Count - 1] - 1], allcustomers[0]);

                        if (NewCost<BestCost)
                        {
                            BestCost = NewCost;
                            for (q=0;q<a.Count;q++)
                            {
                                b[q] = a[q];
                            }
                        }
                    }
                }
                b.Insert(0, 1);
                b.Add(1);
                BestRoute.Add(b);
            }
            return Best Route;
        }
```

PMX crossover

```
  public chromonew PMXcrossover(chromonew chromo1,chromonew chromo2,int spot1,int
spot2)
        {
            int dim = chromo1.gonidia.Length;
            chromonew chromo3 = new chromonew(dim);

            List<Int32> a = new List<Int32>();
            List<Int32> Zero = new List<Int32>();
            for (i = 0; i < dim; i++)
            {
                a.Add(i + 2);
            }

            int[] temp = new int[spot2 - spot1 + 1];
            int[] temp2 = new int[spot2 - spot1 + 1];

            for (j = spot1; j < spot2 + 1;j++ )
            {
                temp[j - spot1] = chromo2.gonidia[j];
                temp2[j - spot1] = chromo1.gonidia[j];
            }

            for (q = 0; q < spot1;q++ )
            {
                if (temp.Contains(chromo1.gonidia[q]) == false)
                {
                    chromo3.gonidia[q] = chromo1.gonidia[q];
                }
                else
                {
                    chromo3.gonidia[q] = 0;
                    Zero.Add(chromo1.gonidia[q]);
                }
            }

            for (q = spot1; (q < spot2 + 1);q++ )
            {
                chromo3.gonidia[q] = temp[q - spot1];
            }
            for (q = spot2 + 1; q < dim;q++ )
            {
                if (temp.Contains(chromo1.gonidia[q]) == false)
                {
                    chromo3.gonidia[q] = chromo1.gonidia[q];
                }
                else
                {
                    chromo3.gonidia[q] = 0;
                    Zero.Add(chromo1.gonidia[q]);
                }
            }
            for (i = 0; i < temp.Length;i++ )
            {
                for (j=0;j<temp.Length;j++)
                {
                    if (temp[j] == temp2[i])
                    {
                        int tempor = temp[j];
                        temp[j] = temp[i];
                        temp[i] = tempor;
                    }
                }
```

```
            }
            int[] real= new int[Zero.Count];

            for (i = 0; i < Zero.Count;i++ )
            {
                real[i] = temp2[Array.IndexOf(temp, Zero[i])];
            }
            o=0;


            maxValue = 0;

            for (q=0;q<dim;q++)
                {
                    if (chromo3.gonidia[q]==0)
                    {
                        maxValue=maxValue+1;
                    }
                }
            for (i = 0; i < dim;i++ )
            {
                if (chromo3.gonidia[i]==0)
                {
                    chromo3.gonidia[i] = real[o];
                    o=o+1;
                }
            }

                return chromo3;
        }
```

Mutation

```
        public chromonew Mutating(chromonew chromo1,int spot1,int spot2)
        {

            int temp = chromo1.gonidia[spot1];
            chromo1.gonidia[spot1] = chromo1.gonidia[spot2];
            chromo1.gonidia[spot2] = temp;

            return chromo1;
        }
```

Inversion

```
public chromonew Inversing(customer[] allcustomers,chromonew chromo1,int spot1,int
spot2,inputdata data23)
        {
            int dim = Math.Abs(spot2 - spot1 + 1);
            int[] temp = new int[dim];
            double dist1;
            double dist2;
            dist1 = DistanceCustomers(allcustomers[chromo1.gonidia[spot1] - 1],
allcustomers[chromo1.gonidia[spot1 - 1] - 1],data23) +
DistanceCustomers(allcustomers[chromo1.gonidia[spot2] - 1],
allcustomers[chromo1.gonidia[spot2 + 1] - 1],data23);
            dist2 = DistanceCustomers(allcustomers[chromo1.gonidia[spot2] - 1],
allcustomers[chromo1.gonidia[spot1 - 1] - 1],data23) +
DistanceCustomers(allcustomers[chromo1.gonidia[spot1] - 1],
allcustomers[chromo1.gonidia[spot2 + 1] - 1],data23);

            if (dist2<dist1)
```

```
    {
        for (i=0;i<dim;i++)
        {
            temp[i] = chromo1.gonidia[spot2 - i];
        }
        for (j=spot1;j<spot2+1;j++)
        {
            chromo1.gonidia[j] = temp[j - spot1];
        }
    }
    return chromo1;
}
```