ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Σχεδιασμός και υλοποίηση υβριδικού μεταευρετικού αλγορίθμου βελτιστοποίησης προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου

Βάθης Μιχάλης

ΜΑΡΤΙΟΣ 2015

Επιβλέπων Καθηγητής : Πόνης Σταύρος

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Σταύρο Πόνη για την άριστη συνεργασία, τις χρήσιμες συμβουλές και τις κατευθύνσεις για μελέτη, τον κ. Κώστα Χατζόγλου για την πολύτιμη βοήθειά του σε καίρια στάδια στην πορεία εξέλιξης της εργασίας και την κ. Έλενα Ρόκου για την αμέριστη βοήθειά της, χωρίς την οποία δεν θα ήταν δυνατή η ολοκλήρωση της παρούσας εργασίας. Τέλος, θέλω να ευχαριστήσω την οικογένειά μου και όλους όσους με στήριξαν τόσο τους τελευταίους μήνες, κατά την εκπόνηση της διπλωματικής μου, όσο και τα υπόλοιπα τα χρόνια των σπουδών μου στη σχολή Μηχανολόγων Μηχανικών.

Περιεχόμενα

ΕΥΧΑΡΙΣΤΙΕΣ ..... **- 2 -**

ΚΑΤΆΛΟΓΟΣ ΠΙΝΆΚΩΝ ..... **- 5 -**

ΚΑΤΆΛΟΓΟΣ ΕΞΙΣΩΣΕΩΝ ..... **- 6 -**

ΚΑΤΆΛΟΓΟΣ ΣΧΗΜΆΤΩΝ ..... **- 8 -**

ΚΑΤΆΛΟΓΟΣ ΑΛΓΟΡΙΘΜΩΝ ..... **- 10 -**

ΈΠΟΨΗ ..... **- 11 -**

SYNOPSIS ..... **- 12 -**

1 ΕΙΣΑΓΩΓΗ ..... **- 13 -**

2 ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ ..... **- 16 -**

2.1 Η ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΩΝ ΠΡΟΒΛΗΜΑΤΩΝ ΜΕΤΑΦΟΡΩΝ ..... - 16 -

2.2 ΤΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ (VRP) ΚΑΙ ΟΙ ΠΑΡΑΛΛΑΓΕΣ ΤΟΥ . - 18 -

2.2.1 Πελάτες και Υπηρεσίες ..... *- 18 -*

2.2.2 Στόλος Οχημάτων και Κέντρο Διανομής ..... *- 20 -*

2.2.3 Χρονικός Ορίζοντας Δρομολόγησης ..... *- 26 -*

2.2.4 Στοχαστικά και Δυναμικά Προβλήματα ..... *- 27 -*

2.2.5 Εμπλουτισμένο VRP (Rich VRP) ..... *- 29 -*

2.3 ΤΟ VRP ΜΕ ΠΑΡΑΘΥΡΑ ΧΡΟΝΟΥ (TW) ..... - 29 -

2.4 ΜΕΘΟΔΟΙ ΕΠΙΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΩΝ ΔΡΟΜΟΛΟΓΗΣΗΣ ΜΕ ΠΑΡΑΘΥΡΑ ΧΡΟΝΟΥ - 31 -

2.4.1 Αναλυτικές Μέθοδοι ..... *- 31 -*

2.4.2 Ευρετικές Μέθοδοι ..... *- 32 -*

2.4.3 Μεταευρετικές Μέθοδοι ..... *- 41 -*

2.5 Η ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΑΠΟΙΚΙΑΣ ΜΥΡΜΗΓΚΙΩΝ (ACO) ..... - 44 -

2.6 Η ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΣΜΗΝΟΥΣ ΣΩΜΑΤΙΔΙΩΝ (PSO) ..... - 47 -

2.6.1 Προσαρμογή του PSO σε προβλήματα ακέραιου προγραμματισμού ..... *- 50 -*

3 ΟΡΙΣΜΟΣ ΤΟΥ ΥΠΟ ΜΕΛΕΤΗ ΠΡΟΒΛΗΜΑΤΟΣ ..... **- 56 -**

4 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ..... **- 57 -**

4.1 ΟΡΙΣΜΟΙ – ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΤΙ ΣΗΜΑΙΝΟΥΝ ΚΑΙ ΠΑΡΑΔΟΧΕΣ ..... - 57 -

4.1.1 Μεταβλητές απόφασης ..... *- 58 -*

4.1.2 Παράμετροι προβλήματος ..... *- 59 -*

4.2 ΑΝΤΙΚΕΙΜΕΝΙΚΗ ΣΥΝΑΡΤΗΣΗ ..... - 62 -

4.3 ΠΕΡΙΟΡΙΣΜΟΙ ..... - 64 -

4.4 ΜΑΘΗΜΑΤΙΚΗ ΜΟΝΤΕΛΟΠΟΊΗΣΗ ..... - 76 -
5 ΠΡΟΤΕΙΝΌΜΕΝΗ ΜΈΘΟΔΟΣ ΕΠΊΛΥΣΗΣ ..... - 78 -
5.1 ΠΕΡΙΓΡΑΦΉ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ..... - 78 -
5.2 ΠΡΟΤΕΙΝΟΜΕΝΟΣ ΑΛΓΟΡΙΘΜΟΣ ..... - 79 -
5.2.1 Διάγραμμα ροής του προτεινόμενου αλγορίθμου ..... - 79 -
5.2.2 Δομικά στοιχεία του προτεινόμενου αλγορίθμου ..... - 80 -
5.3 ΥΛΟΠΟΙΗΣΗ ΠΡΟΤΕΙΝΟΜΕΝΟΥ ΑΛΓΟΡΙΘΜΟΥ ..... - 82 -
5.3.1 Προγραμματιστικό περιβάλλον ..... - 82 -
5.3.2 Βασικοί Παράμετροι προτεινόμενου αλγορίθμου ..... - 82 -
5.3.3 Βασικές μέθοδοι που υλοποιήθηκαν ..... - 84 -
5.4 ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ ΜΗΧΑΝΗΣ ..... - 102 -
6 ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ..... - 106 -
6.1 ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΤΟΥ ΠΡΟΤΕΙΝΟΜΕΝΟΥ ΑΛΓΟΡΙΘΜΟΥ ..... - 106 -
6.2 ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΦΟΡΑΣ ..... - 107 -
6.3 ΠΕΙΡΑΜΑΤΙΚΗ ΕΠΑΛΗΘΕΥΣΗ ΤΟΥ ΠΡΟΤΕΙΝΟΜΕΝΟΥ ΑΛΓΟΡΙΘΜΟΥ ΚΑΙ ΣΤΑΤΙΣΤΙΚΗ ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ..... - 108 -
7 ΣΥΜΠΕΡΑΣΜΑΤΑ ..... - 111 -
8 ΒΙΒΛΙΟΓΡΑΦΙΑ ..... - 113 -

Κατάλογος Πινάκων

Πίνακας 4.1 Μεταβλητές Απόφασης ..... - 58 -

Πίνακας 4.2 Παράμετροι Προβλήματος..... - 61 -

Πίνακας 5.1 Παράμετροι Αλγορίθμου Βελτιστοποίησης Αποικίας Μυρμηγκιών ..... - 83 -

Πίνακας 5.2 Παράμετροι Αλγορίθμου Βελτιστοποίησης σμήνους Σωματιδίων ..... - 84 -

Πίνακας 5.3 Είσοδοι του αλγορίθμου PSO ..... - 95 -

Πίνακας 6.1 Οι προτεινόμενες από τη βιβλιογραφία παράμετροι του ACO..... - 106 -

Πίνακας 6.2 Αποτελέσματα Βελτιστοποίησης των προβλημάτων αναφοράς του Solomon .... - 109 -

Κατάλογος Εξισώσεων

Εξίσωση (2.1) ..... - 35 -
Εξίσωση (2.2) ..... - 35 -
Εξίσωση (2.3) ..... - 48 -
Εξίσωση (2.4) ..... - 48 -
Εξίσωση (2.5) ..... - 50 -
Εξίσωση (2.6) ..... - 50 -
Εξίσωση (2.7) ..... - 50 -
Εξίσωση (2.8) ..... - 51 -
Εξίσωση (4.1) ..... - 62 -
Εξίσωση (4.2) ..... - 63 -
Εξίσωση (4.3) ..... - 65 -
Εξίσωση (4.4) ..... - 65 -
Εξίσωση (4.5) ..... - 67 -
Εξίσωση (4.6) ..... - 68 -
Εξίσωση (4.7) ..... - 69 -
Εξίσωση (4.8) ..... - 69 -
Εξίσωση (4.9) ..... - 73 -
Εξίσωση (4.10) ..... - 75 -
Εξίσωση (4.11) ..... - 76 -
Εξίσωση (4.12) ..... - 76 -
Εξίσωση (4.13) ..... - 76 -
Εξίσωση (4.14) ..... - 76 -
Εξίσωση (4.15) ..... - 76 -
Εξίσωση (4.16) ..... - 76 -
Εξίσωση (4.17) ..... - 76 -
Εξίσωση (4.18) ..... - 76 -
Εξίσωση (4.19) ..... - 76 -
Εξίσωση (4.20) ..... - 76 -
Εξίσωση (4.21) ..... - 76 -
Εξίσωση (5.1) ..... - 89 -

Εξίσωση (5.2) ..... - 99 -
Εξίσωση (5.3) ..... - 99 -
Εξίσωση (5.4) ..... - 100 -

Κατάλογος Σχημάτων

Σχήμα 1.1 : Μέθοδος Έρευνας ..... - 15 -
Σχήμα 2.1 : Παράδειγμα OVRP ..... - 22 -
Σχήμα 2.2 : Παράδειγμα MTVRP ..... - 24 -
Σχήμα 2.3 : Παράδειγμα MDVRP ..... - 25 -
Σχήμα 2.4 : Παράδειγμα PVRP (Yu & Yang, 2011) ..... - 26 -
Σχήμα 2.5 : Παράδειγμα DVRP (Victor Pillac et al., 2013) ..... - 29 -
Σχήμα 2.6 Παράδειγμα VRP με Παράθυρα Χρόνου ..... - 30 -
Σχήμα 2.7 : Παράδειγμα Αλγορίθμου Nearest Neighbor ..... - 34 -
Σχήμα 2.8 : Παράδειγμα Αλγορίθμου "Savings" ..... - 37 -
Σχήμα 2.9 Μέθοδος 2-opt: Αρχική διαδρομή οχήματος ..... - 39 -
Σχήμα 2.10 Μέθοδος 2-opt: Η γειτονιά της αρχικής διαδρομής ..... - 40 -
Σχήμα 2.11 : Η μέθοδος τοπικής αναζήτησης 2-opt ..... - 41 -
Σχήμα 2.12 : Διάγραμμα ροής ACO ..... - 47 -
Σχήμα 2.13 : Διάγραμμα Ροής PSO ..... - 49 -
Σχήμα 2.14: Αναπαράσταση Λύσης VRP του Chen (2006) ..... - 50 -
Σχήμα 2.15 Κωδικοποίηση της λύσης για τον PSO Gong et al. (2012) ..... - 53 -
Σχήμα 2.16 Μέθοδος Path Relinking ..... - 55 -
Σχήμα 4.1 : Παράδειγμα Μεταβλητών Απόφασης $\boldsymbol{xijk}$ ..... - 59 -
Σχήμα 4.2 : Παράδειγμα Παραμέτρων Προβλήματος $\boldsymbol{cij}$ ..... - 62 -
Σχήμα 4.3 Παράδειγμα υπολογισμού της πρώτης αντικειμενικής συνάρτησης ..... - 63 -
Σχήμα 4.4 Πρώτο βήμα υπολογισμών παραδείγματος της δεύτερης αντικειμενικής συνάρτησης ..... - 64 -
Σχήμα 4.5 Δεύτερο βήμα υπολογισμών παραδείγματος της δεύτερης αντικειμενικής συνάρτησης ..... - 64 -
Σχήμα 4.6 Τρίτο βήμα υπολογισμών παραδείγματος της δεύτερης αντικειμενικής συνάρτησης ..... - 64 -
Σχήμα 4.7 Υπολογισμοί παραδείγματος του περιορισμού (4.3) ..... - 65 -
Σχήμα 4.8 Υπολογισμοί παραδείγματος του περιορισμού (4.4) βήμα α' ..... - 66 -
Σχήμα 4.9 Υπολογισμοί παραδείγματος του περιορισμού (4.4) βήμα β' ..... - 66 -
Σχήμα 4.10 Υπολογισμοί παραδείγματος του περιορισμού (4.4) βήμα γ' ..... - 67 -

Σχήμα 4.11 Υπολογισμοί παραδείγματος του περιορισμού (4.5) ..... - 67 -
Σχήμα 4.12 Υπολογισμοί παραδείγματος του περιορισμού (4.6) ..... - 68 -
Σχήμα 4.13 Υπολογισμοί παραδείγματος του περιορισμού (4.7) ..... - 69 -
Σχήμα 4.14 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα α' ..... - 70 -
Σχήμα 4.15 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα β' ..... - 70 -
Σχήμα 4.16 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα γ' ..... - 71 -
Σχήμα 4.17 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα δ' ..... - 71 -
Σχήμα 4.18 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα ε' ..... - 72 -
Σχήμα 4.19 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα στ' ..... - 72 -
Σχήμα 4.20 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα ζ' ..... - 73 -
Σχήμα 4.21 Υπολογισμοί παραδείγματος του περιορισμού (4.9) βήμα α' ..... - 73 -
Σχήμα 4.22 Υπολογισμοί παραδείγματος του περιορισμού (4.9) βήμα β' ..... - 74 -
Σχήμα 4.23 Υπολογισμοί παραδείγματος του περιορισμού (4.9) βήμα γ' ..... - 74 -
Σχήμα 5.1 : Διάγραμμα Ροής του προτεινόμενου αλγορίθμου ..... - 79 -
Σχήμα 5.2 : Σχηματική απεικόνιση των κλάσεων που χρησιμοποιήθηκαν ..... - 81 -
Σχήμα 5.3 : Έξοδος Συστήματος σε μορφή Αρχείου MS Excel ..... - 103 -
Σχήμα 5.4 : Γραφική Έξοδος Συστήματος ..... - 104 -
Σχήμα 5.5 : Παράδειγμα Εξόδου Συστήματος Πραγματικών Δεδομένων ..... - 105 -

Κατάλογος Αλγορίθμων

Αλγόριθμος 2.1 : Αλγόριθμος Nearest Neighbor ..... - 33 -
Αλγόριθμος 2.2 : Αλγόριθμος Nearest Neighbor για το VRPTW (1987) ..... - 36 -
Αλγόριθμος 2.3 : Αλγόριθμος "Savings"..... - 37 -
Αλγόριθμος 2.4 : Αλγόριθμος "Savings" για το VRPTW (Marius M Solomon, 1987)..... - 38 -
Αλγόριθμος 2.5 : Ψευδοκώδικας γενετικού αλγορίθμου ..... - 43 -
Αλγόριθμος 2.6 : Αναζήτηση Taboo ..... - 44 -
Αλγόριθμος 2.7 : Ψευδοκώδικας Βελτιστοποίησης Αποικίας Μυρμηγκιών ..... - 46 -
Αλγόριθμος 2.8 : Αλγόριθμος PSO ..... - 49 -
Αλγόριθμος 2.9 : Αλγόριθμος PSO Belmecheri (2012)..... - 51 -
Αλγόριθμος 2.10 : Αποκωδικοποίηση Λύσης PSO Gong (2012) ..... - 54 -

Έποψη

Στην παρούσα εργασία παρουσιάζεται ένας υβριδικός μεταευρετικός αλγόριθμος για την επίλυση του προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (Vehicle Routing Problem with Time Windows – VRPTW), βασισμένο στον συνδυασμό του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization – ACO) και του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization – PSO).

Το πρόβλημα αποτελείται από ένα σύνολο οχημάτων, μία κεντρική αποθήκη και ένα σύνολο πελατών προς εξυπηρέτηση. Τα οχήματα έχουν περιορισμένη χωρητικότητα και ο κάθε πελάτης χαρακτηρίζεται από προκαθορισμένη ζήτηση, συντεταγμένες αλλά και ένα παράθυρο χρόνου στο οποίο επιτρέπεται η εξυπηρέτησή του από κάποιο όχημα. Ο κάθε πελάτης πρέπει να εξυπηρετηθεί από ένα και μόνο όχημα. Το κάθε όχημα ξεκινά το δρομολόγιό του από την κεντρική αποθήκη και αφού εξυπηρετήσει τους πελάτες που του αντιστοιχούν ανάλογα με το πλάνο δρομολόγησης, πρέπει να επιστρέψει στην κεντρική αποθήκη μέχρι κάποιο προκαθορισμένο χρονικό όριο.

Η κατάστρωση του βέλτιστου δρομολογίου για την εξυπηρέτηση των πελατών γίνεται με κεντρικό άξονα πρώτον την ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και δεύτερον την ελαχιστοποίηση της απόστασης που διανύουν αθροιστικά τα οχήματα κατά την εκτέλεση του δρομολογίου τους, συνεπώς, λύσεις του προβλήματος με λιγότερα οχήματα είναι βέλτιστες ανεξαρτήτου μήκους διαδρομής που αυτά διανύουν.

Η μέθοδος που προτείνεται για την επίλυση του προβλήματος αποτελείται από έναν αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών (ACO) ο οποίος σε κάθε του επανάληψη, μετά την κατασκευή της λύσης που αναπαριστά το κάθε μυρμήγκι, τις βελτιστοποιεί χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων (PSO). Συγκεκριμένα, σε κάθε επανάληψη του αλγορίθμου ACO κατασκευάζεται μία λύση για το κάθε μυρμήγκι, βρίσκεται η τοπικά βέλτιστη λύση χρησιμοποιώντας μία μέθοδο τοπικής αναζήτησης και στη συνέχεια αντιστοιχίζεται το κάθε μυρμήγκι σε ένα σωματίδιο ώστε να βελτιστοποιηθεί ο πληθυσμός τους με τη χρήση του αλγορίθμου PSO. Μετά το πέρας του αλγορίθμου PSO, εφαρμόζεται μετατροπή των βέλτιστων σωματιδίων σε μυρμήγκια, γίνεται ανανέωση των φερομονών και συνεχίζονται οι επαναλήψεις του ACO.

Η αποδοτικότητα και η αποτελεσματικότητα του προτεινόμενου αλγορίθμου εξετάστηκε ενδελεχώς χρησιμοποιώντας πρότυπα προβλήματα από τη βιβλιογραφία. Τα αποτελέσματα είναι ενθαρρυντικά για την περεταίρω έρευνα προς την κατεύθυνση του συνδυασμού των δύο αλγορίθμων και της παραμετροποίησης του αλγορίθμου για τα εκάστοτε προβλήματα.

Synopsis

In the current thesis a hybrid meta-heuristic algorithm is presented based on particle swarm and ant colony optimization, for solving a variant of one of the most popular supply chain management problems, the vehicle routing problem with time windows.

The problem consists of a vehicle fleet, a central depot and a set of customers to be served. Each vehicle has the same capacity and each customer is characterized by predetermined demand, coordinates and a time window in which he is to be served. Each customer must be served by one vehicle and no customer's demand is larger than the vehicle's capacity. Each vehicle starts its route from the main depot and must return to the main depot in a predefined time limit.

The construction of the best solution for the service of the customers is conducted, firstly, with the objective of minimizing the vehicles used and, secondly, with the objective of minimizing the total distance traveled by the vehicle fleet. It is implied that the cost of using an additional vehicle is far superior to the variable cost of transportation, so that solutions that use fewer vehicles are preferred regardless of the distance travelled by the vehicles.

The proposed algorithm for the solution of the vehicle routing problem consists of an ant colony optimization algorithm in which in each iteration after the construction of the solutions, each ant is encoded into a vector that represents the starting position of a particle and the whole swarm is processed by a particle swarm optimization algorithm with a path relinking strategy. Afterwards, the pheromone evaporation and update are executed and the iterations of the ant colony optimization algorithm continue.

The efficiency and effectiveness of the proposed algorithm were thoroughly tested using test cases taken from the literature with very promising preliminary results. Further research has to be done on fine tuning the algorithm for special cases.

1 Εισαγωγή

Ένας από τους σημαντικότερους παράγοντες στη βελτιστοποίηση μιας εφοδιαστικής αλυσίδας είναι αυτός του δικτύου μεταφορών. Σύμφωνα με τη βιβλιογραφία στη πλειονότητα των προϊόντων το κόστος μεταφοράς και διανομής μπορεί να ανέλθει σε επίπεδα από 15% έως 50% του συνολικού κόστους για τον καταναλωτή (Labadie & Prins, 2012). Τα παραπάνω αποτελέσματα φανερώνουν ότι η ελαχιστοποίηση του κόστους μεταφοράς και διανομής μπορεί να προσφέρει ανταγωνιστικό πλεονέκτημα στις επιχειρήσεις που χρησιμοποιούν τέτοιους αλγορίθμους βελτιστοποίησης και να μειώσει τα κόστη που καλούνται να πληρώσουν οι καταναλωτές. Οι παραπάνω ανάγκες αύξησαν τη ζήτηση για αλγορίθμους βελτιστοποίησης προβλημάτων δρομολόγησης και οδήγησαν σε ερευνητική έκρηξη στη επιχειρησιακή έρευνα τα τελευταία 50 χρόνια (Christofides, 1976; Cordeau, Desaulniers, Desrosiers, Solomon, & Soumis, 2001; Tavares, Machado, Pereira, & Costa, 2003; Yu, Yang, & Yao, 2011). Η παρούσα διπλωματική ασχολείται με το πρόβλημα της δρομολόγησης οχημάτων (Vehicle Routing Problem – VRP). Ειδικότερα, επιλέχθηκε η παραλλαγή του VRP με παράθυρα χρόνου (Vehicle Routing Problem with Time Windows - VRPTW) λόγω της μεγάλης συχνότητας εμφάνισης του συγκεκριμένου τύπου σε πραγματικά προβλήματα. Πράγματι, στις περισσότερες περιπτώσεις η ύπαρξη χρονικών παραθύρων για την παράδοση ή και την παραλαβή προϊόντων αποτελεί κοινή πρακτική. Συχνά προβλέπονται οικονομικές ρήτρες και πιθανές καθυστερήσεις εξυπηρέτησης σε περίπτωση μη τήρησης του χρονοδιαγράμματος παράδοσης - παραλαβής. Η εξεταζόμενη προσέγγιση του προβλήματος περιλαμβάνει την ανάπτυξη ενός πληροφοριακού εργαλείου το οποίο δέχεται σαν δεδομένα τα διαθέσιμα οχήματα, το χρονοδιάγραμμα παράδοσης, της ζήτησης και τις αποστάσεις μεταξύ των σημείων παράδοσης και εξάγει παραπλήσιες στη βέλτιστη λύσεις για τη δρομολόγηση οχημάτων με στόχο την ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και του κόστους διανομής. Όσο αφορά τον τρόπο επίλυσης του υφιστάμενου προβλήματος βελτιστοποίησης έχει επιλεχθεί η δημιουργία υβριδικού αλγορίθμου που συνδυάζει τους εξελικτικούς μετα-ευρετικούς αλγορίθμους βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization - ACO) και σμήνους σωματιδίων (Particle Swarm Optimization - PSO), με τελικό στόχο την επίτευξη βελτιωμένης αποτελεσματικότητας και αποδοτικότητας. Ο αλγόριθμος που αναπτύχθηκε συγκρίνεται ως προς την αποτελεσματικότητά του με τα υπάρχοντα προβλήματα αναφοράς (πχ. Solomon Benchmark Problems), όπως αυτά προκύπτουν από τη βιβλιογραφία.

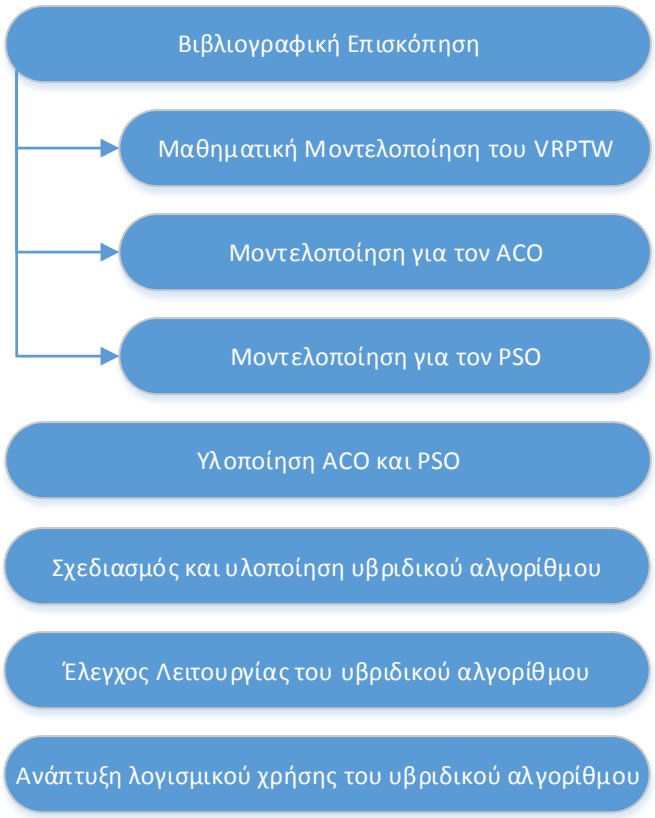
Στο υπό εξέταση VRPTW έχουμε ένα στόλο από πανομοιότυπα (ίδιας χωρητικότητας) οχήματα, με τα οποία θέλουμε να καλυφθεί η ζήτηση ενός συνόλου πελατών. Εξετάζουμε την περίπτωση που μεταφέρεται ένα είδος προϊόντων και ο κάθε πελάτης εξυπηρετείται από κάποιο όχημα μία φορά και το όχημα αυτό καλύπτει ολόκληρη τη ζήτηση του πελάτη. Επίσης, ο κάθε πελάτης ορίζει μια χρονική στιγμή νωρίτερης δυνατής εξυπηρέτησης (earliest arrival) και μία χρονική στιγμή αργότερης δυνατής εξυπηρέτησης (latest arrival). Το όχημα που τελικά θα εξυπηρετήσει τον κάθε πελάτη μπορεί να φτάσει σε αυτόν νωρίτερα από το αριστερό όριο του χρονικού παραθύρου (earliest arrival) και να περιμένει, όχι όμως αργότερα από το δεξί όριο (latest arrival). Επιπρόσθετα, γίνεται ανάπτυξη διεπαφής χρήστη με χρήση της οποίας θα είναι δυνατό να δοθούν σαν είσοδος τα χαρακτηριστικά της

αποθήκης, των οχημάτων και των πελατών και να επιστρέφεται ως αποτέλεσμα η βέλτιστη διαδρομή που πρέπει να ακολουθηθεί από τα οχήματα. Τα αποτελέσματα θα οπτικοποιούνται είτε πάνω σε χάρτη (αν τα στοιχεία αντιστοιχούν σε πραγματικά δεδομένα) είτε σε διάγραμμα αξόνων (πχ. στην περίπτωση εισαγωγής κάποιου προβλήματος αναφοράς).

Η μέθοδος που ακολουθήθηκε για την εκπόνηση της διπλωματικής εργασίας παρουσιάζεται στο Σχήμα 1.1. Αρχικά έγινε βιβλιογραφική επισκόπηση επί του προβλήματος δρομολόγησης οχημάτων και των πιο συχνά χρησιμοποιούμενων παραλλαγών του και οριοθετήθηκε ο χώρος έρευνας και εν γένει ορίστηκε το πλαίσιο της παρούσας εργασίας. Ο κεντρικός θεματικός άξονας της οποίας είναι η δρομολόγηση συγκεκριμένου πλήθους και χωρητικότητας, διαθέσιμων οχημάτων για την εξυπηρέτηση των εκάστοτε πελατών, ο καθένας από τους οποίους ζητάει προκαθορισμένο πλήθος προϊόντων. Επιπλέον στην υπό μελέτη εκδοχή του προβλήματος, λαμβάνεται υπόψη και η ύπαρξη συγκεκριμένων παράθυρων χρόνου για την παράδοση στον κάθε πελάτη.

Συγκεκριμένα, μελετήθηκαν οι προτεινόμενες λύσεις στη βιβλιογραφία τόσο με αναλυτικές μαθηματικές διαδικασίες όσο και κάνοντας χρήση, ευρετικών και μεταευρετικών αλγορίθμων. Στη συνέχεια επιλέχτηκαν δύο μεταευρετικοί αλγόριθμοι, ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών και ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων, οι οποίοι είχαν τη μεγαλύτερη αποτελεσματικότητα στις έως τώρα εφαρμογές τους στο συγκεκριμένο και σε παρεμφερή προβλήματα και χρησιμοποιήθηκαν συνδυαστικά για την επίλυση του υπό μελέτη προβλήματος. Στη συνέχεια, μελετήθηκαν και συγκρίθηκαν οι λύσεις που έχουν προταθεί στη βιβλιογραφία για τη μοντελοποίηση του προβλήματος, τόσο στην περίπτωση που γίνεται χρήση του αλγόριθμου βελτιστοποίησης σμήνους σωματιδίων όσο και για τον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών.

Με βάση τα αποτελέσματα των υπαρχόντων αλγόριθμων και των σημείων που μειονεκτεί και πλεονεκτεί ο καθένας εξ αυτών στο συγκεκριμένο πρόβλημα, σχεδιάστηκε ένας νέος υβριδικός αλγόριθμος για την επίτευξη καλύτερων λύσεων, υπό την έννοια της εύρεσης δρομολογίων που να κάνουν χρήση κατά το δυνατόν λιγότερων οχημάτων και μικρότερου συνολικού μήκους διαδρομών. Μετά τα παραπάνω βήματα, μελετήθηκε το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε και υλοποιήθηκε ο αλγόριθμος. Τέλος, ελέγχθηκε ο αλγόριθμος για την εγκυρότητα και την αποδοτικότητά του στα προβλήματα αναφοράς, σύμφωνα με τα δεδομένα της βιβλιογραφίας.



Σχήμα 1.1 : Μέθοδος Έρευνας

2 Βιβλιογραφική Επισκόπηση

Οι μεταφορές μπορούν να οριστούν ως ο επιστημονικός κλάδος που αφορά την τεχνολογία και τις μεθόδους μεταφορών. Είτε οδικές, είτε θαλάσσιες, είτε εναέριες, όλες οι μεταφορές ακολουθούν θεμελιώδεις κανόνες και έχουν συγκεκριμένες ιδιότητες. Η επιστήμη των μεταφορών ορίζει αυτές τις ιδιότητες και δείχνει πως οι γνώσεις που αποκτώνται για κάποιον τρόπο μεταφοράς μπορούν να χρησιμοποιηθούν ως βάση για να εξηγήσουν τη συμπεριφορά των υπόλοιπων τρόπων μεταφοράς (Christofides, 1976; Labadie & Prins, 2012).

Θεμελιωδώς, η επιστήμη των μεταφορών αποδέχεται πως όλοι οι τρόποι διακίνησης έχουν συγκεκριμένα συστατικά στοιχεία: οχήματα, δρόμους και σταθμούς που όλα τα παραπάνω ακολουθούν κάποια πολιτική λειτουργίας. Στη συνέχεια, γίνεται ανάλυση των διαφορετικών μορφών που λαμβάνει το συγκεκριμένο πρόβλημα με βάση τις εκάστοτε απαιτήσεις και τα υπάρχοντα δεδομένα, ξεκινώντας από τη γενική μοντελοποίηση προβλημάτων μεταφοράς και συνεχίζοντας με τις πιο συχνά συναντώμενες εκδοχές του υπό μελέτη προβλήματος της δρομολόγησης οχημάτων.

2.1 Η μοντελοποίηση των προβλημάτων μεταφορών

Οι μεταφορές χωρίζονται σε μεταφορές πλήρους φορτίου (Truckload Transportation) και σε μεταφορές μη πλήρους φορτίου (Less than Truckload Transportation).

Στις μεταφορές πλήρους φορτίου, η ζήτηση των πελατών είναι αρκετά μεγάλη ώστε να οριστεί ένα όχημα αποκλειστικά για την εξυπηρέτηση ενός πελάτη. Έτσι, το κάθε όχημα φορτώνεται σε κάποιο σημείο παραγωγής του αγαθού ή σε κάποια αποθήκη, μεταφέρει τα αγαθά αυτά σε κάποιον πελάτη, τα ξεφορτώνει εκεί και επιστρέφει πίσω στην αφετηρία του δρομολογίου του (Demir, Bektaş, & Laporte, 2014; Laporte, Toth, & Vigo, 2013).

Στις μεταφορές μη πλήρους φορτίου, η ζήτηση του κάθε πελάτη είναι μικρότερη από τη χωρητικότητα των οχημάτων και έτσι το κάθε όχημα εξυπηρετεί ένα πλήθος πελατών. Συνεπώς, πρέπει να ληφθούν δύο είδη αποφάσεων

1. Η επιλογή των πελατών που θα εξυπηρετηθούν από το κάθε όχημα
2. Η επιλογή της σειράς που θα εξυπηρετηθούν οι πελάτες από το κάθε όχημα

ενώ πρέπει να ικανοποιούνται και συγκεκριμένοι περιορισμοί όπως αναφορικά με τη χωρητικότητα των οχημάτων, την προσβασιμότητα και την απόσταση από τις αποθήκες φόρτωσης των σημείων παράδοσης στους πελάτες, το μέγιστο πλήθος ωρών συνεχούς οδήγησης κλπ. Τελικός στόχος είναι η ελαχιστοποίηση ενός ή περισσότερων αντικειμενικών στόχων, όπως ο αριθμός των οχημάτων που θα χρησιμοποιηθούν και η συνολική απόσταση που θα διανύσουν τα οχήματα.

Το κόστος των μεταφορών αντιπροσωπεύει, κατά μέσο όρο, το 35-70% του συνολικού κόστους της εφοδιαστικής αλυσίδας και περίπου το 25% της τιμής πώλησης του τελικού αγαθού. Έτσι, η βελτίωση της δρομολόγησης των οχημάτων μπορεί να φέρει σημαντική μείωση του κόστους και αποτελεί μια σημαντική κατηγορία της επιχειρησιακής έρευνας. Επίσης, η σημασία της δρομολόγησης έχει αυξηθεί από τις επιπτώσεις των μεταφορών στις

κλιματικές αλλαγές: το 2008 οι μεταφορές ήταν υπεύθυνες για το 30% της παραγωγής διοξειδίου του άνθρακα για τις ΗΠΑ και την ΕΕ (Gendreau, Laporte, Musaraganyi, & Taillard, 1999). Έτσι, δημιουργείται η ανάγκη για εκτενή μελέτη της δρομολόγησης οχημάτων και συνεπώς ανάγκη για την θεωρητική μοντελοποίηση του προβλήματος με έναν γενικό τρόπο.

Οπτική αναπαράσταση των προβλημάτων αυτής της κατηγορίας είθισται να γίνεται χρησιμοποιώντας γράφους, που αποτελούνται από κόμβους (nodes - πόλεις, διασταυρώσεις, σημεία πώλησης, φόρτωσης-εκφόρτωσης κ.ο.κ.) και τις ενώσεις μεταξύ αυτών των κόμβων οι οποίες μπορεί να είναι χωρίς κατεύθυνση (edges), στις περιπτώσεις δρόμων δύο κατευθύνσεων, ή να περιλαμβάνουν κατεύθυνση και άρα είναι διανύσματα (arcs), όπως για παράδειγμα όταν πρόκειται για δρόμους μίας κατεύθυνσης.

Ο σκοπός συνήθως είναι η επίσκεψη κάποιου υποσυνόλου από το συνολικό πλήθος των πελατών για την παράδοση προϊόντων, ή η συντήρηση ή επιθεώρηση δραστηριοτήτων που λαμβάνουν χώρο στον δρόμο από ένα σημείο σε ένα άλλο (πχ. Απομάκρυνση του χιονιού από το δρόμο). Έτσι, προκύπτουν δύο κατηγορίες μοντελοποιημένων προβλημάτων:

1. Τα προβλήματα δρομολόγησης Κόμβων (Node Routing Problem)
2. Τα προβλήματα δρομολόγησης Τόξων (Arc Routing Problem)

Γενικά τα αγαθά έχουν ροή από μια ή περισσότερες κεντρικές αποθήκες προς τους υπόλοιπους κόμβους-πελάτες, αν και σε κάποιες ειδικού τύπου εφαρμογές η ροή αντιστρέφεται, για παράδειγμα στην συλλογή απορριμμάτων. Πάραυτα, οι βασικές αρχές για τις δύο κατευθύνσεις της ροής είναι οι ίδιες (π.χ. η μεταφερόμενη ποσότητα δεν πρέπει να υπερβαίνει τη χωρητικότητα του αντίστοιχου οχήματος, είτε αυτό ξεκινά από κάποια αποθήκη γεμάτο και παραδίδει αγαθά, είτε το όχημα ξεκινά άδειο και φορτώνεται στην πορεία του δρομολογίου του).

Η πρώτη εμφάνιση προβλήματος δρομολόγησης κόμβων είναι το Πρόβλημα Περιοδεύοντος Πωλητή (Travelling Salesman Problem – TSP): Δίνεται ένα σύνολο κόμβων σε ένα γράφο και οι αποστάσεις όλων των κόμβων μεταξύ τους. Ο στόχος είναι να υπολογιστεί η συντομότερη διαδρομή που περιλαμβάνει όλους τους κόμβους (πλην της αφετηρίας) ακριβώς μία φορά και ξεκινάει και επιστρέφει στον κόμβο αφετηρίας, δηλαδή πρόκειται για κυκλική διαδρομή. Η πρώτη εμφάνιση του προβλήματος αυτού ήταν στις αρχές της δεκαετίας 1920-1930 από τον Karl Menger, όμως η πρώτη δημοσίευση που παρουσιάζει το πρόβλημα προτάθηκε από τον Dantzig το (1954).

Το πρόβλημα του Περιοδεύοντος Πωλητή και άλλα προβλήματα ενός οχήματος είναι ανεπαρκή για να λύσουν πραγματικές εφαρμογές που απαιτούν μεγαλύτερο αριθμό οχημάτων. Έτσι, προέκυψε το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem). Η βασική μορφή αυτού του προβλήματος λέγεται Πρόβλημα Δρομολόγησης Οχημάτων Περιορισμένης Δυναμικότητας (Capacitated Vehicle Routing Problem - CVRP) και πολλές φορές οι μελετητές την αναφέρουν απλά ως Πρόβλημα Δρομολόγησης Οχημάτων (VRP). Περιλαμβάνει το σχεδιασμό ενός συνόλου από διαδρομές οχημάτων με όσο το δυνατόν μικρότερο κόστος, που εξυπηρετεί, ξεκινώντας από κάποια κεντρική αποθήκη, ένα σύνολο από πελάτες με γνωστή ζήτηση.

2.2 Το πρόβλημα δρομολόγησης οχημάτων (VRP) και οι παραλλαγές του

Το «Πρόβλημα Δρομολόγησης Οχημάτων» (Vehicle Routing Problem - VRP) ως όρος δόθηκε από τους (Golden, Magnanti, & Nguyen, 1975) σε μία μεγάλη κατηγορία προβλημάτων βελτιστοποίησης, με σκοπό το σχεδιασμό των διαδρομών βέλτιστου συνολικού κόστους για συγκεκριμένο στόλο οχημάτων με προκαθορισμένη χωρητικότητα, που ξεκινούν από μία κεντρική αποθήκη, ώστε να εξυπηρετήσουν τη ζήτηση ενός προκαθορισμένου συνόλου πελατών, ικανοποιώντας συγκεκριμένους περιορισμούς. Εκτός της ονομασίας VRP στη βιβλιογραφία συναντάται η εναλλακτική ονομασία «Capacitated VRP – CVRP» που θεωρούνται ταυτόσημες, καθώς σε κάθε μελέτη του VRP και του συνόλου των παραλλαγών του εμπεριέχεται ο περιορισμός της χωρητικότητας των οχημάτων (Subramanian, Uchoa, & Ochi, 2013). Το VRP παίζει κυρίαρχο ρόλο στη διαχείριση διανομών και αντιμετωπίζεται σε καθημερινή βάση από δεκάδες χιλιάδες εταιρίες μεταφορών παγκοσμίως (Laporte et al., 2013).

Η πρώτη αναφορά του VRP στη βιβλιογραφία γίνεται από τους (G. Dantzig et al., 1954), οι οποίοι μελέτησαν το πρόβλημα Περιοδεύοντος Πωλητή (Travelling Salesman Problem – TSP) απλοποιημένη κατ' ουσία εκδοχή του VRP στην οποία ο στόλος οχημάτων αποτελείται από ένα και μόνο όχημα και η χωρητικότητα του οχήματος υπερβαίνει πάντα τη συνολική ζήτηση. Στη συνέχεια, οι G. B. Dantzig και Ramser (1959) και οι Clarke & Wright (1964) μελέτησαν προβλήματα δρομολόγησης με περισσότερα του ενός οχήματα, και η οποία εν τέλει θεωρήθηκε ως η πρώτη πάνω στο VRP (Eksioglu, Vural, & Reisman, 2009; Maffioli, 2003).

Πάραυτα, στην απλή μορφή του, το VRP δεν μοντελοποιεί με ακρίβεια την πραγματικότητα, δεδομένου ότι στην πράξη η δρομολόγηση διέπεται από περίπλοκους κανόνες και περιορισμούς που εκλείπουν από αυτή την αρχική μορφή του προβλήματος. 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 εν χρήση οχημάτων και δεύτερον τα οχήματα να διανύσουν αθροιστικά τη μικρότερη δυνατή διαδρομή (Labadie & Prins, 2012). Στη συνέχεια, παρουσιάζονται οι παραλλαγές αυτές (κεφάλαια 2.2.1 - 2.2.4) καθώς και οι συνδυασμοί τους (κεφάλαιο 2.2.5).

2.2.1 Πελάτες και Υπηρεσίες

2.2.1.1 VRP με πολλαπλές παραδόσεις (SDVRP)

Το πρόβλημα δρομολόγησης οχημάτων με πολλαπλές παραδόσεις (Split Deliveries Vehicle Routing Problem – SDVRP) προτάθηκε για πρώτη φορά από τους (Dror & Trudeau, 1989, 1990), ενώ οι (Archetti, Savelsbergh, & Speranza, 2006) έδειξαν πως, ενώ οι περισσότεροι πελάτες προτιμούν να εξυπηρετούνται μία φορά, οι πολλαπλές εξυπηρετήσεις μπορούν να φέρουν σημαντική μείωση του κόστους – η βέλτιστη λύση ενός SDVRP μπορεί να είναι δύο

φορές πιο φτηνή από το αντίστοιχο πρόβλημα CVRP (Capacitated VRP – πρόβλημα δρομολόγησης οχημάτων με περιορισμένη χωρητικότητα οχημάτων). Στο CVRP ο κάθε κόμβος-πελάτης πρέπει να εξυπηρετηθεί από ένα και μόνο όχημα. Αντίθετα, στην παραλλαγή του VRP με πολλαπλές παραδόσεις απαλείφεται αυτός ο περιορισμός και ο κάθε πελάτης μπορεί να εξυπηρετηθεί από περισσότερα του ενός οχήματα.

Συνεπώς, στο SDVRP δύναται να εξυπηρετηθούν πελάτες των οποίων η ζήτηση ξεπερνά τη χωρητικότητα των οχημάτων του στόλου οχημάτων που χρησιμοποιείται. Οι (Archetti, Savelsbergh, et al., 2006) προτείνουν την προσθήκη του χαρακτηριστικού της ύπαρξης ή μη, ενός ή περισσότερων πελατών η ζήτηση του οποίου να ξεπερνά τη χωρητικότητα των οχημάτων του στόλου οχημάτων και ορίζουν τις παρακάτω παραλλαγές του VRP:

- VRP : το πρόβλημα της εύρεσης των βέλτιστων διαδρομών όταν η ζήτηση του κάθε πελάτη είναι μικρότερη ή ίση με τη χωρητικότητα των οχημάτων και ο κάθε πελάτης εξυπηρετείται ακριβώς μία φορά
- SDVRP : το πρόβλημα της εύρεσης των βέλτιστων διαδρομών όταν η ζήτηση του κάθε πελάτη είναι μικρότερη ή ίση με τη χωρητικότητα των οχημάτων και δεν υπάρχει όριο στο πλήθος των εξυπηρετήσεων για τον κάθε πελάτη
- VRP+ : το πρόβλημα εύρεσης των βέλτιστων διαδρομών όταν η ζήτηση τουλάχιστον ενός πελάτη είναι μεγαλύτερη από τη χωρητικότητα των οχημάτων του στόλου και το πλήθος των εξυπηρετήσεων για το κάθε πελάτη είναι το ελάχιστο δυνατό, δηλαδή το πλήθος εξυπηρετήσεων για τον κάθε πελάτη είναι ο λόγος της ζήτησης του πελάτη προς τη χωρητικότητα των οχημάτων στρογγυλοποιημένος προς τον επόμενο μεγαλύτερο ακέραιο αριθμό
- SDVRP+ : το πρόβλημα εύρεσης των βέλτιστων διαδρομών όταν η ζήτηση τουλάχιστον ενός πελάτη είναι μεγαλύτερη από τη χωρητικότητα των οχημάτων του στόλου και δεν υπάρχει όριο στο πλήθος των εξυπηρετήσεων για τον κάθε πελάτη

Στη βιβλιογραφία έχουν προταθεί πολλές λύσεις για το SDVRP που εξετάζουν διαφορετικές εκδοχές του χρησιμοποιώντας αναλυτικές, ευρετικές και μεταευρετικές μεθόδους για την επίτευξη εύρεσης της βέλτιστης ή μιας αποδεκτής υποβέλτιστης λύσης σε ικανοποιητικά χρονικά πλαίσια (Archetti, Savelsbergh, et al., 2006; Archetti & Speranza, 2008; Archetti, Speranza, & Hertz, 2006; Archetti, Speranza, & Savelsbergh, 2008; Boudia, Prins, & Reghioui, 2007; S. Chen, Golden, & Wasil, 2007; Dror & Trudeau, 1989, 1990) .

2.2.1.2 Το VRP με παραδόσεις και παραλαβές (Pickup and Delivery VRP)

Τα θεμελιώδη προβλήματα δρομολόγησης θεωρείται ότι μοντελοποιούν κυρίως δίκτυα διανομής. Πάραυτα, είναι απολύτως εφαρμόσιμα και σε δίκτυα συλλογής αγαθών, αν θεωρηθεί ότι η ροή των αγαθών είναι αντίστροφη, όπως για παράδειγμα στη συλλογή απορριμμάτων. Συνεπώς, προέκυψε η παραλλαγή του προβλήματος δρομολόγησης οχημάτων που περιλαμβάνει παραδόσεις αλλά και παραλαβές (Pickup and Delivery Vehicle Routing Problem - PDVRP).

Στο τυπικό PDVRP ο κάθε πελάτης ζητάει είτε να του παραδοθούν αγαθά, είτε να παραληφθούν από αυτόν αγαθά, είτε και τα δύο, με τον περιορισμό τα αγαθά που θα του

παραδοθούν να προέρχονται από την κεντρική αποθήκη και τα αγαθά που θα παραληφθούν από αυτόν να καταλήξουν στην κεντρική αποθήκη. Σύμφωνα με τους (Subramanian et al., 2013) οι εκδοχές του VRP με παραδόσεις και παραλαβές είναι οι εξής:

- VRPSPD – VRP with Simultaneous Pickup and Delivery (VRP με ταυτόχρονες παραδόσεις και παραλαβές): στην συγκεκριμένη εκδοχή ο κάθε πελάτης προς εξυπηρέτηση διέπεται από προκαθορισμένη ποσότητα αγαθών προς παραλαβή και από προκαθορισμένη ποσότητα αγαθών προς παράδοση (Subramanian, Drummond, Bentes, Ochi, & Farias, 2010; Zachariadis & Kiranoudis, 2011).
- VRPMPD – VRP with Mixed Pickup and Delivery (VRP με μικτές παραδόσεις και παραλαβές): στη συγκεκριμένη εκδοχή ο κάθε πελάτης προς εξυπηρέτηση διέπεται είτε από προκαθορισμένη ποσότητα αγαθών προς παραλαβή είτε από προκαθορισμένη ποσότητα αγαθών προς παράδοση (Gajpal & Abad, 2009; Ropke & Pisinger, 2006).

Επιπροσθέτως, έχει μελετηθεί και η παραλλαγή του VRPPD στην οποία δεν υπάρχει απαραίτητα κεντρική αποθήκη και η κάθε παραγγελία για αγαθά προς εξυπηρέτηση χαρακτηρίζεται από προκαθορισμένο αποστολέα και παραλήπτη (Parragh, Doerner, & Hartl, 2008; Yanik, Bozkaya, & deKervenoael, 2014).

2.2.2 Στόλος Οχημάτων και Κέντρο Διανομής

2.2.2.1 VRP με ανομοιογενή στόλο οχημάτων (Heterogeneous Fleet VRP)

Οι περισσότεροι στόλοι οχημάτων συνδυάζουν αρκετούς τύπους οχημάτων, κυρίως λόγω της σταδιακής αντικατάστασής τους αλλά και για την αποτελεσματικότερη λειτουργία τους. Επομένως, αναπτύχθηκε η παραλλαγή του VRP με ανομοιογενή στόλο οχημάτων (Heterogeneous Fleet Vehicle Routing Problem - HFVRP) στην οποία ο στόλος οχημάτων αποτελείται από προκαθορισμένους τύπων οχημάτων, ενώ ο κάθε τύπος οχήματος αντιστοιχεί σε συγκεκριμένη χωρητικότητα.

Εκτός αυτών των χαρακτηριστικών για τον κάθε τύπο οχήματος που συναντώνται στην πρώτη εκδοχή του HFVRP, μελετητές του προβλήματος έχουν προσθέσει περισσότερα χαρακτηριστικά όπως εφάπαξ κόστος στην περίπτωση που το όχημα χρησιμοποιηθεί (Fixed Cost), προκαθορισμένο μεταβλητό κόστος ανά μονάδα διανυόμενης απόστασης (Variable Cost) και περιορισμένο αριθμό διαθέσιμων οχημάτων του κάθε είδους (Fleet Size and Mix - FSM) (Gendreau, Laporte, et al., 1999; Lima, Goldbarg, & Goldbarg, 2004; Soonpracha, Mungwattana, Janssens, & Manisri, 2014).

Σύμφωνα με τους (Subramanian, Penna, Uchoa, & Ochi, 2012), οι εκδοχές του HFVRP είναι οι ακόλουθες:

- HVRPFV – Fixed and Variable Cost: με εφάπαξ κόστος στην περίπτωση που το όχημα χρησιμοποιηθεί, προκαθορισμένο μεταβλητό κόστος ανά μονάδα διανυόμενης απόστασης, και περιορισμένο αριθμό διαθέσιμων οχημάτων του κάθε είδους

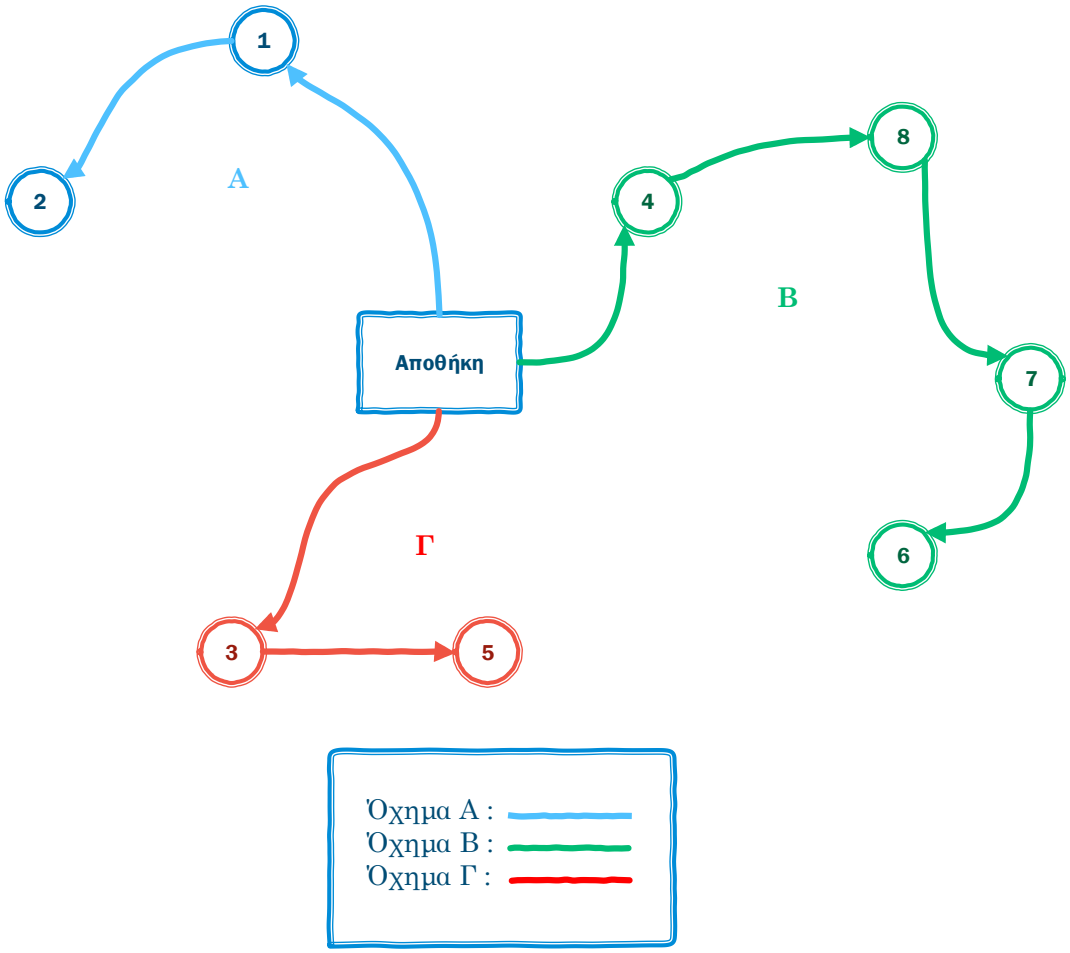
- HVRPV – Variable Cost: περιορισμένο αριθμό διαθέσιμων οχημάτων του κάθε είδους, προκαθορισμένο μεταβλητό κόστος ανά μονάδα διανυόμενης απόστασης, χωρίς ΕΚΧ
- FSMFV – Fixed and Variable Cost: εφάπαξ κόστος στην περίπτωση που το όχημα χρησιμοποιηθεί, προκαθορισμένο μεταβλητό κόστος ανά μονάδα διανυόμενης απόστασης, χωρίς περιορισμένο αριθμό διαθέσιμων οχημάτων του κάθε είδους
- FSMF – Fixed Cost: εφάπαξ κόστος στην περίπτωση που το όχημα χρησιμοποιηθεί, χωρίς περιορισμένο αριθμό διαθέσιμων οχημάτων του κάθε είδους και χωρίς προκαθορισμένο μεταβλητό κόστος ανά μονάδα διανυόμενης απόστασης
- FSMV – Variable Cost: προκαθορισμένο μεταβλητό κόστος ανά μονάδα διανυόμενης απόστασης, χωρίς περιορισμένο αριθμό διαθέσιμων οχημάτων του κάθε είδους και χωρίς εφάπαξ κόστος στην περίπτωση που το όχημα χρησιμοποιηθεί

Μερικοί ακόμη μελετητές του HFVRP είναι οι (Brandão, 2011; Leung, Zhang, Zhang, Hua, & Lim, 2013; Prins, 2002, 2009).

2.2.2.2 Ανοιχτό VRP (Open VRP)

Το «Ανοιχτό VRP» (Open Vehicle Routing Problem – OVRP) πρωτοεμφανίζεται στη βιβλιογραφία μόλις το 2000 από τους (Sariklis & Powell, 2000) οι οποίοι το ορίζουν ως την παραλλαγή του VRP στην οποία τα οχήματα μετά το πέρας του δρομολογίου τους δεν απαιτείται να επιστρέψουν στην κεντρική αποθήκη, ενώ στην περίπτωση που επιστρέφουν στην αποθήκη, επιστρέφουν από τη διαδρομή που ακολούθησαν για την εξυπηρέτηση των πελατών. Δηλαδή, η διαδρομή του κάθε οχήματος δε σχηματίζει κύκλο Hamilton (κλειστή διαδρομή) όπως στο VRP, αντίθετα σχηματίζει μονοπάτι Hamilton (ανοιχτή διαδρομή) (W. Wang, Wu, Zhao, & Feng, 2006). Συνεπώς, το πρόβλημα OVRP ανάγεται στο πρόβλημα εύρεσης του βέλτιστου μονοπατιού Hamilton (Hamiltonian Path), μετά τον καταμερισμό των πελατών στο κάθε όχημα (P. P. Repoussis, Tarantilis, Bräysy, & Ioannou, 2010).

Για την καλύτερη κατανόηση του OVRP παρουσιάζεται ένα παράδειγμα οκτώ πελατών και τριών οχημάτων στο οποίο φαίνεται πως οι διαδρομές των οχημάτων από κάποια κεντρική αποθήκη είναι «ανοιχτές», δηλαδή αποτελούν Μονοπάτι Hamilton.



Σχήμα 2.1 : Παράδειγμα OVRP

Η παραλλαγή αυτή χρησιμοποιείται σε πολλά πεδία, όπως στην περίπτωση συνεργασία με εταιρείες εφοδιαστικής τρίτου προσώπου (3PL), στη διανομή εφημερίδων κατ' οίκον και στη μεταφορά μαθητών από ιδιωτικά λεωφορεία, διότι οι εταιρίες αυτού του είδους προσλαμβάνουν οδηγούς με ιδιόκτητα οχήματα, συνεπώς δεν απαιτείται να επιστρέψουν στην κεντρική αποθήκη στο τέλος της βάρδιας τους (W. Wang et al., 2006).

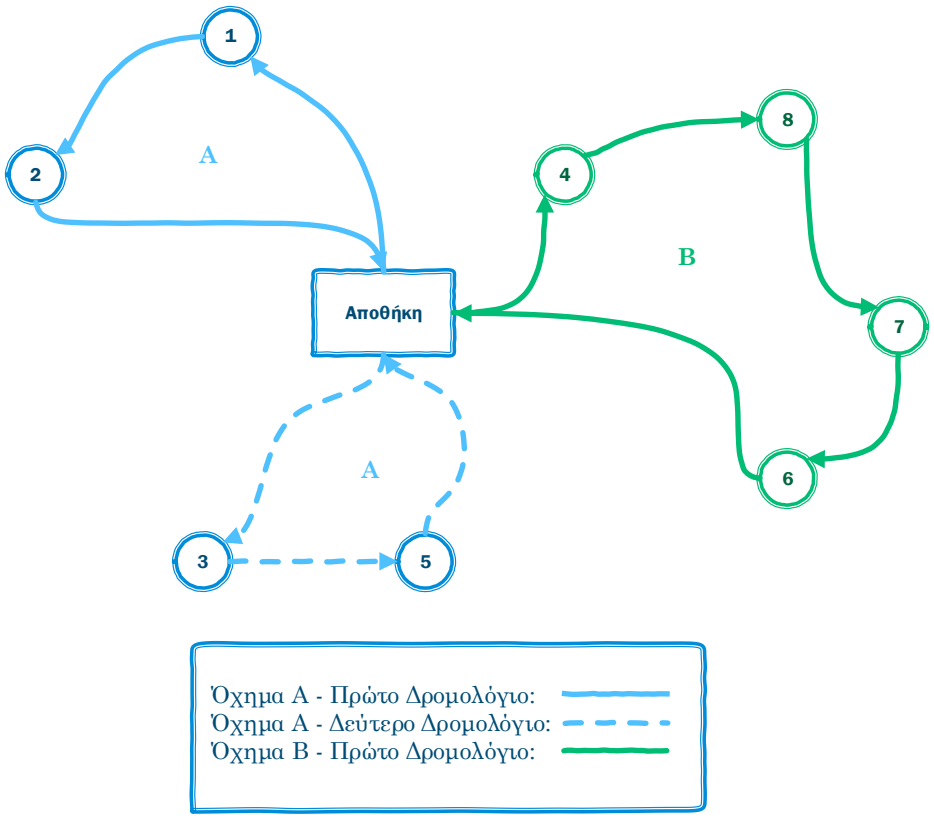
Η πλειονότητα των μελετητών του OVRP, θεωρούν ότι το κόστος της χρήσης επιπλέον οχημάτων ξεπερνά κατά πολύ το μεταβλητό κόστος κίνησης των οχημάτων, συνεπώς βελτιστοποιείται αρχικά το πλήθος των οχημάτων-διαδρομών και στη συνέχεια η συνολική απόσταση που διανύεται αθροιστικά από τα οχήματα (Fleszar, Osman, & Hindi, 2009; Z Fu, Eglese, & Li, 2006; Zhuo Fu, Eglese, & Li, 2005; Li, Golden, & Wasil, 2007; Zachariadis & Kiranoudis, 2010).

2.2.2.3 VRP με πολλαπλά δρομολόγια (Multi-Trip VRP)

Στο πρόβλημα δρομολόγησης οχημάτων με πολλαπλά δρομολόγια (Multi-Trip Vehicle Routing Problem – MTVRP), αντίθετα με το Πρόβλημα Δρομολόγησης Οχημάτων (VRP) στο οποίο το κάθε όχημα εκτελεί ένα και μόνο δρομολόγιο, δύναται ένα ή περισσότερα οχήματα να εκτελέσουν περισσότερα από ένα δρομολόγια, στην περίπτωση που ο αριθμός των δρομολογίων ξεπεράσει το μέγεθος του στόλου οχημάτων, ενώ η πρώτη βιβλιογραφική μελέτη που περιλαμβάνει το MTVRP είναι αυτή του Fleischmann (1990). Η μοντελοποίηση προβλημάτων διανομών ως MTVRP συναντάται κυρίως σε πρακτικές εφαρμογές στις οποίες απαιτείται μικρό μέγεθος οχημάτων– συνεπώς μικρή χωρητικότητα- συνεπώς η πιθανότητα της επιστροφής των οχημάτων στην κεντρική αποθήκη νωρίτερα του τέλους της ημέρας είναι μεγάλη και τα οχήματα δύναται να εκτελέσουν περισσότερα του ενός δρομολόγια (Battarra, Monaci, & Vigo, 2009). Στη συνέχεια, παρουσιάζονται οι περιορισμοί που χαρακτηρίζουν το MTVRP σύμφωνα με τους Cattaruzza, Absi, Feillet, και Vidal (2014).

- Η κάθε διαδρομή του κάθε οχήματος ξεκινά και καταλήγει στην κεντρική αποθήκη
- Ο κάθε πελάτης εξυπηρετείται κατά τη διάρκεια μίας και μόνο διαδρομής ενός και μόνο οχήματος
- Η ζήτηση των πελατών που εξυπηρετούνται από το ίδιο όχημα κατά τη διάρκεια της ίδιας διαδρομής του δεν ξεπερνά αθροιστικά τη χωρητικότητα του οχήματος
- Η συνολική διάρκεια των διαδρομών που διανύει το κάθε όχημα δεν ξεπερνά μία προκαθορισμένη τιμή

Στο Σχήμα 2.2 παρουσιάζεται παράδειγμα ενός MTVRP με δύο οχήματα στο οποίο το πρώτο όχημα (A) εκτελεί δύο δρομολόγια δύο πελατών το καθένα (το δρομολόγιο 1-2 και το δρομολόγιο 3-5) ενώ το δεύτερο όχημα (B) εκτελεί ένα δρομολόγιο τεσσάρων πελατών (το δρομολόγιο 4-8-7-6).



Σχήμα 2.2 : Παράδειγμα MTVRP

Το MTVRP έχει μελετηθεί αρκετά τις τελευταίες δύο δεκαετίες και έχουν προταθεί αρκετοί, κυρίως μεταευρετικοί μέθοδοι, για την επίλυσή του (Brandão & Mercer, 1997; Cattaruzza, Absi, Feillet, Guyon, & Libeaut, 2013; Petch & Salhi, 2003; Prins, 2002; Salhi & Petch, 2007; Şen & Bülbül, 2008).

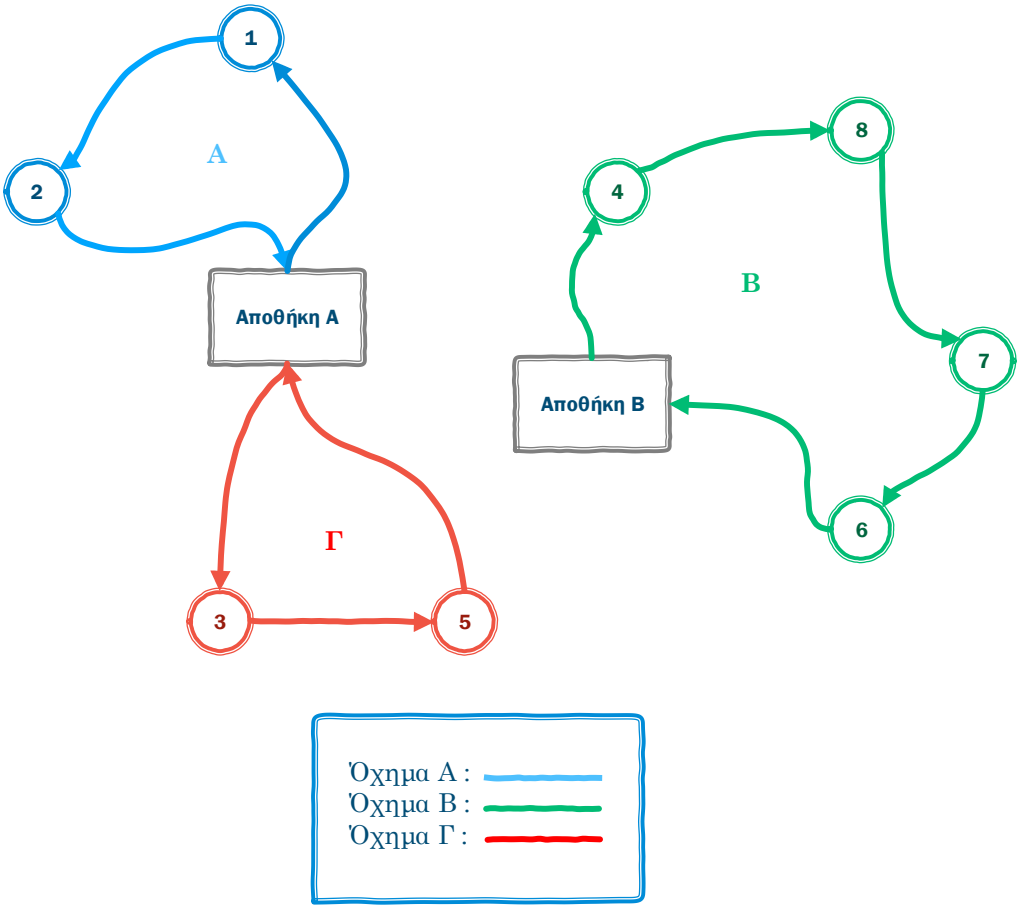
2.2.2.4 VRP με πολλαπλές αποθήκες (Multi Depot VRP)

Η συγκεκριμένη παραλλαγή του προβλήματος δρομολόγησης οχημάτων με πολλαπλές αποθήκες (Multi Depot Vehicle Routing Problem – MDVRP) μελετήθηκε αρχικά τη δεκαετία 1970-1980 (Gillett & Johnson, 1976; Tillman & Cain, 1972; Wren & Holliday, 1972) και περιλαμβάνει περισσότερες από μία κεντρικές αποθήκες από τις οποίες ξεκινούν και τερματίζουν τις διαδρομές τους τα οχήματα προς δρομολόγηση. Παρουσιάζονται οι περιορισμοί που διέπουν το MDVRP σύμφωνα με τους Escobar, Linfati, Toth και Baldoquin (2014) :

- τα οχήματα στο τέλος του δρομολογίου που εκτελούν για την εξυπηρέτηση των πελατών επιστρέφουν στην αποθήκη από την οποία ξεκίνησαν
- κάθε πελάτης πρέπει να εξυπηρετηθεί ακριβώς μία φορά και από ένα και μόνο όχημα
- η συνολική ζήτηση που ικανοποιεί το κάθε όχημα δεν πρέπει να ξεπερνά τη χωρητικότητα του οχήματος αυτού

- το πλήθος των οχημάτων που ξεκινούν τη διαδρομή τους από την κάθε αποθήκη δεν πρέπει να ξεπερνούν μία προκαθορισμένη τιμή
- η συνολική διάρκεια της διαδρομής του κάθε οχήματος δεν πρέπει να ξεπερνά μία προκαθορισμένη τιμή

Για την καλύτερη κατανόηση του MDVRP παρουσιάζεται ένα παράδειγμα οκτώ πελατών και δύο αποθηκών:



Σχήμα 2.3 : Παράδειγμα MDVRP

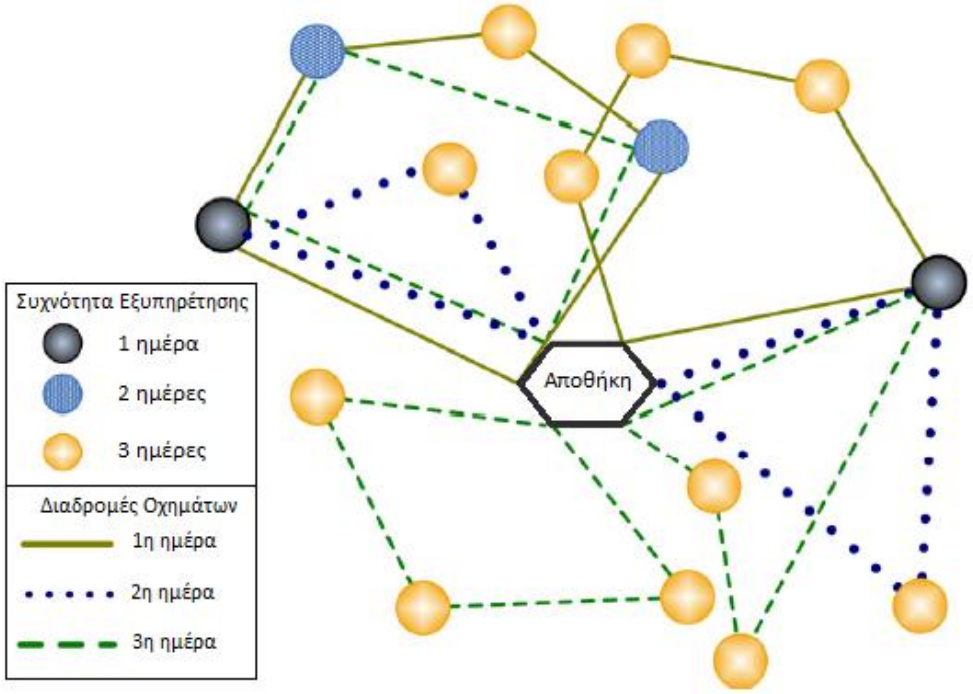
Το MDVRP δύναται να μοντελοποιηθεί ως PVRP (κεφάλαιο 2.2.3), καθώς οι πολλαπλές αποθήκες του MDVRP παραλληλίζονται με τις πολλαπλές περιόδους του PVRP, συνεπώς κάθε αλγόριθμος που λύνει PVRP μπορεί εξίσου να λύσει MDVRP (Contardo & Martinelli, 2014). Το MDVRP έχει μελετηθεί εκτενώς και έχουν προταθεί τόσο αναλυτικές όσο και ευρετικές – μεταευρετικές προσεγγίσεις για την επίλυσή του (Baldacci & Mingozzi, 2009; Gulczynski, Golden, & Wasil, 2011; Kuo & Wang, 2012).

2.2.3 Χρονικός Ορίζοντας Δρομολόγησης

2.2.3.1 Το VRP με πολλαπλές περιόδους (Periodic VRP)

Η ανάγκη για σχεδιασμό δρομολογίων για πολλαπλές περιόδους (για κάθε ημέρα της εβδομάδας, ή για κάθε εβδομάδα του μήνα) οδήγησε στην παραλλαγή του προβλήματος δρομολόγησης οχημάτων με πολλαπλές περιόδους (Periodic Vehicle Routing Problem – PVRP). Η διαφορά του PVRP με τις υπόλοιπες παραλλαγές του VRP έγκειται στο χαρακτηριστικό του χρονικού ορίζοντα και της περιόδου: στο PVRP πρέπει να εξυπηρετηθεί ένα προκαθορισμένο πλήθος πελατών με συγκεκριμένη συχνότητα εξυπηρέτησης ανά χρονικό ορίζοντα και συγκεκριμένη ζήτηση ανά εξυπηρέτηση, ενώ απαιτείται ο σχεδιασμός δρομολογίων για πολλαπλές χρονικές περιόδους οι οποίες αθροιστικά αποτελούν τον χρονικό ορίζοντα προγραμματισμού (Christofides & Beasley, 1984).

Για την καλύτερη κατανόηση της παραλλαγής αυτής παρουσιάζεται στο Σχήμα 2.4 το παράδειγμα του Yu and Yang (2011) όπου ο χρονικός ορίζοντας σχεδιασμού των δρομολογίων είναι τρεις ημέρες, η περίοδος δρομολόγησης είναι μία ημέρα, ενώ η συχνότητα εξυπηρέτησης των πελατών είναι είτε καθημερινή είτε κάθε δύο είτε κάθε τρεις ημέρες.



Σχήμα 2.4 : Παράδειγμα PVRP (Yu & Yang, 2011)

Το VRP με πολλαπλές περιόδους μελέτησαν μεταξύ άλλων οι Francis, Smilowitz, και Tzur (2008) οι οποίοι μελέτησαν διάφορες παραλλαγές του VRP με πολλαπλές περιόδους, όπως την παραλλαγή όπου η ζήτηση του κάθε πελάτη εξαρτάται από το χρόνο που έχει περάσει από την τελευταία φορά που εξυπηρετήθηκε. Το PVRP έχει μελετηθεί εκτενώς, κυρίως τα

τελευταία δεκαπέντε έτη (Amberg, Domschke, & Voß, 2000; Ombuki-Berman & Hanshar, 2009).

2.2.4 Στοχαστικά και Δυναμικά Προβλήματα

Στην πλειονότητα των παραλλαγών του προβλήματος δρομολόγησης οχημάτων τα δεδομένα του κάθε προβλήματος είναι ντετερμινιστικά (D. J. Bertsimas, 1992), θεωρείται δηλαδή ότι έχουν συγκεκριμένη τιμή η οποία δε θα μεταβληθεί κατά τη διάρκεια, ή μετά το πέρας του σχεδιασμού των δρομολογίων των οχημάτων. Πάραυτα, στην πράξη, η πληροφορία έχει δύο θεμελιώδη χαρακτηριστικά: την ποιότητα και την πιθανή εξέλιξή της (Psaraftis, 1980). Συνεπώς, δεδομένα όπως ο χρόνος ταξιδιού των οχημάτων από και προς την αποθήκη και τους πελάτες, η ζήτηση των πελατών και ο χρόνος εξυπηρέτησής τους, αλλά ακόμα και το πλήθος των οχημάτων ή των πελατών διέπονται από ισχυρή αβεβαιότητα και πιθανά μεταβάλλονται κατά τη διάρκεια της εκτέλεσης των δρομολογίων των οχημάτων (Labadie & Prins, 2012). Επομένως, εμφανίζονται στη βιβλιογραφία παραλλαγές του προβλήματος δρομολόγησης οχημάτων στις οποίες συνεκτιμούνται η αβεβαιότητα των αρχικών δεδομένων του προβλήματος ως στοχαστικά προβλήματα δρομολόγησης οχημάτων (Stochastic Vehicle Routing Problem - SVRP) αλλά και η μετατροπή των δρομολογίων κατά τη διάρκεια της εκτέλεσής τους ως δυναμικά προβλήματα δρομολόγησης οχημάτων (Dynamic Vehicle Routing Problem - DVRP).

2.2.4.1 Στοχαστικό VRP (Stochastic VRP)

Πρώτος μελετητής του στοχαστικού προβλήματος δρομολόγησης οχημάτων (Stochastic Vehicle Routing Problem – SVRP) είναι ο Tillman (1969) ο οποίος πρότεινε μία μέθοδο “Savings” για το πρόβλημα SVRP με πολλαπλές αποθήκες (Multi-Depot Stochastic Vehicle Routing Problem - MDSVRP). Στο MDSVRP, εκτός της ύπαρξης πολλαπλών αποθηκών, η ζήτηση του κάθε πελάτη δεν είναι ντετερμινιστικά προκαθορισμένη κατά τη σχεδίαση των δρομολογίων, πάραυτα ακολουθεί κανονική κατανομή με προκαθορισμένο μέσο όρο και τυπική απόκλιση (Moghaddam, Ruiz, & Sadjadi, 2012).

Στο SVRP μοντελοποιείται η αβεβαιότητα των αρχικών δεδομένων των πελατών, των οχημάτων και της αποθήκης αντικαθιστώντας τη ντετερμινιστική τιμή τους από μία στατιστική κατανομή με γνωστό μέσο όρο και τυπική απόκλιση. Σύμφωνα με τους Cordeau, Laporte, Savelsbergh, και Vigo (2006) και τους Victor Pillac, Gendreau, Guéret, και Medaglia (2013), έχουν μελετηθεί τρεις θεμελιώδης παράγοντες αβεβαιότητας των αρχικών δεδομένων του VRP:

- Η αβεβαιότητα εξυπηρέτησης των πελατών – στην εκδοχή αυτή του SVRP ο κάθε πελάτης διέπεται από προκαθορισμένη πιθανότητα εξυπηρέτησης ή μη εξυπηρέτησης (D. Bertsimas, 1988; Waters, 1989)
- Η αβεβαιότητα των χρόνων εξυπηρέτησης και των χρόνων ταξιδιού – στην εκδοχή αυτή ο χρόνος που θα χρειαστεί για την εξυπηρέτηση του κάθε πελάτης αλλά και ο χρόνος που απαιτείται για την μετακίνηση των οχημάτων ακολουθούν κάποια στατιστική κατανομή με προκαθορισμένο μέσο όρο και τυπική απόκλιση (Kenyon &

Morton, 2003; Laporte, Louveaux, & Mercure, 1992; Verweij, Ahmed, Kleywegt, Nemhauser, & Shapiro, 2003).

- Η αβεβαιότητα της ζήτησης των πελατών – στην εκδοχή αυτή του SVRP η ζήτηση του κάθε πελάτη αντί συγκεκριμένης τιμής διέπεται από προκαθορισμένη μέση αναμενόμενη τιμή και τυπική απόκλιση (Christiansen & Lysgaard, 2007; Dror, Laporte, & Trudeau, 1989; Laporte, Louveaux, & Van Hamme, 2002; Mendoza, Castanier, Guéret, Medaglia, & Velasco, 2010)

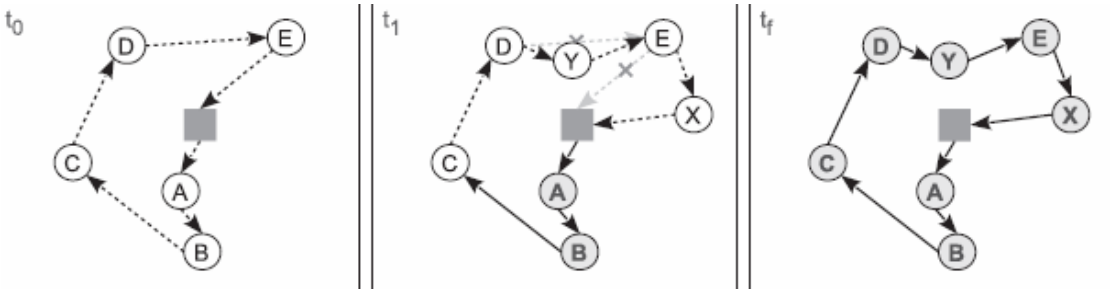
Οι πιο συνήθεις τακτικές για την επίλυση προβλήματος VRP με στοχαστική ζήτηση είναι ο προγραμματισμός με περιορισμούς πιθανοτήτων (Chance Constrained Programming – CCP) και ο στοχαστικός προγραμματισμός με ανάδραση (Stochastic Programming with Recourse). Οι δύο παραπάνω τακτικές είναι βασισμένες σε μία διαδικασία δύο φάσεων. Στην πρώτη, υπολογίζεται ένα αρχικό πλάνο δρομολόγησης και στη δεύτερη φάση, πραγματοποιούνται διορθωτικές κινήσεις κατά τη διάρκεια που εκτελείται το αρχικό πλάνο δρομολόγησης. Η κύρια διαφορά μεταξύ των δύο τακτικών είναι ο αντικειμενικός στόχος της πρώτης φάσης. Η τακτική CCP εγγυάται ένα άνω όριο στην πιθανότητα αποτυχίας του πλάνου δρομολόγησης, όμως δεν ελέγχει το αναμενόμενο κόστος από τις διορθωτικές κινήσεις της δεύτερης φάσης, ενώ η τακτική SPR επιδιώκει την ελαχιστοποίηση του κόστους των διορθωτικών κινήσεων της δεύτερης φάσης (Gendreau, Laporte, & Séguin, 1996; Labadie & Prins, 2012).

Έχουν αναπτυχθεί διάφορες εναλλακτικές προσεγγίσεις για την επίλυση του SVRP (Gendreau et al., 1996; Park & Hong, 2003; Roberts & Hadjiconstantinou, 1998; Shen, Ordónez, & Dessouky, 2009; Sungur, Ordónez, & Dessouky, 2008).

2.2.4.2 Δυναμικό VRP (Dynamic VRP)

Το Δυναμικό Πρόβλημα Δρομολόγησης Οχημάτων (Dynamic Vehicle Routing Problem – VRP) περιλαμβάνει τη δυνατότητα εισαγωγής νέων ή μετατροπής των προκαθορισμένων δεδομένων κατά τη διάρκεια είτε του σχεδιασμού είτε της εκτέλεσης του πλάνου δρομολόγησης (Victor Pillac et al., 2013). Για την επίλυση τέτοιων προβλημάτων, κατά τη διάρκεια εκτέλεσης των δρομολογίων εκτελείται ανά διαστήματα κάποιος αλγόριθμος που προσθέτει τα νέα δεδομένα στο μοντέλο του προβλήματος ή μετατρέπει τα προκαθορισμένα δεδομένα και επανα-βελτιστοποιεί το αρχικό πλάνο δρομολόγησης.

Για την καλύτερη κατανόηση του δυναμικού VRP παρουσιάζεται ένα παράδειγμα του Victor Pillac (2013) στο Σχήμα 2.5 με στόλο οχημάτων ενός και μόνο οχήματος. Στην πρώτη εικόνα παρουσιάζεται το δρομολόγιο του οχήματος κατά την εκτέλεση του αρχικού πλάνου δρομολόγησης (A-B-C-D-E), στη δεύτερη εικόνα παρουσιάζονται οι δύο νέοι πελάτες (πελάτες X και Y) προς δρομολόγηση και στην τρίτη παρουσιάζεται η μετατροπή του αρχικού δρομολογίου του οχήματος ώστε να τους εξυπηρετήσει (A-B-C-D-Y-E-X).



Σχήμα 2.5 : Παράδειγμα DVRP (Victor Pillac et al., 2013)

Η πρώτη βιβλιογραφική αναφορά σε δυναμικό πρόβλημα δρομολόγησης είναι από τους Wilson and Colvin (1977) για το δυναμικό πρόβλημα δρομολόγησης τόξων (Dynamic Arc Routing Problem – DARP) που αντιστοιχεί άμεσα στο DVRP.

Ο αντικειμενικός στόχος για τη βελτιστοποίηση ενός DVRP είναι πολλαπλός: εκτός της ελαχιστοποίησης της συνολικής απόστασης που διανύουν τα οχήματα, πρέπει να ελαχιστοποιηθεί ο μέσος χρόνος απόκρισης του συστήματος μετά από νέα παραγγελία, η πιθανότητα απόρριψης της νέας παραγγελίας αλλά και περισσότερες μεταβλητές ανάλογα με το υπό μελέτη πρόβλημα, όπως για παράδειγμα στη δρομολόγηση μέσων μαζικής μεταφοράς, ο μέσος χρόνος παραμονής ενός επιβάτη μέσα σε όχημα, (Ferrucci, Bock, & Gendreau, 2013; Gendreau, Guertin, Potvin, & Taillard, 1999; Montemanni, Gambardella, Rizzoli, & Donati, 2005; Novoa & Storer, 2009; V Pillac, Guéret, & Medaglia, 2010; Powell, 1986; Psaraftis, 1995; van Hemert & La Poutré, 2004)

2.2.5 Εμπλουτισμένο VRP (Rich VRP)

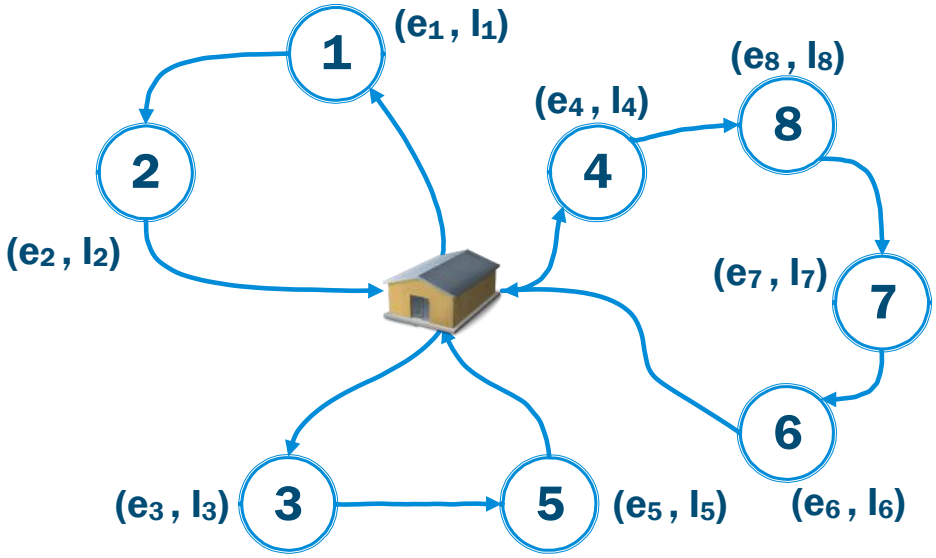
Το πρώτο βήμα στην γεφύρωση της ακαδημαϊκής έρευνας και των πραγματικών εφαρμογών δρομολόγησης είναι η κατάστρωση προβλημάτων που συνδυάζουν δύο ή περισσότερες από τις παραπάνω παραλλαγές του VRP - το πρόβλημα που προκύπτει ονομάζεται Εμπλουτισμένο Πρόβλημα Δρομολόγησης Οχημάτων (Rich Vehicle Routing Problem – RVRP). Τα περισσότερα εμπορικά λογισμικά έχουν τη δυνατότητα να συνδυάσουν πολλαπλές παραλλαγές του VRP, δηλαδή να λύσουν RVRP, όμως δεν χρησιμοποιούν αλγορίθμους τελευταίας τεχνολογίας. Συγκεκριμένα, χρησιμοποιούν κατασκευαστικούς ευρετικούς αλγορίθμους που, ως «γενικοί» αλγόριθμοι, υλοποιούνται και συντηρούνται ευκολότερα. Οι αποδοτικότεροι και αποτελεσματικότεροι ακαδημαϊκοί αλγόριθμοι κατασκευάζονται για συγκεκριμένη παραλλαγή του VRP και μπορεί να χάσουν την αποδοτικότητα και την αποτελεσματικότητά τους σε περίπτωση εφαρμογής τους σε RVRP (Labadie & Prins, 2012). Πάραυτα, η τάση στην ακαδημαϊκή έρευνα είναι προς τη μελέτη συνδυασμών των παραλλαγών του VRP ώστε να πλησιάσουν τις πραγματικές εφαρμογές.

2.3 Το VRP με παράθυρα χρόνου (TW)

Η πλειονότητα των πραγματικών εφαρμογών δρομολόγησης αφορούν πελάτες που απαιτούν παράδοση αγαθών σε συγκεκριμένο παράθυρο χρόνου, για παράδειγμα όταν οι πελάτες είναι καταστήματα, κατά τη διάρκεια του ωραρίου λειτουργίας τους. Συχνά, οι πελάτες προσδιορίζουν αρκετά στενά παράθυρα χρόνου, για παράδειγμα όταν έχουν

παραδόσεις από διαφορετικούς προμηθευτές και θέλουν να εξυπηρετηθούν από τον κάθε προμηθευτή ξεχωριστά. Έτσι, προέκυψε η παραλλαγή του VRP με παράθυρα χρόνου (Vehicle Routing Problem with Time Windows) με πρώτο μελετητή του τον (Marius Mihai Solomon, 1984).

Στο Σχήμα παρουσιάζεται ένα παράδειγμα VRP με Παράθυρα Χρόνου οκτώ πελατών και τριών οχημάτων, όπου το παράθυρο χρόνου σε κάθε πελάτη i αναπαριστάται μια χρονική στιγμή νωρίτερης δυνατής εξυπηρέτησης e_i και μια χρονική στιγμή αργότερης δυνατής εξυπηρέτησης l_i.



Σχήμα 2.6 Παράδειγμα VRP με Παράθυρα Χρόνου

Το VRPTW είναι αναμφίβολα η πιο πολυμελετημένη παραλλαγή του VRP και παρουσιάζει πρόκληση στη δρομολόγηση οχημάτων ενώ οι περισσότεροι μελετητές θεωρούν ως πρωταρχικό στόχο την ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και δευτερεύον τη συνολική απόσταση που διανύουν τα οχήματα. Στις περισσότερες μελέτες του VRPTW δεν ελαχιστοποιείται ο χρόνος που απαιτείται για την ολοκλήρωση της κάθε διαδρομής από τα οχήματα. Ο χρόνος αυτός χρησιμοποιείται αποκλειστικά για τον έλεγχο εγκυρότητας της διαδρομής του κάθε οχήματος με βάση τα παράθυρα χρόνου των πελατών που εξυπηρετεί το όχημα.

Ανάλογα την εκδοχή του VRPTW, τα παράθυρα χρόνου των πελατών είναι είτε Αυστηρά Παράθυρα Χρόνου (Hard Time Windows) όπου τα οχήματα δεν επιτρέπεται να φθάσουν σε κάποιον πελάτη μετά το πέρας του χρονικού παραθύρου, είτε Ελαστικά Παράθυρα Χρόνου (Soft Time Windows) όπου τα οχήματα μπορεί να εξυπηρετήσουν κάποιον πελάτη μετά το πέρας του χρονικού του παραθύρου με κάποια επιβάρυνση κόστους (penalty cost).

Έχουν αναπτυχθεί αρκετοί αποδοτικοί αναλυτικοί αλγόριθμοι – μπορούν να λύσουν προβλήματα μέχρι εκατό πελατών, και κατά περίπτωση ακόμη και μεγαλύτερα (Jepsen, Petersen, Spoorendonk, & Pisinger, 2008). Μερικοί μελετητές του VRPTW που ανέπτυξαν μεταευρετικούς αλγορίθμους είναι οι (Labadi, Prins, & Reghioui, 2008; Nagata, Bräysy, & Dullaert, 2010; Panagiotis P Repoussis, Tarantilis, & Ioannou, 2009).

2.4 Μέθοδοι επίλυσης προβλημάτων δρομολόγησης με παράθυρα χρόνου

Το πρόβλημα VRPTW είναι τύπου πολυπλοκότητας NP-hard διότι το VRP, το οποίο έχει αποδειχθεί ότι είναι τύπου πολυπλοκότητας NP-hard (Lenstra & Kan, 1981), αποτελεί υποπερίπτωσή του VRPTW στην περίπτωση που όλοι οι πελάτες έχουν απεριόριστο παράθυρο χρόνου. Συγκεκριμένα, η διερεύνηση της ύπαρξης ή μη εφικτών λύσεων καθώς και η εύρεση μίας εφικτής λύσης σε κάποιο πρόβλημα VRPTW είναι προβλήματα τύπου δυσκολίας NP-Complete (Savelsbergh, 1985).

Για τα δύσκολα προβλήματα βελτιστοποίησης όπως το VRPTW, έχουν αναπτυχθεί τρεις κατηγορίες μεθόδων επίλυσης (Panagiotis P Repoussis et al., 2009):

- Οι ακριβείς ή αναλυτικοί αλγόριθμοι (exact algorithms) που βρίσκουν πάντοτε τη βέλτιστη λύση, πάραυτα καθυστερούν σε προβλήματα μεγάλης έκτασης
- Οι ευρετικοί αλγόριθμοι (heuristic algorithms) οι οποίοι παρέχουν συγκριτικά ταχέως μία εφικτή λύση με αποδεκτή ποιότητα
- Οι μεταευρετικοί αλγόριθμοι (metaheuristic algorithms) οι αποτελούν μεθόδους που χρησιμοποιούν ευρετικές μεθόδους με ειδικές τακτικές ώστε να οδηγηθούν, μετά από πολλαπλά «τρεξίματα» σε καλύτερης ποιότητας λύσεις

Στη συνέχεια, παρουσιάζονται οι αλγόριθμοι επίλυσης των τριών αυτών κατηγοριών που έχουν χρησιμοποιηθεί για την επίλυση του VRPTW στα κεφάλαια 2.4.1 - 2.4.3, με ιδιαίτερη έμφαση στις ευρετικές και μεταευρετικές μεθόδους, καθώς χρησιμοποιήθηκαν στην παρούσα μελέτη.

2.4.1 Αναλυτικές Μέθοδοι

Οι αναλυτικές μέθοδοι (ή ακριβείς μέθοδοι) είναι συνήθως είτε αλγόριθμοι γραμμικού ή δυναμικού προγραμματισμού (linear or dynamic programming) είτε τεχνικές περιορισμού και διακλάδωσης (branch and bound techniques). Οι μέθοδοι αυτοί μπορεί να βρίσκουν πάντοτε τη βέλτιστη λύση, όμως έχουν το μειονέκτημα των μεγάλων υπολογιστικών χρόνων, ειδικά σε προβλήματα μεγάλου αριθμού πελατών, που καθιστά τη χρήση τους πολλές φορές ασύμφορη.

Οι αναλυτικές μέθοδοι που έχουν αναπτυχθεί είναι βασισμένοι στη βελτιστοποίηση δικτύων, στο γραμμικό και στον ακέραιο προγραμματισμό. Οι κύριες κατευθύνσεις των αναλυτικών μεθόδων είναι ο δυναμικός προγραμματισμός, η «χαλάρωση» περιορισμών του Lagrange (Langrangian Relaxation) και η μέθοδος παραγωγής στηλών (Column Generation). Οι δύο τελευταίες κατευθύνσεις στηρίζονται στην αρχή της αποικοδόμησης (decomposition), δηλαδή το κύριο πρόβλημα διασπάται σε δύο ή περισσότερα προβλήματα για τη διευκόλυνση της επίλυσής του.

Στη μέθοδο Langrangian Relaxation επιλεγμένοι περιορισμοί «χαλαρώνουν», δηλαδή αφαιρούνται από το σύνολο των περιορισμών και μετατρέπονται σε όρους που προσθέτονται στην αντικειμενική συνάρτηση κατά κανόνα με τη μορφή ποινών.

Συγκεκριμένα, ο κάθε όρος της αντικειμενικής συνάρτησης που έχει προκύψει από χαλάρωση κάποιου περιορισμού πολλαπλασιάζεται με έναν συντελεστή «ποινής» (penalty factor) που αντιστοιχεί στο συντελεστή «λ» του Langrange (Fisher, 1985; Geoffrion, 1974).

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

- Το κύριο πρόβλημα (Master Problem – MP), το οποίο στην περίπτωση του προβλήματος VRPTW μοντελοποιείται συνήθως σαν πρόβλημα διαχωρισμού του συνόλου των πελατών σε υποσύνολα, το καθένα από τα οποία εξυπηρετείται από ένα όχημα
- Το υπο-πρόβλημα (Sub-Problem - SP) που στην περίπτωση του προβλήματος VRPTW είναι ένα πρόβλημα ελάχιστης διαδρομής με παράθυρα χρόνου και χωρητικότητα (Shortest Path Problem with Time Windows and Capacity Constraints – SPPTWCC) και βελτιστοποιεί τη σειρά με την οποία εξυπηρετούνται οι πελάτες του κάθε υποσυνόλου

2.4.2 Ευρετικές Μέθοδοι

Ορίζεται ως ευρετική μέθοδος μια λογική ακολουθία βημάτων, η οποία δε δίνει απαραίτητα τη βέλτιστη λύση, αλλά μια αρκετά καλή λύση ώστε να μπορεί να χρησιμοποιηθεί στην πράξη. Οι ευρετικές μέθοδοι βρίσκουν λύση στο πρόβλημα δρομολόγησης οχημάτων που κατά κανόνα είναι, όσο αφορά την συνολική απόσταση που διανύουν τα οχήματα, 5-25% χειρότερες από τη βέλτιστη. Το μειονέκτημα των ευρετικών μεθόδων είναι ότι σπάνια βρίσκουν λύση που βρίσκονται πλησίον της βέλτιστης κάτω από 5% ενώ το πλεονέκτημά τους είναι η απλότητά και ο μικρός χρόνος εκτέλεσής τους.

Στο πρόβλημα δρομολόγησης οχημάτων οι ευρετικές μέθοδοι χωρίζονται σε μεθόδους που κατασκευάζουν λύσεις (Route Construction Heuristics) και σε μεθόδους που βελτιώνουν ήδη υπάρχουσες λύσεις του προβλήματος και αποκαλούνται Ευρετικές Μέθοδοι Βελτίωσης Διαδρομών (Route Improvement Heuristics) ή Ευρετικές Μέθοδοι Τοπικής αναζήτησης (Local Search Methods).

2.4.2.1 Κατασκευαστικές Ευρετικές Μέθοδοι

Οι Κατασκευαστικές Μέθοδοι για τα προβλήματα VRP, παίρνουν σαν είσοδο ένα σύνολο μη δρομολογημένων οχημάτων και ένα σύνολο μη δρομολογημένων πελατών και δημιουργούν μία καινούργια λύση που ικανοποιεί τους περιορισμούς του προβλήματος. Χωρίζονται σε δύο κατηγορίες, ανάλογα με τον τρόπο κατασκευής της λύσης:

1. Στην πρώτη κατηγορία ανήκουν οι μέθοδοι που κατασκευάζουν κάθε διαδρομή οχήματος ξεχωριστά - σειριακά (route by route). Αυτές ξεκινούν από μία άδεια διαδρομή και προσθέτουν σε αυτήν πελάτες μέχρι να ολοκληρωθεί, δηλαδή να μη μπορεί να προστεθεί άλλος πελάτης λόγω κάποιου περιορισμού (στο παρόν πρόβλημα είτε λόγο περιορισμού χωρητικότητας οχήματος, είτε λόγω περιορισμών χρονικών παραθύρων). Αφού ολοκληρωθεί η πρώτη διαδρομή δημιουργούν δεύτερη

κ.ο.κ. μέχρι να φτάσουν οι διαδρομές τον αριθμό των διαθέσιμων οχημάτων, ή μέχρι να εξυπηρετηθούν όλοι οι πελάτες.

2. Στη δεύτερη κατηγορία ανήκουν οι μέθοδοι που κατασκευάζουν παράλληλα-ταυτόχρονα τις διαδρομές των φορτηγών (parallel routing). Αυτές οι μέθοδοι ξεκινούν με τόσες διαδρομές όσες και τα οχήματα και κατανέμουν σειριακά όλους τους πελάτες σε κάποια από αυτές.

Στη συνέχεια, παρουσιάζονται οι βασικές Ευρετικές Κατασκευαστικές Μέθοδοι που έχουν προταθεί από τον Solomon (1987) καθώς, παρότι προτάθηκαν στις αρχές της επιστημονικής έρευνας για το VRP, αποτελούν τη βάση πάνω στην οποία αναπτύχθηκε πληθώρα ευρετικών κατασκευαστικών μεθόδων για το VRPTW (Bräysy & Gendreau, 2005; Foisy & Potvin, 1993; Ioannou, Kritikos, & Prastacos, 2001). Επιπλέον, παρουσιάζονται και οι μέθοδοι για το VRP από τις οποίες επηρεάστηκε ο Solomon ώστε να προτείνει τις μεθόδους για το VRPTW.

2.4.2.1.1 Μέθοδος Πλησιέστερου Γείτονα (Nearest Neighbor) για το VRP

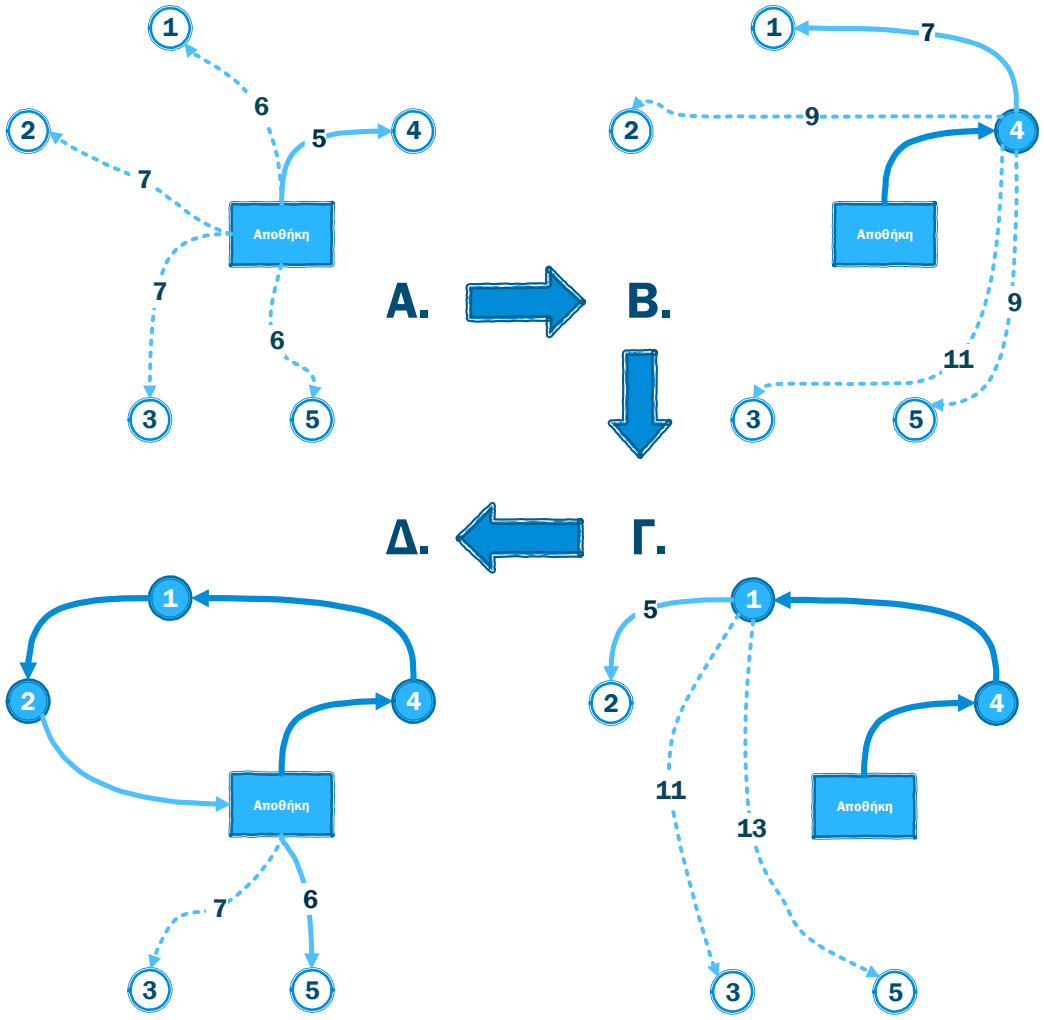
Αρχικά παρουσιάζεται ο κλασσικός αλγόριθμος «Nearest Neighbor» για το πρόβλημα VRP ο οποίος δρομολογεί κάθε όχημα σειριακά. Ξεκινά τη δρομολόγηση της κάθε διαδρομής προσθέτοντας σε αυτήν τον πελάτη που είναι κοντινότερα στην κεντρική αποθήκη. Στη συνέχεια, προσθέτει επαναληπτικά στη διαδρομή αυτή τον πελάτη που είναι κοντινότερα στον τελευταίο πελάτη που προστέθηκε στη διαδρομή, μέχρι να μη δύναται να προστεθεί άλλος πελάτης στη δεδομένη διαδρομή, όποτε ο αλγόριθμος ξεκινά τη δρομολόγηση της επόμενης διαδρομής μέχρι να δρομολογηθούν όλοι οι πελάτες.

Παρουσιάζεται ο αρχικός αλγόριθμος «Nearest Neighbor» :

1. Όσο υπάρχουν μη δρομολογημένοι πελάτες
    a. Δημιούργησε μία νέα διαδρομή
    b. Θέσε την κεντρική αποθήκη ως τη τρέχουσα θέση του οχήματος της διαδρομής
    c. Αν δεν υπάρχει μη δρομολογημένος πελάτης που να μπορεί να προστεθεί στη διαδρομή χωρίς να παραβιάζεται ο περιορισμός της χωρητικότητας του οχήματος πήγαινε στο βήμα <g>
    d. Πρόσθεσε στο τέλος της τρέχουσας διαδρομής τον κοντινότερο από την τρέχουσα θέση του οχήματος, μη δρομολογημένο πελάτη
    e. Θέσε τον πελάτη που προστέθηκε ως την τρέχουσα θέση του οχήματος
    f. Πήγαινε στο βήμα <c>
    g. Ολοκλήρωση της δρομολόγησης της τρέχουσας διαδρομής

Αλγόριθμος 2.1 : Αλγόριθμος Nearest Neighbor

Για την καλύτερη κατανόηση παρουσιάζεται παράδειγμα των πρώτων βημάτων του αλγορίθμου «Nearest Neighbor» για ένα VRP πέντε πελατών.



Σχήμα 2.7 : Παράδειγμα Αλγορίθμου Nearest Neighbor

Στην εικόνα Α. του

Σχήμα 2.7 η τρέχουσα θέση του πρώτου οχήματος προς δρομολόγηση είναι η αποθήκη, εξετάζεται η απόσταση μεταξύ της αποθήκης και του κάθε μη δρομολογημένου πελάτη και επιλέγεται ο πελάτης 4 που απέχει λιγότερο από την αποθήκη (Αλγόριθμος 2.1 Βήματα a μέχρι e). Στη συνέχεια στην εικόνα Β. και Γ. ανανεώνεται η τρέχουσα θέση του οχήματος, και επαναλαμβάνονται τα βήματα a μέχρι f του Αλγόριθμος 2.1, ενώ στην εικόνα Δ. θεωρείται ότι οι πελάτες 3 και 5 δεν μπορούν να προστεθούν στη διαδρομή του πρώτου οχήματος, έτσι αυτό επιστρέφει στην αποθήκη και ξεκινά η δρομολόγηση του δεύτερου οχήματος.

2.4.2.1.2 Μέθοδος Πλησιέστερου Γείτονα (Nearest Neighbor) για το VRPTW

Η ευρετική κατασκευαστική μέθοδος του πλησιέστερου γείτονα για το πρόβλημα VRPTW, η οποία είναι προσαρμογή της κλασσικής μεθόδου «Nearest Neighbor» (NN) για το VRP, προτάθηκε από τον Solomon και τους συνεργάτες του (1987) και διαφέρει από τον NN σε δύο σημεία:

A. Ελέγχει τους περιορισμούς των χρονικών παραθύρων παράδοσης των πελατών
B. Εισάγει την έννοια του χρόνου στον υπολογισμό της απόστασης: για την επιλογή του επόμενου πελάτη προς δρομολόγηση (Αλγόριθμος 2.1 : Αλγόριθμος Nearest Neighbor
C. – Βήμα a-d), εκτός της γεωγραφικής απόστασης ελέγχεται, το χρονικό διάστημα ανάμεσα στο τέλος εξυπηρέτησης του τελευταίου πελάτη της τρέχουσας διαδρομής και την αρχή της εξυπηρέτησης του πελάτη προς δρομολόγηση και ο «επείγοντας χαρακτήρας» (urgency) της εξυπηρέτησης του κάθε πελάτη προς δρομολόγηση

Συγκεκριμένα, ο Solomon ανέπτυξε έναν δείκτη κόστους c_{ij} της προσθήκης ενός πελάτη i στο τέλος μιας διαδρομής που εξυπηρετεί τελευταίο έναν πελάτη j:

$$c_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 v_{ij} \quad \textbf{(2.1)}$$

Όπου

$$T_{ij} = b_j - (b_i + s_i)$$

$$v_{ij} = l_j - (b_i + s_i + t_{ij}) \quad \textbf{(2.2)}$$

$$\delta_1 + \delta_2 + \delta_3 = 1,\ \delta_1 \geq 0, \delta_2 \geq 0, \delta_3 \geq 0$$

c_{ij} – Κόστος ένωσης του πελάτη i με τον πελάτη j

d_{ij} – παράγοντας απόστασης ίσος με την γεωγραφική απόσταση από τον πελάτη i στον πελάτη j

T_{ij} – Παράγοντας του χρόνου ίσος με το χρονικό διάστημα ανάμεσα στο τέλος εξυπηρέτησης του τελευταίου πελάτη της τρέχουσας διαδρομής (i) και την αρχή της εξυπηρέτησης του πελάτη προς δρομολόγηση (j)

v_{ij} – παράγοντας «επείγοντος χαρακτήρα» ίσος με το χρονικό περιθώριο που απομένει από τη χρονική στιγμή εξυπηρέτησης του πελάτη j, αν τελικά δρομολογηθεί μετά τον πελάτη i, μέχρι το πέρας του χρονικού παραθύρου εξυπηρέτησης

b_j – η χρονική στιγμή εξυπηρέτησης του πελάτη j

b_i – η χρονική στιγμή εξυπηρέτησης του πελάτη i

s_i – ο απαιτούμενος χρόνος εξυπηρέτησης του πελάτη i

s_j – ο απαιτούμενος χρόνος εξυπηρέτησης του πελάτη j

l_j – η αργότερη δυνατή στιγμή έναρξης της εξυπηρέτησης (latest arrival), δηλαδή η χρονική στιγμή που λήγει το παράθυρο χρόνου του πελάτη j

$\delta_1, \delta_2, \delta_3$ – τα βάρη του Κόστους ένωσης των πελατών i και j για τον παράγοντα απόστασης, χρόνου και «επείγοντος χαρακτήρα» αντίστοιχα

Παρουσιάζεται ο ψευδοκώδικας της προσαρμογής του αλγορίθμου NN στο VRPTW από τον Solomon:

1. Όσο υπάρχουν μη δρομολογημένοι πελάτες
    a. Δημιούργησε μία νέα διαδρομή
    b. Θέσε την κεντρική αποθήκη ως τη θέση του οχήματος της διαδρομής
    c. Αν δεν υπάρχει μη δρομολογημένος πελάτης που να μπορεί να προστεθεί στη διαδρομή χωρίς να παραβιάζεται ο περιορισμός της χωρητικότητας του οχήματος και των χρονικών παραθύρων εξυπηρέτησης πήγαινε στο βήμα <g>
    d. Πρόσθεσε στο τέλος της τρέχουσας διαδρομής τον πελάτη με το μικρότερο κόστος ένωσης από τον τελευταίο πελάτη της τρέχουσας διαδρομή που δεν έχει δρομολογηθεί
    e. Θέσε τον πελάτη που προστέθηκε ως την τρέχουσα θέση του οχήματος
    f. Πήγαινε στο βήμα <c>
    g. Ολοκλήρωση της δρομολόγησης της τρέχουσας διαδρομής

Αλγόριθμος 2.2 : Αλγόριθμος Nearest Neighbor για το VRPTW (1987)

2.4.2.1.3 Μέθοδος Savings για το VRP

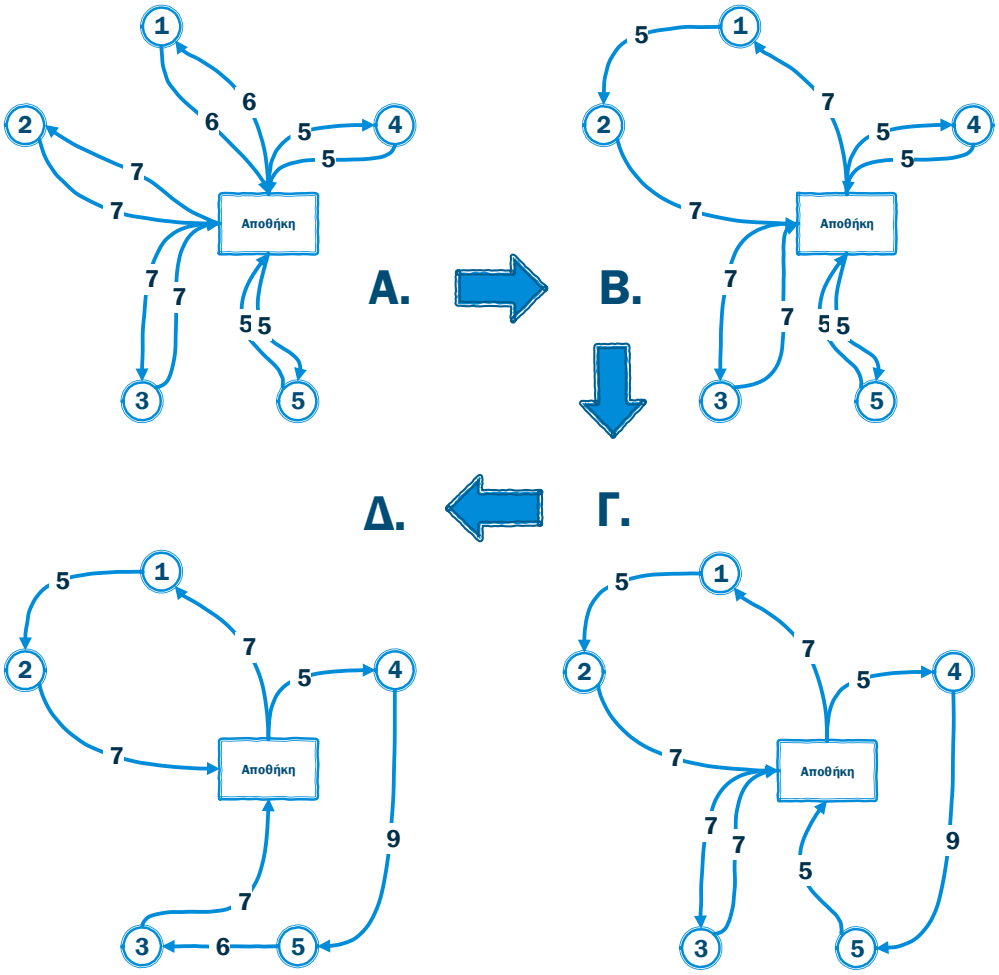
Η ευρετική κατασκευαστική μέθοδος «Savings» για το VRP προτάθηκε από τους Clarke and Wright (1964) και ανήκει στην κατηγορία «Parallel Routing». Ξεκινά κατασκευάζοντας μια λύση στην οποία υπάρχουν τόσες διαδρομές-οχήματα όσοι και οι πελάτες και το κάθε όχημα εξυπηρετεί έναν και μόνο πελάτη. Στη συνέχεια υπολογίζεται η μείωση κόστους που προκύπτει από τη συνένωση δύο διαδρομών ως εξής: αν οι δύο διαδρομές εξυπηρετούν τους πελάτες i και j, δηλαδή είναι οι $R_i : 0 - i - 0$ και $R_i : 0 - j - 0$, και αν η απόσταση μεταξύ δύο οποιονδήποτε σημείων x και y είναι d_{xy}, τότε το κόστος των δύο διαδρομών είναι $C_{Ri} = d_{0i} + d_{i0}$ και $C_{Rj} = d_{0j} + d_{j0}$. Το κόστος της διαδρομής που θα προκύψει από τη συννένωσή τους είναι $C_{Rij} = d_{0i} + d_{ij} + d_{j0}$ και η μείωση του κόστους από τη συνένωση των δύο διαδρομών θα είναι $S_{ij} = d_{0i} + d_{i0} + d_{0j} + d_{j0} - d_{0i} - d_{ij} - d_{j0} \Rightarrow S_{ij} = d_{0i} + d_{0j} - d_{ij}$. Οι Clarke and Wright (1964) επιλέγουν το τόξο (i, j) που συνδέει τους πελάτες i και j με το μεγαλύτερο S_{ij}, που να ικανοποιεί τους περιορισμούς, δηλαδή να είναι εφικτή η ένωση των δύο διαδρομών. Έτσι, οι διαδρομές συνενώνονται διαδοχικά, μέχρις ότου να φτάσουν τον αριθμό των διαθέσιμων οχημάτων (Bräysy & Gendreau, 2005).

Παρουσιάζεται η μέθοδος των Clarke and Wright (1964) σε ψευδοκώδικα

1) Δημιουργία μιας διαδρομής για κάθε ένα πελάτη
2) Όσο τα κριτήρια τερματισμού δεν ισχύουν
   a) Υπολογισμός της μείωσης κόστους από τη συνένωση κάθε χωρητικά εφικτού ζεύγους διαδρομών
   b) Συνένωση των διαδρομών με τη μεγαλύτερη μείωση κόστους συνένωσης

Αλγόριθμος 2.3 : Αλγόριθμος "Savings"

Για την καλύτερη κατανόηση παρουσιάζεται σχηματική απεικόνιση ενός παραδείγματος των πρώτων τριών βημάτων του αλγορίθμου «Savings» για ένα VRP πέντε πελατών.



Σχήμα 2.8 : Παράδειγμα Αλγορίθμου "Savings"

Στο Α. Παρουσιάζεται το βήμα 1 του Αλγόριθμος 2.3, όπου δημιουργούνται τόσες διαδρομές όσες και οι πελάτες, ενώ στα Β, Γ, Δ παρουσιάζονται τρεις επαναλήψεις των βημάτων 2.a και 2.b κατά τα οποία εκτελείται συνένωση των διαδρομών:

2.4.2.1.4 Αλγόριθμος Savings για το VRPTW

Η μέθοδος Savings των Clarke and Wright (1964), προσαρμόστηκε στο VRPTW από τον Solomon (1987) και είναι από τις καλύτερες κατασκευαστικές μεθόδους μέχρι στιγμής. Συγκεκριμένα στη μέθοδο που πρότεινε ο Solomon, πρώτον, σε κάθε βήμα της μεθόδου Savings ελέγχεται αν ικανοποιούνται οι περιορισμοί που αφορούν τα προκαθορισμένα παράθυρα χρόνου των πελατών. Δεύτερον, καθώς η αρχική μέθοδος Savings των Clarke και Wright θεωρεί συμφέρον να χρησιμοποιηθεί ένα τόξο που ενώνει δύο πελάτες που βρίσκονται γεωγραφικά κοντά, όμως έχουν μεγάλη διαφορά στα προκαθορισμένα χρονικά παράθυρα. Συνεπώς, προκύπτουν αυξημένοι χρόνοι αναμονής των οχημάτων, και δημιουργείται μεγάλο κόστος ευκαιρίας, καθώς τα οχήματα θα μπορούσαν να εξυπηρετούν περισσότερους πελάτες. Συνεπώς, ο Solomon προτείνει τον περιορισμό της αύξησης χρόνου αναμονής για τη χρήση ενός τόξου και την απόρριψη συνένωσης διαδρομών που υπερβαίνουν ένα προκαθορισμένο όριο ακόμη και στην περίπτωση που η συνένωση αυτή έχει τη μεγαλύτερη μείωση κόστους. Ακολουθεί ψευδοκώδικας με τη μέθοδο «Savings» του Solomon (1987)

1) Δημιουργία μιας διαδρομής για κάθε ένα πελάτη
2) Όσο τα κριτήρια τερματισμού δεν ισχύουν
   c) Υπολογισμός της μείωσης κόστους και της αύξησης χρόνου αναμονής από τη συνένωση κάθε χρονικά και χωρητικά εφικτού ζεύγους διαδρομών
   d) Συνένωση του ζεύγους διαδρομών με τη μεγαλύτερη μείωση κόστους συνένωσης με την προϋπόθεση η αύξηση χρόνου αναμονής του να μην υπερβαίνει το προκαθορισμένο όριο

Αλγόριθμος 2.4 : Αλγόριθμος "Savings" για το VRPTW (Marius M Solomon, 1987)

2.4.2.2 Μέθοδοι Τοπικής αναζήτησης

Οι μέθοδοι τοπικής αναζήτησης είναι μία γενική κατηγορία προσεγγιστικών αλγορίθμων οι οποίες επαναληπτικά βελτιώνουν μία αρχική εφικτή λύση του προβλήματος διερευνώντας τις γειτονικές σε αυτή λύσεις, δηλαδή λύσεις που προκύπτουν πραγματοποιώντας μικρό αριθμό «κινήσεων» πχ. εναλλαγή της σειράς δύο πελατών σε κάποια διαδρομή, ή μετακίνηση ενός πελάτη σε διαφορετική διαδρομή. Συνεπώς, κάθε μέθοδος τοπικής αναζήτησης διέπεται από τα εξής χαρακτηριστικά (Bräysy & Gendreau, 2005)

- Τον μηχανισμό δημιουργίας γειτονικών εφικτών λύσεων: ακολουθούνται δύο τακτικές για το VRP
  - Οι μέθοδοι (Intra-Route) τοπικής αναζήτησης ερευνούν γειτονικές λύσεις της αρχικής, πραγματοποιώντας μετατροπές που επηρεάζουν μόνο τη σειρά με την οποία το κάθε όχημα επισκέπτεται τους πελάτες προς εξυπηρέτηση
  - Οι μέθοδοι (Inter-Route) τοπικής αναζήτησης ερευνούν γειτονικές στην αρχική λύσεις πραγματοποιώντας μετατροπές που μπορεί να

επηρεάσουν εκτός της σειράς με την οποία θα εξυπηρετηθούν οι πελάτες και το όχημα από το οποίο θα εξυπηρετηθούν

- Το κριτήριο αποδοχής γειτονικής λύσης: ακολουθούνται δύο τακτικές
  - Η τακτική αποδοχής του πρώτου (First-Accept – FA) στο οποίο η πρώτη λύση που υπολογίζεται ότι ικανοποιεί το κριτήριο αποδοχής γειτονικής λύσης γίνεται αποδεκτή
  - Η τακτική αποδοχής του καλύτερου (Best-Accept – BA) στο οποίο υπολογίζεται το κριτήριο γειτονικής λύσης για ολόκληρη τη γειτονιά και γίνεται αποδεκτή η καλύτερη λύση
- Το κριτήριο τερματισμού της μεθόδου: συνηθίζεται το κριτήριο τερματισμού να είναι προκαθορισμένος αριθμός επαναλήψεων ή χρόνου εκτέλεσης της μεθόδου

Η τοπικά βέλτιστη λύση που παράγεται από οποιαδήποτε μέθοδο τοπικής αναζήτησης δύναται να διαφέρει σημαντικά από την ολικά βέλτιστη λύση του εκάστοτε προβλήματος, καθώς οι μέθοδοι τοπικής αναζήτησης πραγματοποιούν μετατροπές στην αρχική λύση μόνο στην περίπτωση που αυτές προκαλούν βελτίωση της ποιότητας της λύσης. Συνεπώς, η ποιότητα των λύσεων των μεθόδων τοπικής αναζήτησης εξαρτάται από την ποιότητα της αρχικής λύσης και στο μηχανισμό δημιουργίας γειτονικών λύσεων.

2.4.2.2.1 Μέθοδος 2-opt

Η μέθοδος 2-opt θεωρείται μέθοδος τοπικής αναζήτησης ενδο-διαδρομής και βελτιώνει συγκεκριμένη διαδρομή οχήματος την οποία δέχεται ως είσοδο, επηρεάζοντας τη σειρά με την οποία εξυπηρετούνται οι πελάτες και όχι το σύνολο των πελατών που εξυπηρετούνται από τη δεδομένη διαδρομή.

Συγκεκριμένα παράγει την γειτονιά της κάθε διαδρομής εναλλάσσοντας για κάθε πελάτη της διαδρομής τη θέση του με τον επόμενο από αυτόν πελάτη στη σειρά προτεραιότητας που εξυπηρετούνται από το όχημα.

Παρουσιάζεται ένα παράδειγμα για τη διαδρομή ενός οχήματος που εξυπηρετεί τέσσερις πελάτες:

Αποθήκη	Πελάτης 4	Πελάτης 7	Πελάτης 8	Πελάτης 6	Αποθήκη

Σχήμα 2.9 Μέθοδος 2-opt: Αρχική διαδρομή οχήματος

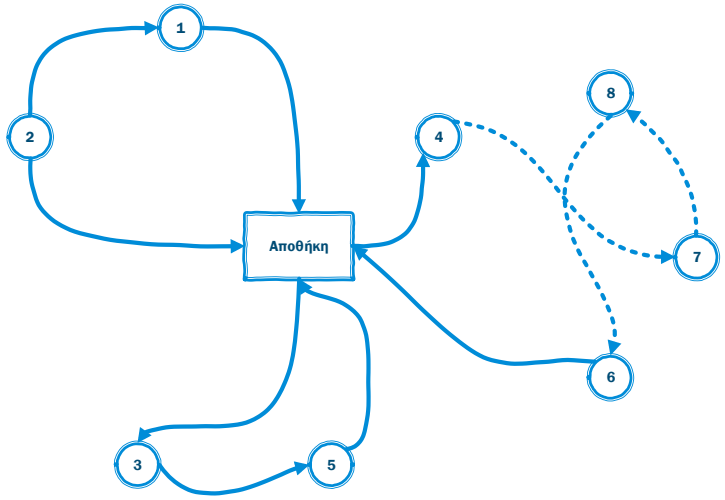
Κατά την εκτέλεση της μεθόδου τοπικής αναζήτησης 2-opt, δημιουργείται η γειτονιά της αρχικής διαδρομής του οχήματος (Σχήμα 2.9) εναλλάσσοντας τη σειρά που εξυπηρετούνται ζευγάρια πελατών, τα οποία περιλαμβάνουν πελάτες που εξυπηρετούνται διαδοχικά. Συνεπώς, στο εν λόγω παράδειγμα δημιουργούνται τρεις γειτονικές διαδρομές (Σχήμα 2.10) εναλλάσσοντας τον πρώτο με τον δεύτερο, τον δεύτερο με τον τρίτο και τον τρίτο με τον τέταρτο πελάτη αντίστοιχα.

Αποθήκη	Πελάτης 7	Πελάτης 4	Πελάτης 8	Πελάτης 6	Αποθήκη
Αποθήκη	Πελάτης 4	Πελάτης 8	Πελάτης 7	Πελάτης 6	Αποθήκη
Αποθήκη	Πελάτης 4	Πελάτης 7	Πελάτης 6	Πελάτης 8	Αποθήκη

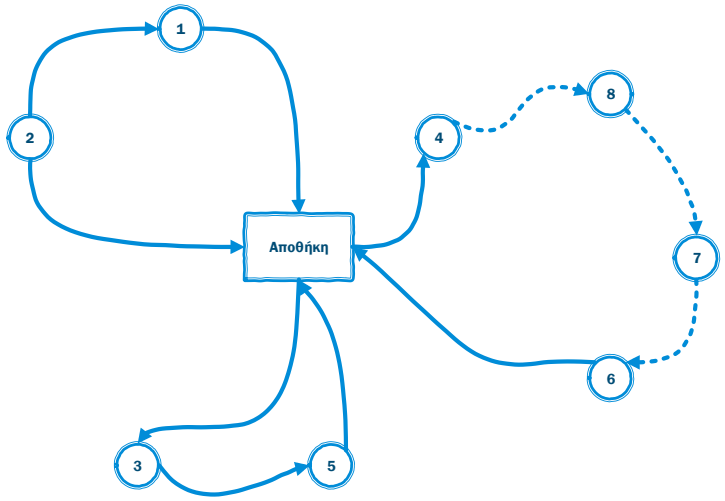
Σχήμα 2.10 Μέθοδος 2-opt: Η γειτονιά της αρχικής διαδρομής

Στο παράδειγμα αυτό θεωρείται ότι η δεύτερη διαδρομή βελτιώνει το κόστος της συνολικής διαδρομής σε σχέση με την αρχική διαδρομή, ενώ η πρώτη και η τρίτη διαδρομή είτε δεν βελτιώνουν τη λύση, είτε δεν είναι εφικτές, είτε βελτιώνουν το κόστος της λύσης λιγότερο από την δεύτερη διαδρομή.

Παρουσιάζεται γραφική απεικόνιση της αρχικής διαδρομής πριν την τοπική αναζήτηση και της τελικής διαδρομής του οχήματος μετά την τοπική αναζήτηση.



A.



B.

Σχήμα 2.11 : Η μέθοδος τοπικής αναζήτησης 2-opt

2.4.3 Μεταευρετικές Μέθοδοι

Οι μεταευρετικές μέθοδοι, ή γενικές ευρετικές μέθοδοι ή γενικές μέθοδοι τοπικής αναζήτησης έχουν γνωρίσει μεγάλη ανάπτυξη τα τελευταία δεκαπέντε χρόνια. Ανάμεσα στις πιο επιτυχημένες τεχνικές είναι οι Γενετικοί αλγόριθμοι (Genetic Algorithms), ο αλγόριθμος Προσομοίωσης Ανόπτυσης (Simulated Annealing), ο αλγόριθμος Αναζήτησης Taboo (Taboo

Search) και φυσικά οι αλγόριθμοι βελτιστοποίησης Σμήνους Σωματιδίων (Particle Swarm Optimization) και βελτιστοποίησης Αποικίας Μυρμηγκιών (Ant Colony Optimization). Η επιτυχία της κάθε μεθόδου εξαρτάται από πολλούς παράγοντες, όπως η ευκολία υλοποίησής τους, η ικανότητά τους να συμπεριλάβουν συγκεκριμένους περιορισμούς που εμφανίζονται σε πραγματικές εφαρμογές, ο χρόνος εκτέλεσης που απαιτούν και η ποιότητα της λύσης που τελικά δίνουν.

Πάραυτα, από θεωρητική άποψης, η χρήση αυτών των μεθόδων δεν έχει αιτιολογηθεί επαρκώς, αφού δεν υπάρχει απόδειξη για την σίγουρη σύγκλισή τους. Για παράδειγμα, υπάρχουν κάποια θεωρήματα σύγκλισης για τον αλγόριθμο Προσομοίωσης Ανόπτησης και τον αλγόριθμο Αναζήτησης Taboo (Aarts & Laarhoven, 1985; Faigle & Kern, 1992; Hajek, 1988), τα οποία όμως δεν μπορούν να χρησιμοποιηθούν στην πράξη. Αυτά αποδεικνύουν πως η αναζήτηση μέσω των παραπάνω μεθόδων έχει μια πολύ μεγάλη πιθανότητα να καταλήξει σε βέλτιστη λύση, αν της δοθεί αρκετά μεγάλος χρόνος εκτέλεσης, όμως ο χρόνος αυτός είναι μεγαλύτερος από τον απαιτούμενο χρόνο για την αναζήτηση σε ολόκληρο το πεδίο λύσεων (Aarts & Laarhoven, 1985).

Παρόλα αυτά, οι μεταευρετικές μέθοδοι είναι πολύ ανταγωνιστικές σε πρακτικές εφαρμογές και μάλιστα, οι πιο ανταγωνιστικές τεχνικές συνδυάζουν δύο ή περισσότερες μεταευρετικές μεθόδους. Το αποτέλεσμα είναι παρόμοιες τεχνικές επίλυσης προβλημάτων να είναι γνωστές κάτω από πολύ διαφορετικά ονόματα όπως Γενετικά Υβρίδια (Genetic Hybrids), Αναζήτηση Taboo με πιθανότητες (Probabilistic Taboo Search), Προσομοίωση Αποικίας Μυρμηγκιών Μέγιστου-Ελάχιστου (MAX-MIN Ant System). Όλες αυτές οι τεχνικές έχουν τρία κοινά στοιχεία: πρώτον, έχουν κάποιο μηχανισμό μνήμης για τις λύσεις που παράγονται κατά το τρέξιμό τους, δεύτερον περιλαμβάνουν μια διαδικασία που κατασκευάζει καινούργιες λύσεις χρησιμοποιώντας τις πληροφορίες που έχουν στη μνήμη τους και τρίτον χρησιμοποιούν μεθόδους τοπικής αναζήτησης για να βελτιώσουν τις αρχικές τους λύσεις. Παρατηρώντας τις ομοιότητες μεταξύ αυτών των μεθόδων, έχει προταθεί από τους Taillard και τους συνεργάτες του (Taillard, Gambardella, Gendreau, & Potvin, 2001) το όνομα Προγραμματισμός Προσαρμοζόμενης Μνήμης (Adaptive Memory Programming) για όλες τις παραπάνω τεχνικές.

Οι μεταευρετικές μέθοδοι βελτιστοποίησης ξεκινούν συνήθως από ένα σύνολο εφικτών λύσεων και πραγματοποιούν χειρισμούς που βελτιώνουν τις αρχικές αυτές λύσεις. Οι χειρισμοί αυτοί επαναλαμβάνονται μέχρι να βρεθεί μία αρκετά καλή λύση ή για συγκεκριμένο αριθμό επαναλήψεων.

Το πρώτο μειονέκτημα των μεταευρετικών μεθόδων, όπως και των ευρετικών, είναι η εγκυρότητά τους, που υπολογίζεται με τη λύση προβλημάτων αναφοράς, τα οποία έχουν γνωστές βέλτιστες λύσεις. Ένα δεύτερο μειονέκτημα είναι ότι δεν δύναται να εγγυηθεί κανείς εκ των προτέρων ποιά από τις μεταευρετικές μεθόδους ή ποιός συνδυασμός τους θα δώσει τα καλύτερα αποτελέσματα σε ένα δεδομένο πρόβλημα. Παρόλα αυτά, οι μεταευρετικές μέθοδοι χρησιμοποιούνται ευρέως στην πράξη για την αντιμετώπιση σύνθετων και συνδυαστικών προβλημάτων, όπως είναι το VRPTW και κυρίως στην περίπτωση μεγάλου πλήθους πελατών και όταν απαιτείται μικρός χρόνος εκτέλεσης.

2.4.3.1 Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι προσομοιώνουν την εξελικτική διαδικασία που χρησιμοποιούν τα είδη που αναπαράγονται αμφιγονικά. Αυτή η εξελικτική διαδικασία είναι η εξής: κατά την αναπαραγωγή, ένα νέο άτομο, διαφορετικό από τους γονείς του, δημιουργείται, από την πράξη δύο θεμελιωδών μηχανισμών. Ο πρώτος είναι η «διασταύρωση χρωμοσωμάτων» που συνδυάζει το μισό γενετικό υλικό του κάθε γονιού για να παράγει το γενετικό υλικό του νέου ατόμου. Ο δεύτερος μηχανισμός είναι η «μετάλλαξη» με τον οποίο γίνεται μία «αυθόρμητη» αλλαγή του γενετικού υλικού. Το νέο άτομο που λοιπόν που θα δημιουργηθεί, το οποίο ονομάζεται «παιδί» ή «απόγονος», είναι διαφορετικό από τους γονείς του, όμως έχει έναν αριθμό χαρακτηριστικών τους. Αν το παιδί κληρονομήσει «καλά» χαρακτηριστικά από τους γονείς του, η πιθανότητα να επιβιώσει είναι μεγαλύτερη, αντιθέτως με τα παιδιά που κληρονομούν «άσχημα» χαρακτηριστικά. Έτσι, το παιδί με τα «καλά» χαρακτηριστικά έχει μεγαλύτερη πιθανότητα να αναπαραχθεί και να δώσει αυτά τα χαρακτηριστικά στους απογόνους του. Η αναλογία μεταξύ αυτής της εξελικτικής διαδικασίας και του Μεταευρετικού Γενετικού Αλγορίθμου που ξεκίνησε από τον (Holland, 1975) μπορεί να οριστεί με τον παρακάτω τρόπο:

Το «άτομο» μπορεί να ταυτιστεί με μια εφικτή λύση στο προς επίλυση πρόβλημα. Η λύση κωδικοποιείται σε ένα δυαδικό διάνυσμα και η «διασταύρωση» των γονιών πραγματοποιείται κρατώντας ένα κομμάτι από κάθε γονέα, δημιουργώντας έτσι έναν απόγονο. Ακολουθεί η «μετάλλαξη» του απογόνου η οποία συμβαίνει με μία μικρή πιθανότητα και εάν συμβεί, εναλλάσσει δύο ψηφία του δυαδικού διανύσματος του απογόνου. Τέλος, εξετάζεται η ποιότητα του απογόνου εφαρμόζοντας το δυαδικό διάνυσμά του στην αντικειμενική συνάρτηση της λύσης του προβλήματος.

Ακολουθεί ψευδοκώδικας ενός τυπικού γενετικού αλγορίθμου

```
1. Δημιουργία ενός πληθυσμού δυαδικών διανυσμάτων (άτομα)
2. Όσο τα κριτήρια τερματισμού δεν ισχύουν
    a. Επέλεξε μία ομάδα γονιών από τον πληθυσμό
    b. Διασταύρωσε τυχαία την ομάδα γονιών
    c. Μετάλλαξε ένα ποσοστό των παιδιών
    d. Εξέτασε την ποιότητα των παιδιών
    e. Εισήγαγε τα παιδιά στον αρχικό πληθυσμό
    f. Αφαίρεσε τα «χειρότερα» άτομα από τον πληθυσμό
```

Αλγόριθμος 2.5 : Ψευδοκώδικας γενετικού αλγορίθμου

Τυπικά, η παραπάνω διαδικασία σταματάει μετά από ένα συγκεκριμένο αριθμό επαναλήψεων (γενεές) ή όταν η καλύτερη λύση που έχει βρεθεί δε βελτιωθεί για κάποιον προκαθορισμένο αριθμό γενεών. Είναι σημαντικό να σημειωθεί πως η επιλογή των γονιών είναι τυχαία, πάραυτα ευνοούνται τα «καλά» άτομα. Πρόσφατες παραλλαγές του Γενετικού Αλγόριθμου δεν χρησιμοποιούν δυαδικά διανύσματα για την κωδικοποίηση της λύσης του προβλήματος που λύνουν. Συγκεκριμένα για το πρόβλημα του περιοδεύοντος πωλητή και το πρόβλημα δρομολόγησης οχημάτων χρησιμοποιούνται διανύσματα ακεραίων αριθμών για

την κωδικοποίηση των λύσεων γεγονός που επηρεάζει άμεσα τους μηχανισμούς «διασταύρωσης» και «μετάλλαξης».

2.4.3.2 Αναζήτηση Taboo

Ο όρος «Αναζήτηση Taboo» (“Taboo Search”) προτάθηκε για πρώτη φορά από τον Glover (1986). Η βασική ιδέα τη μεθόδου είναι να τροποποιεί τοπικά και επαναλαμβανόμενα τη λύση, ενώ ταυτόχρονα κρατάει σε μνήμη αυτές τις μετατροπές ώστε να αποφευχθεί η επιστροφή σε μία λύση που έχει εξετασθεί. Συγκεκριμένα, οι μετατροπές που δέχεται η λύση διατηρούνται σε μια λίστα «απαγορευμένων μετατροπών» και ο αλγόριθμος απαγορεύει τη χρήση τους για ένα συγκεκριμένο αριθμό επαναλήψεων. Η πιο σημαντική ιδέα πάνω στην Αναζήτηση Taboo είναι η χρήση μια βραχυπρόθεσμης μνήμης, γνωστή ως «Λίστα Taboo». Ακολουθεί ο αλγόριθμος μιας βασική μορφή της Αναζήτησης Taboo

1. Κατασκεύασε μια αρχική λύση και αρχικοποίησε τη μνήμη
2. Όσο τα κριτήρια τερματισμού δεν ισχύουν
    a. Επέλεξε μία γειτονική λύση της αρχικής χωρίς να κάνεις κινήσεις που βρίσκονται στην «Λίστα Taboo»
    b. Αν η γειτονική λύση είναι καλύτερη της αρχικής, θέσε αυτήν σαν αρχική λύση
    c. Ανανέωσε τη μνήμη

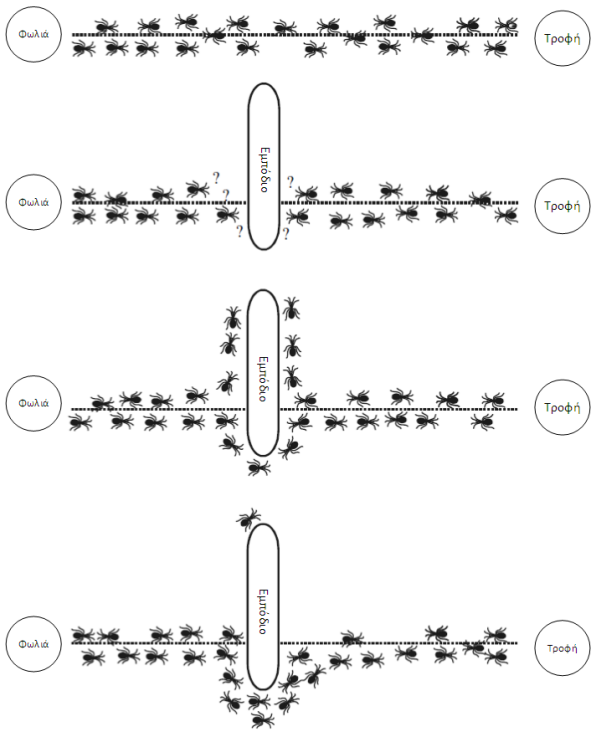
Αλγόριθμος 2.6 : Αναζήτηση Taboo

Τυπικά, το κριτήριο τερματισμού αντιστοιχεί σε έναν συγκεκριμένο αριθμό επαναλήψεων ή σε ένα συγκεκριμένο αριθμό συνεχόμενων επαναλήψεων χωρίς να βελτιωθεί η καλύτερη λύση που έχει βρεθεί από τον αλγόριθμο. Κάθε βήμα του παραπάνω αλγόριθμου μπορεί να είναι πολύ απλό ή και πολύ σύνθετο, ανάλογα την υλοποίηση. Για παράδειγμα, το βήμα (2.a) μπορεί να υλοποιηθεί εξετάζοντας όλη τη γειτονιά της αρχικής λύσης και επιλέγοντας την καλύτερη «Μη Taboo» λύση ή μπορεί να υλοποιηθεί χρησιμοποιώντας μια διαδικασία επιλογής γειτονικής λύσης πιο πολύπλοκη, για παράδειγμα χρησιμοποιώντας κάποιο διαφορετικό μεταευρετικό αλγόριθμο (Rochat & Taillard, 1995).

2.5 Η βελτιστοποίηση αποικίας μυρμηγκιών (ACO)

Η ιδέα της μίμησης της συμπεριφοράς των μυρμηγκιών για την εύρεση λύσεων στα προβλήματα βελτιστοποίησης ξεκίνησε στις αρχές της δεκαετίας του 90’, προτάθηκε υπό την ονομασία «Ant System – AS» (Colorni, Dorigo, & Maniezzo, 1991; Dorigo, Maniezzo, & Colorni, 1991; Dorigo, Maniezzo, Colorni, & Maniezzo, 1991) και χρησιμοποιήθηκε στο πρόβλημα πλανόδιου πωλητή (Traveling Salesman Problem – TSP). Η προσομοίωση προήλθε από τον τρόπο με τον οποίο τα μυρμήγκια ψάχνουν το βέλτιστο δρόμο για την τροφή τους αλλά και τον δρόμο για να επιστρέψουν στην φωλιά τους. Αρχικά τα μυρμήγκια εξερευνούν μια μεγάλη περιοχή γύρω από τη φωλιά τους κάνοντας τυχαίες διαδρομές. Όταν κάποιο μυρμήγκι βρει κάποια πηγή φαγητού, αξιολογεί την πηγή (σε ποιότητα και σε ποσότητα) και μεταφέρει ένα μέρος της τροφής στην φωλιά. Στο ταξίδι του γυρισμού προς τη φωλιά, το μυρμήγκι αφήνει στη διαδρομή που διανύει μία χημική ουσία που ονομάζεται φερομόνη. Η

πυκνότητα της φερομόνης που αφήνει το μυρμήγκι γυρίζοντας στη φωλιά του εξαρτάται από την ποιότητα της πηγής τροφής που βρήκε, ενώ ο ρόλος της φερομόνης αυτής είναι να οδηγήσει τα υπόλοιπα μυρμήγκια προς την πηγή της τροφής. Αν η ποιότητα της πηγής της τροφής που βρέθηκε από το μυρμήγκι είναι καλή, στο μονοπάτι που οδηγεί σε αυτήν θα συσσωρευτεί μεγάλη ποσότητα φερομόνης, επειδή όλο και περισσότερα μυρμήγκια θα το επιλέγουν. Επίσης, οι πηγές τροφής που βρίσκονται πιο κοντά στη φωλιά, θα έχουν περισσότερες επισκέψεις μυρμηγκιών, έτσι η φερομόνες στο μονοπάτι που οδηγεί σε αυτές θα αυξηθούν γρηγορότερα. Το τελικό αποτέλεσμα της διαδικασίας αυτής είναι ότι τα μυρμήγκια σταδιακά θα βελτιώνουν το χρόνο που χρειάζονται για να μεταφέρουν τροφή στη φωλιά τους.



Εικόνα 2.1 : Ο αλγόριθμος ACO προήλθε από τον τρόπο που τα μυρμήγκια επιλέγουν τη διαδρομή για την τροφή τους

Η τεχνική αυτή αναζήτησης φαγητού μπορεί να μεταφερθεί σε ένα αλγοριθμικό πλαίσιο για την επίλυση προβλημάτων βελτιστοποίησης, χρησιμοποιώντας την αναλογία μεταξύ:

1. Της περιοχή αναζήτησης των πραγματικών μυρμηγκιών και τον χώρο λύσεων ενός προβλήματος βελτιστοποίησης
2. Της ποιότητα της πηγής της τροφής (ποιοτικά και ποσοτικά) και της αντικειμενικής συνάρτησης ενός προβλήματος βελτιστοποίησης
3. Του ίχνους φερομόνης και της χρήσης μνήμης στην επίλυση ενός προβλήματος βελτιστοποίησης

Μια λεπτομερής περιγραφή του πως αυτές οι αναλογίες μπορούν να χρησιμοποιηθούν στην επίλυση ενός προβλήματος περιοδεύοντος πωλητή (Travelling Salesman Problem) βρίσκεται στη μελέτη του Dorigo και τους συνεργάτες του (1996).

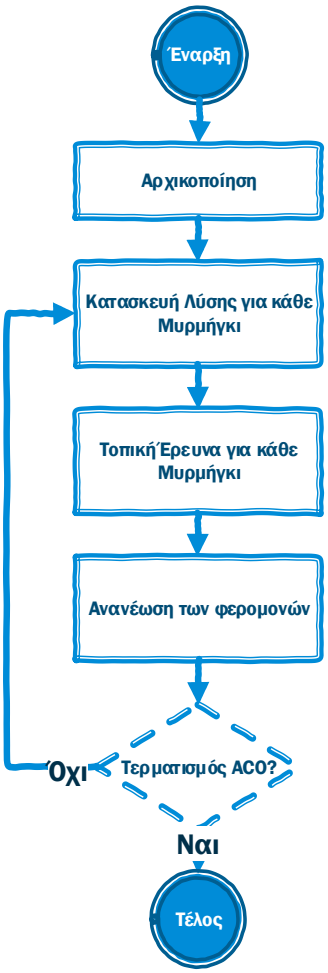
Παρουσιάζεται η βασική μορφή του αλγορίθμου προσομοίωσης αποικίας μυρμηγκιών σε ψευδοκώδικα:

```
1. Αρχικοποίηση του ίχνους φερομόνης
3. Όσο τα κριτήρια τερματισμού δεν ισχύουν
    a. Για κάθε μυρμήγκι, κατασκεύασε μια καινούργια λύση
       χρησιμοποιώντας τα ίχνη φερομόνης και αξιολόγησε αυτή
       τη λύση σύμφωνα με την αντικειμενική συνάρτηση
    b. Ανανέωσε τα ίχνη φερομόνης των διαδρομών που
       χρησιμοποίησαν τα μυρμήγκια στις λύσεις τους αναλογα
       με την αξιολόγηση της εκάστοτε λύσης
```

Αλγόριθμος 2.7 : Ψευδοκώδικας Βελτιστοποίησης Αποικίας Μυρμηγκιών

Το πιο σημαντικό στοιχείο ενός αλγορίθμου προσομοίωσης αποικίας μυρμηγκιών είναι η διαχείριση του ίχνους φερομόνης. Ο αλγόριθμος προσομοίωσης αποικίας μυρμηγκιών στην απλή του μορφή συνδυάζει τα ίχνη φερομόνης με την αντικειμενική συνάρτηση κατά την κατασκευή νέων λύσεων από κάθε μυρμήγκι. Στη συνέχεια, αφού κάθε μυρμήγκι έχει κατασκευάσει μια λύση, όλα τα ίχνη φερομόνης μειώνονται σύμφωνα με ένα συντελεστή, ώστε να προσομοιωθεί η εξάτμιση της φερομόνης. Μετέπειτα, τα ίχνη φερομόνης που αντιστοιχούν στις λύσεις που κατασκευάστηκαν από τα μυρμήγκια αυξάνονται αναλογικά με την ποιότητα της εκάστοτε λύσης. Βασισμένες πάνω στις παραπάνω θεμελιώδεις αρχές έχουν προταθεί διάφορες παραλλαγές του αλγορίθμου προσομοίωσης αποικίας μυρμηγκιών.

Παρουσιάζεται το διάγραμμα ροής του αλγορίθμου ACO:



Σχήμα 2.12 : Διάγραμμα ροής ACO

2.6 Η βελτιστοποίηση σμήνους σωματιδίων (PSO)

Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization – PSO) αναπτύχθηκε από τους Kennedy και Eberhart (Kennedy & Eberhart, 1995) και, λόγω της αποδοτικότητας, της αποτελεσματικότητας και της ευκολίας υλοποίησής του σε μεγάλη ποικιλία προβλημάτων βελτιστοποίησης, έχει μελετηθεί εκτενώς (Bratton & Kennedy, 2007; Liang, Qin, Suganthan, & Baskar, 2006; Poli, Kennedy, & Blackwell, 2007; Zhang, Yu, & Hu, 2005).

Ο αλγόριθμος PSO είναι εμπνευσμένος από τον τρόπο που πλησιάζουν την τροφή τους τα πτηνά, έτσι προσομοιώνει ένα σμήνος σωματιδίων τα οποία έχουν συγκεκριμένη θέση και ταχύτητα, ενώ τείνουν να κινούνται δυναμικά προς δύο τοποθεσίες: την ολική βέλτιστη θέση του σμήνους και τη βέλτιστη θέση από την οποία έχει περάσει το εκάστοτε σωματίδιο (Gong et al., 2012).

Συγκεκριμένα, ο PSO δέχεται ως είσοδο

- ένα πλήθος P διανυσμάτων πραγματικών αριθμών διαστάσεως N, κατασκευασμένα είτε τυχαία, είτε από διαφορετική μέθοδο
- μία μέθοδο αξιολόγησης των N – διάστατων αυτών διανυσμάτων

τα διανύσματα αναπαριστούν λύσεις στο προς επίλυση πρόβλημα και αποτελούν τα διανύσματα θέσης του αρχικού πληθυσμού σωματιδίων του PSO, ενώ η μέθοδος αξιολόγησής τους αντιπροσωπεύει την αντικειμενική συνάρτηση του προβλήματος, στην οποία, κατά την πλειονότητα των μελετών της βιβλιογραφίας, οι περιορισμοί του εκάστοτε προβλήματος έχουν εισαχθεί ως ποινές.

Εκτός του διανύσματος θέσης, το κάθε σωματίδιο, χαρακτηρίζεται από ένα διάνυσμα ταχύτητας, πλήθους διαστάσεων ίσο με το πλήθος διαστάσεων του διανύσματος θέσης, ενώ σε κάθε επανάληψη του αλγορίθμου η θέση του κάθε σωματιδίου μεταβάλλεται σύμφωνα με την ταχύτητά του. Συγκεκριμένα, αν X_i^d είναι η θέση του σωματιδίου i για την διάσταση d και V_i^d είναι η ταχύτητα του σωματιδίου i για την διάσταση d, η μεταβολή τους ορίζεται ως

$$V_i^d = wV_i^d + a_1 r_1 \left(pBest_i^d - X_i^d\right) + a_2 r_2 \left(gBest^d - X_i^d\right) \quad \textbf{(2.3)}$$

Και

$$X_i^d = X_i^d + V_i^d \quad \textbf{(2.4)}$$

Όπου

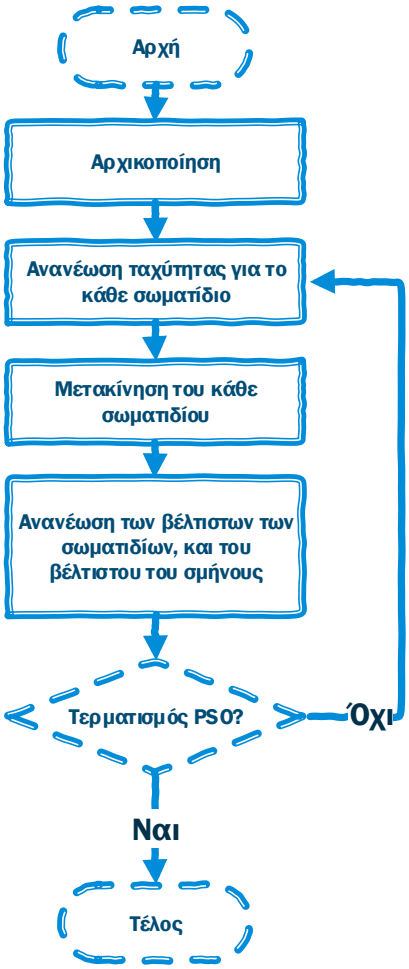
w	συντελεστής αδράνειας (inertia weight)
a_1	Συντελεστής σύγκλισης του σωματιδίου προς τη βέλτιστη θέση από την οποία έχει περάσει
a_2	Συντελεστής σύγκλισης του σωματιδίου προς τη βέλτιστη θέση ολόκληρου του σμήνους σωματιδίων
r_1, r_2	Τυχαίοι πραγματικοί αριθμοί μεταξύ του 0 και του 1
$pBest_i^d$	η διάσταση d της βέλτιστης θέσης από την οποία έχει περάσει το σωματίδιο i
$pBest_i^d$	η διάσταση d της βέλτιστης θέσης ολόκληρου του σμήνους σωματιδίων

Παρουσιάζεται ο αλγόριθμος PSO για την καλύτερη κατανόησή του

1. Αρχικοποίηση της θέσης και της ταχύτητας του σμήνους σωματιδίων, του βέλτιστου του κάθε σωματιδίου και του ολικού βέλτιστου
2. Όσο τα κριτήρια τερματισμού δεν ικανοποιούνται
    a. Ανανέωσε την ταχύτητα του κάθε σωματιδίου σύμφωνα με το βέλτιστό του και σύμφωνα με το ολικό βέλτιστο
    b. Μετακίνησε τα σωματίδια ανάλογα με την ταχύτητά τους
    c. Αξιολόγησε την ποιότητα του κάθε σωματιδίου και ανανέωσε το βέλτιστο του κάθε σωματιδίου και το ολικό βέλτιστο

Αλγόριθμος 2.8 : Αλγόριθμος PSO

Παρουσιάζεται ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων σε διάγραμμα ροής για την καλύτερη κατανόησή του:



Σχήμα 2.13 : Διάγραμμα Ροής PSO

2.6.1 Προσαρμογή του PSO σε προβλήματα ακέραιου προγραμματισμού

Παρότι ο αλγόριθμος PSO αναπτύχθηκε για συνεχή προβλήματα βελτιστοποίησης (προβλήματα βελτιστοποίησης με συνεχής μεταβλητές απόφασης), έχει προσαρμοστεί και σε ακέραια προβλήματα βελτιστοποίησης όπως είναι το VRPTW (Belmecheri, Prins, Yalaoui, & Amodeo, 2010; Pang, Wang, Zhou, & Dong, 2004; K.-P. Wang, Huang, Zhou, & Pang, 2003; Zhong, Zhang, & Chen, 2007). Για την χρήση του PSO για την επίλυση ακέραιων προβλημάτων βελτιστοποίησης απαιτείται η αναπαράστασης της λύσης του εκάστοτε προβλήματος ώστε να μπορεί να αναπαραστήσει τη θέση ενός σωματιδίου του αλγορίθμου PSO, ή η προσαρμογή του αλγορίθμου ώστε να χρησιμοποιεί ακέραια διανύσματα θέσης και ταχύτητας. Στη συνέχεια, παρουσιάζονται μερικές προσεγγίσεις χρήσης του PSO για την επίλυση προβλημάτων VRP από τη βιβλιογραφία.

2.6.1.1 Προσέγγιση των A.-l. Chen, Yang, and Wu (2006), Yang et al. (2004)

Στην προσέγγιση του Chen (2006) η λύση ενός VRP αναπαρίσταται από ένα δυαδικό διάνυσμα $N * K$ διαστάσεων, όπου N το πλήθος των πελατών και K το πλήθος των οχημάτων. Έτσι αν $D = a * N + b$, η διάσταση D είναι μηδέν αν το όχημα α δεν επισκέπτεται τον πελάτη b και άσσος αν τον επισκέπτεται. Παρουσιάζεται παράδειγμα της αναπαράστασης λύσης αυτής για την καλύτερη κατανόησή της, για ένα παράδειγμα 4 πελατών και 2 οχημάτων

1	2	3	4	5	6	7	8
0	1	0	1	1	0	1	0

Σχήμα 2.14: Αναπαράσταση Λύσης VRP του Chen (2006)

Στο Σχήμα 2.14 το όχημα 1 (διαστάσεις 1 έως 4) εξυπηρετεί τους πελάτες 2 και 4, και το όχημα 2 εξυπηρετεί τους πελάτες 1 και 3.

Για την προσαρμογή του PSO σε δυαδικό διάνυσμα θέσεων τροποποιούνται οι εξισώσεις μετακίνησης και ανανέωσης της ταχύτητας των σωματιδίων έτσι ώστε η ταχύτητα να συμβολίζει την πιθανότητα η θέση του σωματιδίου να είναι άσσος.

$$V_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\mu\acute{\eta}\nu ous} = \alpha X_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\mu\acute{\eta}\nu ous} + \beta(1 - X_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\mu\acute{\eta}\nu ous}) \quad (2.5)$$

$$V_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\omega\mu\alpha\tau\iota\delta\acute{\iota}ou} = \alpha X_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\omega\mu\alpha\tau\iota\delta\acute{\iota}ou} + \beta(1 - X_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\omega\mu\alpha\tau\iota\delta\acute{\iota}ou}) \quad (2.6)$$

$$V_i^d = wV_i^d + a_1 V_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\omega\mu\alpha\tau\iota\delta\acute{\iota}ou} + a_2 V_{\beta\varepsilon\lambda\tau\iota\sigma\tau o\ \sigma\mu\acute{\eta}\nu ous} \quad (2.7)$$

$$X_i^d = 1, \alpha\nu\ V_i^d > r$$
$$X_i^d = 0, \alpha\nu\ V_i^d < r \quad \textbf{(2.8)}$$

Όπου r ένας τυχαίος αριθμός μεταξύ του 0 και του 1 και οι α και β παράμετροι του αλγορίθμου. Τέλος ο αλγόριθμος του Chen, μετά την κάθε μετακίνηση σωματιδίου του σμήνους ελέγχει αν η λύση που αυτό αντιπροσωπεύει είναι εφικτή και στην περίπτωση που δεν είναι την μετατρέπει σε εφικτή.

2.6.1.2 Προσέγγιση των (Belmecheri et al. (2010); Belmecheri, Prins, Yalaoui, and Amodeo (2012))

Στην προσέγγιση του Belmecheri αντί της αναπαράστασης της λύσης του VRP σε διάνυσμα πραγματικών αριθμών, προσαρμόζεται ο αλγόριθμος PSO ώστε το κάθε σωματίδιο δεν αναπαριστά μία λύση του VRP, αλλά μία σειρά προτίμησης των πελατών προς εξυπηρέτηση. Για την αξιολόγηση του κάθε σωματιδίου, η σειρά προτίμησης του (το διάνυσμα θέσης του) εισάγεται σε έναν ευρετικό αλγόριθμο ο οποίος κατασκευάζει ντετερμινιστικά μία λύση για το πρόβλημα και την αξιολογεί. Έτσι, ο αλγόριθμος διαιρείται σε δύο υπό-αλγορίθμους:

- έναν ευρετικό αλγόριθμο επίλυσης του VRP που παίρνει ως είσοδο μία σειρά προτίμησης πελατών και επιστρέφει μία λύση
- έναν PSO που τα σωματίδια του σμήνους του αναπαριστούν σειρές προτίμησης των πελατών. Ο αλγόριθμος αυτός πρακτικά βελτιστοποιεί τη λίστα αυτή.

Τέλος, ο αλγόριθμος επιστρέφει ως βέλτιστη λύση τη λύση που κατασκευάζεται από το βέλτιστο σωματιδίου που επιστρέφει ο αλγόριθμος PSO.

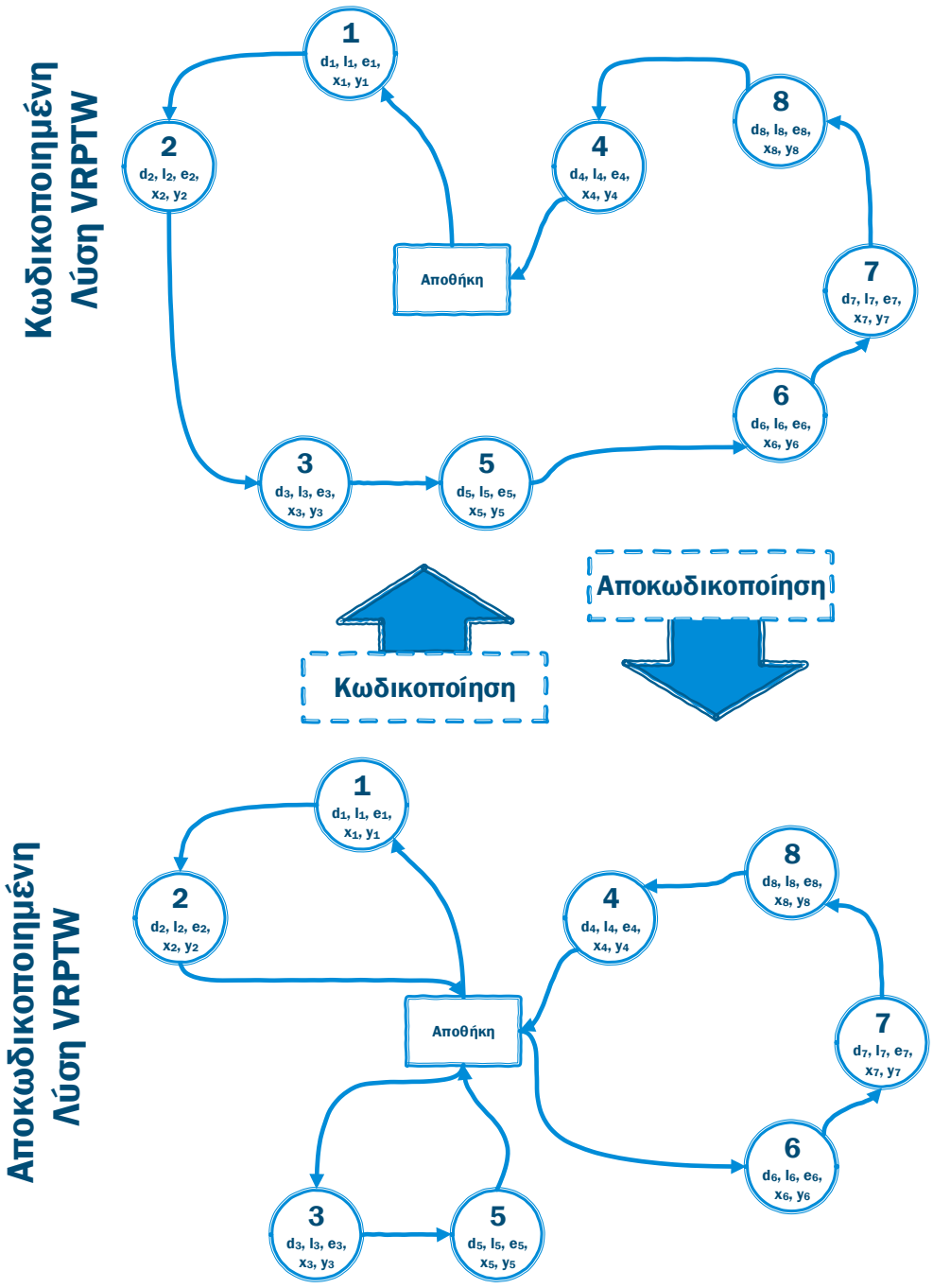
Παρουσιάζεται σε αλγόριθμο η προσέγγιση του Belmecheri

1. Αρχικοποίηση της θέσης και της ταχύτητας κάθε σωματιδίου του σμήνους
2. Δημιούργησε λύση του προβλήματος για κάθε σωματίδιο και αξιολόγησέ τη
3. Ανανέωσε το βέλτιστο του κάθε σωματιδίου και το ολικό βέλτιστο
4. Όσο τα κριτήρια τερματισμού δεν ικανοποιούνται
    a. Ανανέωσε την ταχύτητα του κάθε σωματιδίου σύμφωνα με το βέλτιστό του και σύμφωνα με το ολικό βέλτιστο
    b. Μετακίνησε τα σωματίδια ανάλογα με την ταχύτητά τους
    c. Δημιούργησε λύση του προβλήματος από τη θέση-λίστα προτίμησης του κάθε σωματιδίου
    d. Αξιολόγησε την ποιότητα του κάθε σωματιδίου και ανανέωσε το βέλτιστο του κάθε σωματιδίου και το ολικό βέλτιστο
5. Δημιούργησε λύση του προβλήματος από το βέλτιστο σωματίδιο του σμήνους και επέστρεψέ το ως έξοδο του αλγορίθμου

Αλγόριθμος 2.9 : Αλγόριθμος PSO Belmecheri (2012)

2.6.1.3 Προσέγγιση του (Gong et al., 2012)

Στην προσέγγιση του (Gong et al.) δίνεται έμφαση στην αναπαράσταση της λύσης του VRP με τέτοιο τρόπο, ώστε κατά την αποκωδικοποίηση της λύσης να μην είναι δυνατή η παραγωγή αδύνατης λύσης. Συγκεκριμένα, κατά την κωδικοποίηση της λύσης δημιουργείται μία αλληλουχία πελατών που εξυπηρετούνται από ένα και μόνο όχημα, το οποίο ξεκινά και καταλήγει στην κεντρική αποθήκη. Η κωδικοποιημένη μορφή της λύσης μπορεί να εισαχθεί στον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων ως ένα διάνυσμα διαστάσεων ίσου πλήθους με τον αριθμό των πελατών προς εξυπηρέτηση, όπου η κάθε διάσταση αναπαριστά τη σειρά προτεραιότητας του πελάτη στην κωδικοποιημένη μορφή της λύσης.



Σχήμα 2.15 Κωδικοποίηση της λύσης για τον PSO Gong et al. (2012)

Κατά την αποκωδικοποίηση της λύσης, ξεκινά η κατασκευή μιας λύσης του VRP σειριακά: αρχικά δημιουργείται μία διαδρομή οχήματος στην οποία προστίθενται σειριακά πελάτες σύμφωνα με τη σειρά προτεραιότητάς τους, μέχρις ότου ο επόμενος πελάτης στη σειρά προτεραιότητας δε μπορεί να προστεθεί λόγω παραβίασης κάποιου περιορισμού. Στη συνέχεια, δημιουργείται νέα διαδρομή οχήματος και συνεχίζεται η κατασκευή της λύσης μέχρι να δρομολογηθούν όλοι οι πελάτες.

Παρουσιάζεται ο αλγόριθμος της αποκωδικοποίησης σε ψευδοκώδικα στον οποίο θεωρείται ως είσοδος η σειρά εξυπηρέτησης του κάθε πελάτη – προτεραιότητα – στην κωδικοποιημένη λύση.

```
1. Δημιούργησε την πρώτη διαδρομή και θέσε την ως τρέχουσα
2. Όσο υπάρχουν αδρομολόγητοι πελάτες
    a. Επέλεξε τον αδρομολόγητο πελάτη με την μεγαλύτερη
       προτεραιότητα
    b. Αν μπορεί να δρομολογηθεί στην τρέχουσα διαδρομή,
       πρόσθεσέ τον σε αυτήν
    c. Αν δεν μπορεί να δρομολογηθεί στην τρέχουσα διαδρομή,
       δημιούργησε νέα διαδρομή και θέσε την ως τρέχουσα, και
       πρόσθεσέ τον σε αυτήν
```

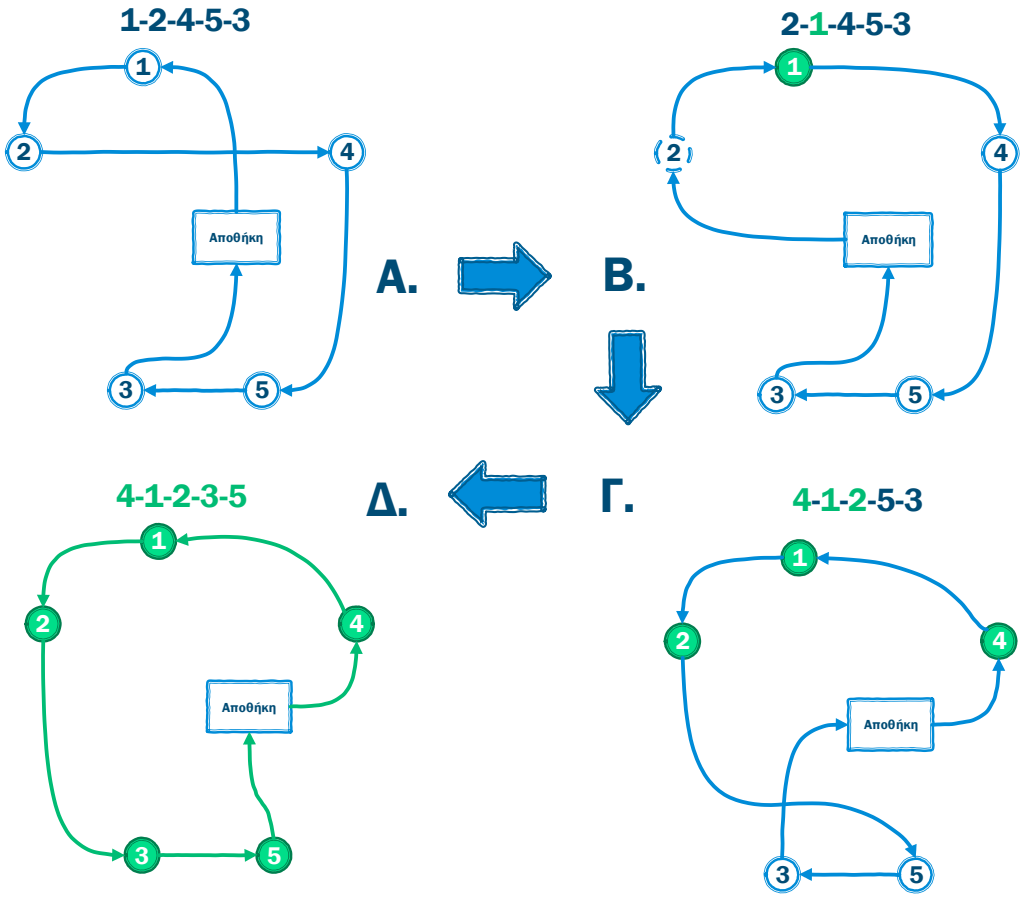
Αλγόριθμος 2.10 : Αποκωδικοποίηση Λύσης PSO Gong (2012)

2.6.1.4 Προσέγγιση των (Marinakis, Marinaki, & Dounias, 2010)

Στην προσέγγιση των (Marinakis et al., 2010) η κάθε διαδρομή της λύσης ενός VRP αναπαρίσταται από ένα δυαδικό διάνυσμα N διαστάσεων, όπου N το πλήθος των πελατών που εξυπηρετεί το όχημα που εκτελεί τη διαδρομή αυτή. Οι τιμές του διανύσματος αρχικά δηλώνουν την σειρά εξυπηρέτησής τους και παίρνουν ακέραιες τιμές, ενώ μετά την κωδικοποίησή του για την χρήση τους από τον αλγόριθμο προσομοίωσης σωματιδίων προσαρμόζονται ώστε να έχουν τιμές εντός του διαστήματος $(0\,,1)$ με τέτοιο τρόπο ώστε να διατηρείται η προτεραιότητα των πελατών.

Ιδιαίτερη έμφαση δίνεται σε μία μέθοδο που εκτελείται μετά την μετατόπιση του κάθε σωματιδίου, η οποία ονομάζεται Path Relinking, και είναι μια προσανατολισμένη μορφή μεθόδου τοπικής αναζήτησης. Η μέθοδος Path Relinking δέχεται ως είσοδο μία αρχική λύση, στην παρούσα περίπτωση την τρέχουσα θέση του σωματιδίου και μία λύση-στόχο. Κατά την εκτέλεσή της, ξεκινώντας από την αρχική λύση χρησιμοποιεί ένα μηχανισμό εναλλαγής της σειράς των πελατών ώστε να τη μετατρέψει στη λύση-στόχο. Στην πορεία της μεθόδου, αξιολογείται η κάθε ενδιάμεση λύση που εμφανίζεται και στην περίπτωση που είναι καλύτερη από την λύση-στόχο, το σωματίδιο παίρνει αυτήν την τιμή και η μέθοδος τερματίζει.

Παρουσιάζεται ένα παράδειγμα εκτέλεσης της μεθόδου Path Relinking, για ένα πρόβλημα πέντε πελατών και ενός οχήματος, για την καλύτερη κατανόηση της μεθόδου.



Σχήμα 2.16 Μέθοδος Path Relinking

Στο συγκεκριμένο παράδειγμα, η μέθοδος δέχεται ως αρχική λύση την λύση Α. και ως λύση στόχο την λύση Δ., ενώ κατά την εκτέλεσή της αξιολογεί διαδοχικά τις λύσεις Β. και Γ. Στην περίπτωση που κάποια από αυτές είναι καλύτερη της λύσης στόχου Δ., τότε η μέθοδος Path Relinking την επιστρέφει ως έξοδο.

3 Ορισμός του υπό μελέτη προβλήματος

Το πρόβλημα δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) είναι ειδίκευση του προβλήματος δρομολόγησης οχημάτων (VRP) συνεπώς παρουσιάζονται οι παραδοχές που ισχύουν αυτούσια στα δύο προβλήματα δρομολόγησης και στη συνέχεια παρουσιάζονται οι παραδοχές που ισχύουν αποκλειστικά στο VRPTW.

Στο πρόβλημα VRP έχουμε ένα στόλο από οχήματα, με τα οποία επιδιώκουμε να καλύψουμε τη ζήτηση ενός συνόλου πελατών, με προϊόντα που είναι αποθηκευμένα σε κάποια κεντρική αποθήκη. Κάνουμε τις παρακάτω παραδοχές-περιορισμούς:

- Τα διαθέσιμα οχήματα είναι πανομοιότυπα, δηλαδή έχουν την ίδια χωρητικότητα
- Το κάθε όχημα εκτελεί ένα και μόνο δρομολόγιο. Ξεκινά το δρομολόγιό από την κεντρική αποθήκη και καταλήγει στο τέλος της διαδρομής του στην ίδια κεντρική αποθήκη
- Οι πελάτες θα εξυπηρετηθούν μία φορά από ένα και μόνο ένα όχημα του στόλου. Θεωρείται ότι η ζήτηση του κάθε πελάτη είναι μικρότερη ή ίση με τη χωρητικότητα του κάθε οχήματος.
- Είναι προφανές ότι το άθροισμα της ζήτησης όλων των πελατών που θα εξυπηρετηθούν από κάποιο όχημα δεν πρέπει να ξεπερνά την χωρητικότητα του οχήματος

Στο πρόβλημα VRPTW προσθέτουμε τις παρακάτω παραδοχές-περιορισμούς:

- Το όχημα που θα εξυπηρετήσει τον κάθε πελάτη θα πρέπει να φτάσει σε αυτόν σε συγκεκριμένο «παράθυρο χρόνου», το οποίο ορίζει ο πελάτης
- Τα οχήματα μπορούν να φτάσουν σε κάποιον πελάτη νωρίτερα από την νωρίτερη δυνατή άφιξη που έχει ορίσει ο πελάτης (και να περιμένουν να «ανοίξει» το παράθυρο χρόνου) όμως δεν μπορούν να φτάσουν αργότερα από την αργότερη δυνατή άφιξη
- Όλα τα οχήματα πρέπει να έχουν επιστρέψει στην αποθήκη μέχρι κάποια συγκεκριμένη χρονική στιγμή

Στόχος του αλγορίθμου που αναπτύσσεται είναι η εύρεση του δρομολογίου που πρέπει να κάνει το κάθε όχημα, δηλαδή από ποιούς πελάτες θα περάσει το κάθε διαθέσιμο όχημα του στόλου και με ποιά σειρά, έτσι ώστε να ικανοποιούνται οι παραπάνω περιορισμοί και να ελαχιστοποιούνται τα οχήματα που χρησιμοποιούνται και το συνολικό κόστος μεταφοράς των προϊόντων. Οι αντικειμενικοί αυτοί στόχοι προς βελτιστοποίηση αναφέρονται αναλυτικά στο κεφάλαιο 4.2.

4 Μοντελοποίηση του Προβλήματος

4.1 Ορισμοί – Μεταβλητές και τι σημαίνουν και παραδοχές

Το VRPTW μοντελοποιείται ως ένα σύνολο ομοίων οχημάτων, ένα κόμβο κεντρικής αποθήκης, ένα σύνολο κόμβων πελατών και ένα δίκτυο που συνδέει την κεντρική αποθήκη και τους πελάτες μεταξύ τους.

Θεωρείται ότι υπάρχουν $N + 1$ κόμβοι όπου ο κόμβος 0 αντιπροσωπεύει την κεντρική αποθήκη και οι υπόλοιποι αντιπροσωπεύουν τους πελάτες προς εξυπηρέτηση. Τα οχήματα είναι πλήθους K. Κάθε τόξο του δικτύου αναπαριστά την ένωση μεταξύ δύο κόμβων καθώς και την κατεύθυνση με την οποία ταξιδεύει. Ο αριθμός των διαδρομών στο δίκτυο είναι ίσος με τον αριθμό των χρησιμοποιούμενων οχημάτων. Κάθε όχημα εκτελεί αποκλειστικά μία διαδρομή, ενώ η κάθε διαδρομή αποτελείται από ένα σύνολο κόμβων με συγκεκριμένη σειρά προτεραιότητας, ενώ πρώτος και τελευταίος κόμβος είναι ο κόμβος 0 που αναπαριστά την κεντρική αποθήκη. Ένα κόστος c_{ij} και ένα χρονικό διάστημα t_{ij} χαρακτηρίζουν το κάθε τόξο του δικτύου. Κάθε πελάτης του δικτύου εξυπηρετείται μία και μόνο φορά από ένα και μόνο όχημα του στόλου οχημάτων. Κάθε όχημα έχει την ίδια χωρητικότητα Q_k και κάθε πελάτης χαρακτηρίζεται από μία ζήτηση q_i. Η χωρητικότητα του κάθε οχήματος πρέπει να είναι ίση ή μεγαλύτερη από την ζήτηση των πελατών που εξυπηρετεί η αντίστοιχη διαδρομή. Ο περιορισμός παραθύρων χρόνου υποδηλώνεται από τη νωρίτερη δυνατή έναρξη εξυπηρέτησης e_i και την αργότερη δυνατή έναρξη εξυπηρέτησης l_i που χαρακτηρίζει τον κάθε πελάτη i. Το κάθε όχημα φτάνει στους πελάτες που εξυπηρετεί νωρίτερα της αργότερης δυνατής έναρξης εξυπηρέτησής τους l_i, ενώ αν φτάσει πριν της νωρίτερης δυνατής έναρξης εξυπηρέτησης e_i αναμένει στον πελάτη μέχρι να δύναται να τον εξυπηρετήσει. Ο κάθε πελάτης χαρακτηρίζεται από έναν απαιτούμενο για την εξυπηρέτησή του χρονικό διάστημα s_i που αναπαριστά τον απαιτούμενο χρόνο για την φόρτωση ή εκφόρτωση των αγαθών. Τα οχήματα περιορίζονται να ολοκληρώσουν τη διαδρομή τους μέχρι ένα χρονικό όριο T_k.

Ορίζεται η μεταβλητή απόφασης $x_{ijk} (i, j \in \{0, \dots, N\}; k \in \{1, \dots, K\})$ είναι 1 στην περίπτωση που το όχημα k ταξιδεύει από τον πελάτη i στον πελάτη j, δηλαδή αν χρησιμοποιείται το τόξο $i - j$ ενώ σε διαφορετική περίπτωση είναι 0. Ορίζεται μεταβλητή απόφασης t_i η οποία δηλώνει τη χρονική στιγμή που ένα όχημα καταφθάνει στον πελάτη i και η μεταβλητή απόφασης w_i η οποία δηλώνει το χρόνο που περιμένει το όχημα που εξυπηρετεί τον πελάτη i για να ανοίξει το χρονικό παράθυρο εξυπηρέτησής του. Οι μεταβλητές t_i και w_i χρησιμοποιούνται ώστε να γίνει δυνατή η χρήση χρονικών περιορισμών και υπολογίζονται από τις παραμέτρους του προβλήματος και από τη μεταβλητή απόφασης x_{ijk}.

4.1.1 Μεταβλητές απόφασης

Ορίζονται τρία είδη μεταβλητών x_{ijk}, t_i και w_i, εκ των οποίων το πρώτο (x_{ijk}) αναπαριστά την εκτέλεση ή μη της διαδρομής από τον πελάτη i προς τον πελάτη j από το όχημα k κατά την εκτέλεση του δρομολογίου του. Συγκεκριμένα, αν το εν λόγω όχημα εκτελεί την διαδρομή αυτή, η αντίστοιχη μεταβλητή x_{ijk} παίρνει την τιμή 1 ενώ σε διαφορετική περίπτωση παίρνει την τιμή 0. Για ένα πρόβλημα N πελατών και K οχημάτων, δημιουργούνται $N+1$ δυαδικές ανεξάρτητες μεταβλητές για κάθε πελάτη και για κάθε όχημα, ώστε να συμπεριληφθούν όλοι οι πιθανοί προορισμοί οποιουδήποτε οχήματος από αυτόν τον πελάτη, συνεπώς, το πλήθος των μεταβλητών απόφασης x_{ijk} είναι $(N+1)*N*K$.

Το δεύτερο είδος μεταβλητών απόφασης είναι οι t_i που αναπαριστούν τη χρονική στιγμή της έναρξης της εξυπηρέτησης του κάθε πελάτη i. Συνεπώς, για ένα πρόβλημα N πελατών το πλήθος των μεταβλητών απόφασης t_i είναι N. Οι μεταβλητές t_i υπολογίζονται από τις μεταβλητές απόφασης x_{ijk} και από τις παραμέτρους του προβλήματος t_{ij}, s_i και e_i.

Το τρίτο είδος μεταβλητών είναι οι w_i που αναπαριστούν το χρονικό διάστημα ανάμεσα στην άφιξη του οχήματος που θα εξυπηρετήσει τον πελάτη i στον πελάτη i και την έναρξη εξυπηρέτησης του πελάτη i. Συνεπώς, για ένα πρόβλημα N πελατών το πλήθος των μεταβλητών απόφασης w_i είναι N. Οι μεταβλητές w_i υπολογίζονται από τη χρονική στιγμή που καταφθάνει κάποιο όχημα στον πελάτη i (μεταβλητές απόφασης t_i) και από τη νωρίτερη δυνατή εξυπηρέτηση του κάθε πελάτη (παράμετροι του προβλήματος e_i).

Παρουσιάζονται συνοπτικά οι μεταβλητές απόφασης του VRPTW:

Πίνακας 4.1 Μεταβλητές Απόφασης

Μεταβλητή	Είδος	Περιγραφή
x_{ijk}	δυαδική	1 , αν το όχημα k χρησιμοποιεί το τόξο $i-j$
		0, αν το όχημα k δεν χρησιμοποιεί το τόξο $i-j$
t_i	πραγματική	η χρονική στιγμή που ξεκινά να εξυπηρετείται ο πελάτης i
w_i	πραγματική	το χρονικό διάστημα ανάμεσα στην άφιξη του οχήματος στον πελάτη i, και της έναρξης εξυπηρέτησης του πελάτη i

Για την καλύτερη κατανόησή τους, παρουσιάζονται οι πίνακες που περιλαμβάνουν τις ανεξάρτητες μεταβλητές απόφασης x_{ijk}, μετά τη λύση ενός προβλήματος οκτώ πελατών και τριών οχημάτων. Συνεπώς, $N=8$ και $K=3$. Δημιουργούνται τρεις πίνακες μεταβλητών x_{ijk}, ένας για κάθε όχημα, ενώ ο κάθε πίνακας αποτελείται από $(N+1)*N=(8+1)*8=72$ μεταβλητές, ώστε να περιλαμβάνει μία μεταβλητή απόφασης για κάθε αφετηρία προς κάθε προορισμό, είτε πελάτη είτε την κεντρική αποθήκη. Συνολικά, δημιουργούνται $K*(N+1)*N=3*(8+1)*8=216$ μεταβλητές απόφασης x_{ijk}. Δεν περιλαμβάνονται

οι μεταβλητές που αποτελούν τη διαγώνιο των πινάκων, καθώς η αφετηρία και ο προορισμός του οχήματος σε κάθε του μετάβαση επιβάλλεται να μην ταυτίζονται.

προς τον πελάτη **j**

Xij1	0	1	2	3	4	5	6	7	8
0	0	**1**	0	0	0	0	0	0	0
1	0	0	**1**	0	0	0	0	0	0
2	**1**	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

από τον πελάτη **i** — **k=1**

προς τον πελάτη **j**

Xij2	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	**1**	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	**1**	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	**1**	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

από τον πελάτη **i** — **k=2**

προς τον πελάτη **j**

Xij3	0	1	2	3	4	5	6	7	8
0	0	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	**1**
5	0	0	0	0	0	0	0	0	0
6	**1**	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	**1**	0	0
8	0	0	0	0	0	0	0	**1**	0

από τον πελάτη **i** — **k=3**

Σχήμα 4.1 : Παράδειγμα Μεταβλητών Απόφασης x_{ijk}

4.1.2 Παράμετροι προβλήματος

Για τον καθορισμό ενός προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) απαιτείται ο προκαθορισμός συγκεκριμένων παραμέτρων που περιγράφουν τα χαρακτηριστικά των πελατών, των οχημάτων, της κεντρικής αποθήκης και του δικτύου μετάβασης των οχημάτων από και προς κάθε κόμβο, είτε αυτός είναι πελάτης είτε η κεντρική αποθήκη.

- το πλήθος των πελατών N, όπου το N είναι φυσικός αριθμός
- το πλήθος των διαθέσιμων οχημάτων K, όπου το K είναι φυσικός αριθμός
- η χωρητικότητά των διαθέσιμων οχημάτων Q_k, όπου το Q_k είναι πραγματικός αριθμός και αντιπροσωπεύει την ποσότητα των αγαθών προς μεταφορά που δύναται να μεταφέρει το κάθε όχημα
- η ζήτησή των πελατών q_i, όπου $i = 1, \dots, N$ η οποία είναι πραγματικός αριθμός που αντιπροσωπεύει την ποσότητα των αγαθών προς μεταφορά που ζητά ο κάθε πελάτης i και δεν ξεπερνά τη χωρητικότητα των διαθέσιμων οχημάτων Q_k
- ο απαιτούμενος χρόνος για την εξυπηρέτησή των πελατών s_i, όπου $i = 1, \dots, N$ και s_i πραγματικός αριθμός που αντιπροσωπεύει το χρόνο που χρειάζεται για την εκφόρτωση της ποσότητας q_i των αγαθών προς μεταφορά στον πελάτη i.

- τα όρια του παραθύρου χρόνου e_i και l_i , όπου $i = 1, \dots, N$ και e_i και l_i είναι πραγματικοί αριθμοί που αντιπροσωπεύουν τη νωρίτερη και την αργότερη δυνατή στιγμή εξυπηρέτησης του πελάτη i, αντίστοιχα.
- το χρονικό όριο επιστροφής τους στην κεντρική αποθήκη T_k , όπου T_k είναι πραγματικός αριθμός.
- το κόστος μετάβασης κάποιου οχήματος από κάθε πελάτη (ή την κεντρική αποθήκη) προς κάθε πελάτη (ή την κεντρική αποθήκη) c_{ij}, όπου $i = 0, \dots, N$ και $j = 0, \dots, N$ και $i \neq j$, καθώς η αφετηρία και ο προορισμός του οχήματος επιβάλλεται να μην ταυτίζονται. Συνεπώς, η παράμετρος c_{ij} αντιπροσωπεύει το κόστος της μεταφοράς αγαθών από τον πελάτη (ή την κεντρική αποθήκη) i στον πελάτη (ή την κεντρική αποθήκη) j.
- Τον χρόνο μετάβασης κάποιου οχήματος από κάθε πελάτη (ή την κεντρική αποθήκη) προς κάθε πελάτη (ή την κεντρική αποθήκη) t_{ij}, όπου $i = 0, \dots, N$ και $j = 0, \dots, N$ και $i \neq j$, καθώς η αφετηρία και ο προορισμός του οχήματος επιβάλλεται να μην ταυτίζονται. Συνεπώς, η παράμετρος t_{ij} αντιπροσωπεύει το χρόνο που απαιτείται για τη μετάβαση κάποιου οχήματος από τον πελάτη (ή την κεντρική αποθήκη) i στον πελάτη (ή την κεντρική αποθήκη) j.

Πίνακας 4.2 Παράμετροι Προβλήματος

Μεταβλητή	Είδος	Περιγραφή
N	ακέραιος	το πλήθος των πελατών
K	ακέραιος	το πλήθος των διαθέσιμων οχημάτων
c_{ij}	πραγματικός	το κόστος μετάβασης κάποιου οχήματος από τον πελάτη i στον πελάτη j (ή την κεντρική αποθήκη για $i = 0$ ή $j = 0$)
q_i	πραγματικός	η ζήτηση του πελάτη i
t_{ij}	πραγματικός	ο χρόνος που απαιτείται από κάποιο όχημα για την μετάβαση από τον κόμβο i στον κόμβο j
s_i	πραγματικός	ο χρόνος που απαιτείται για να εξυπηρετηθεί ο πελάτης i
e_i	πραγματικός	η χρονική στιγμή έναρξης του χρονικού παραθύρου του πελάτη i
l_i	πραγματικός	η χρονική στιγμή λήξης του χρονικού παραθύρου του πελάτη i
T_k	πραγματικός	η χρονική στιγμή μέχρι την οποία επιτρέπεται η επιστροφή του οχήματος k στην αποθήκη
Q_k	πραγματικός	Χωρητικότητα του οχήματος k

Για την καλύτερη κατανόηση των παραμέτρων του VRPTW παρουσιάζονται οι μεταβλητές c_{ij} στο Σχήμα (4.2) που αναπαριστούν τη μετάβαση από κάθε κόμβο (είτε πελάτη, είτε κεντρική αποθήκη) προς κάθε άλλο, για ένα παράδειγμα προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου με οκτώ πελατών ($N = 8$) και τριών οχημάτων ($K = 3$). Συνεπώς, δημιουργούνται $N * (N + 1) = 8 * (8 + 1) = 72$ μεταβλητές απόφασης c_{ij} για τον καθορισμό του προβλήματος.

προς τον πελάτη **j**

από τον πελάτη **i**

Cij	0	1	2	3	4	5	6	7	8
0	0	3.0382	0.9952	1.8873	0.7353	2.7657	2.59	2.4736	0.8147
1	3.0382	0	2.6352	0.5651	3.4743	1.195	1.0261	1.8137	3.059
2	0.9952	2.6352	0	2.6984	2.1063	0.925	1.8297	3.4545	0.7757
3	1.8873	0.5651	2.6984	0	2.9062	1.1781	0.508	0.8843	2.8633
4	0.7353	3.4743	2.1063	2.9062	0	2.7422	0.8081	3.3272	1.661
5	2.7657	1.195	0.925	1.1781	2.7422	0	2.4627	3.0536	2.1663
6	2.59	1.0261	1.8297	0.508	0.8081	2.4627	0	2.8238	3.4356
7	2.4736	1.8137	3.4545	0.8843	3.3272	3.0536	2.8238	0	2.4851
8	0.8147	3.059	0.7757	2.8633	1.661	2.1663	3.4356	2.4851	0

Σχήμα 4.2 : Παράδειγμα Παραμέτρων Προβλήματος c_{ij}

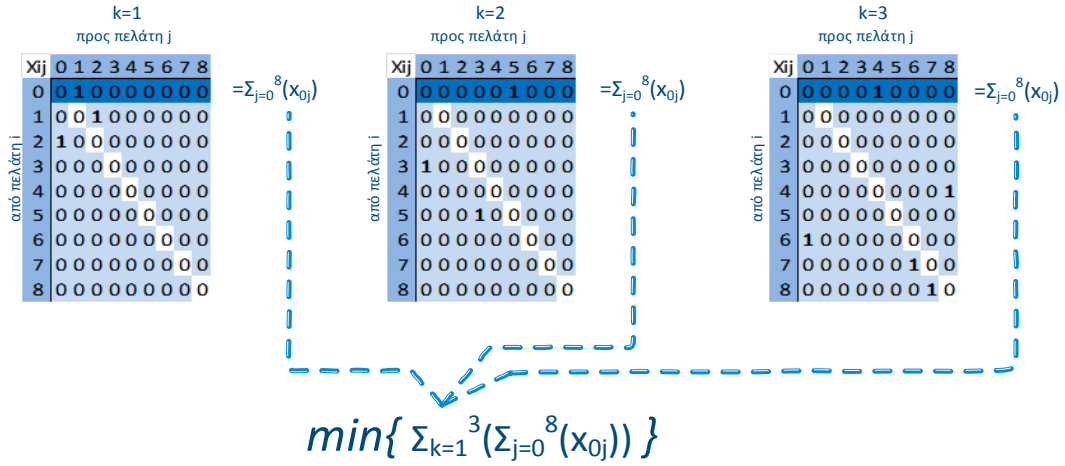
4.2 Αντικειμενική συνάρτηση

Στην παρούσα μελέτη θεωρείται πως το αρχικό εφάπαξ κόστος χρήσης των οχημάτων είναι κατά πολύ μεγαλύτερο του μεταβλητού κόστους που εξαρτάται από την διανυόμενη απόσταση, συνεπώς οι λύσεις του προβλήματος που υπερτερούν σε πλήθος οχημάτων θεωρούνται βέλτιστες ανεξάρτητα με την διανυόμενη απόσταση των οχημάτων. Αντικειμενικοί στόχοι για την επίλυση του προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου στην παρούσα μελέτη είναι αρχικά η ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων και στη συνέχεια η ελαχιστοποίηση της διανυόμενης διαδρομής από το σύνολο των οχημάτων. Αντίστοιχα με τους αντικειμενικούς στόχους δημιουργούνται δύο αντικειμενικές συναρτήσεις που καθορίζουν το βασικό άξονα βελτιστοποίησης της προτεινόμενης μεθόδου.

Η πρώτη αντικειμενική συνάρτηση ορίζει τον πρωταρχικό στόχο προς βελτιστοποίηση, ο οποίος είναι η ελαχιστοποίηση των χρησιμοποιούμενων οχημάτων. Συγκεκριμένα, ο αριθμός των χρησιμοποιούμενων οχημάτων ισούται με τον αριθμό των οχημάτων που αναχωρούν από την κεντρική αποθήκη με προορισμό οποιονδήποτε πελάτη.

$$min\left\{\sum_{k=1}^{K}\sum_{j=1}^{N} x_{hjk}\right\}, για\ h = 0 \qquad (4.1)$$

Για την καλύτερη κατανόηση της πρώτης αντικειμενικής συνάρτησης παρουσιάζονται οι πίνακες των μεταβλητών απόφασης x_{ijk} από τους οποίους προκύπτει, για ένα παράδειγμα οκτώ πελατών ($N = 8$) και τριών οχημάτων ($K = 3$).



Σχήμα 4.3 Παράδειγμα υπολογισμού της πρώτης αντικειμενικής συνάρτησης

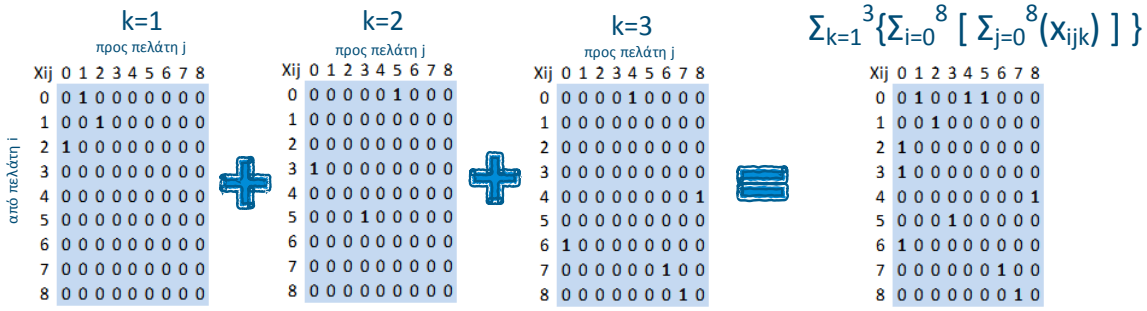
Όπως παρουσιάζεται στο Σχήμα 4.3, αρχικά προσθέτονται τα στοιχεία της πρώτης γραμμής του κάθε πίνακα και στη συνέχεια αυτά αθροίζονται μεταξύ τους ώστε να δημιουργηθεί η παράσταση προς ελαχιστοποίηση.

Η δεύτερη αντικειμενική συνάρτηση ορίζει το δευτερεύον στόχο προς βελτιστοποίηση, δηλαδή την ελαχιστοποίηση της συνολικής απόστασης που διανύουν τα οχήματα κατά την εκτέλεση των δρομολογίων τους. Συγκεκριμένα, η συνολική απόσταση που διανύει το κάθε όχημα ισούται με το άθροισμα του κόστους των διαδρομών που χρησιμοποιεί. Συνεπώς, αθροίζονται κατά στοιχείο οι πίνακες των μεταβλητών x_{ijk} για κάθε όχημα $k = 1, \dots, K$ ώστε να δημιουργηθεί ο πίνακας με τις διαδρομές που χρησιμοποιούνται από οποιοδήποτε όχημα. Στη συνέχεια, ο πίνακας που προκύπτει πολλαπλασιάζεται κατά στοιχείο με τον πίνακα κόστους των διαδρομών c_{ij}, όπου $i = 0, \dots, N$ και $j = 0, \dots, N$, ώστε να αφαιρεθούν από αυτόν τα κόστη των διαδρομών που δεν χρησιμοποιούνται. Τέλος, αθροίζονται τα στοιχεία του πίνακα που προκύπτει ώστε να προκύψει η ποσότητα προς ελαχιστοποίηση.

$$min\left\{\sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} c_{ij}x_{ijk}\right\} \quad \textbf{(4.2)}$$

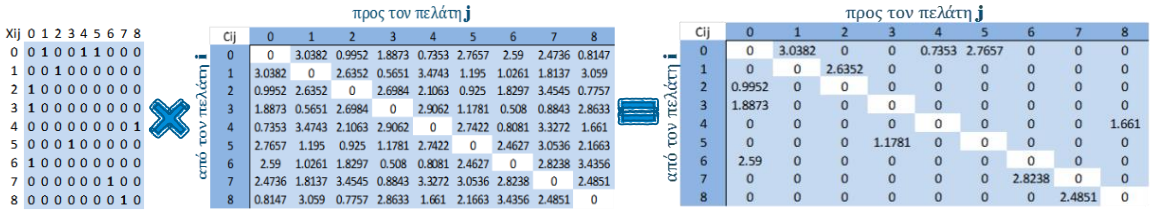
Για την καλύτερη κατανόηση της δεύτερης αντικειμενικής συνάρτησης παρουσιάζονται οι πίνακες των μεταβλητών x_{hjk} και των παραμέτρων c_{ij} από τους οποίους προκύπτει, για ένα παράδειγμα οκτώ πελατών και τριών οχημάτων.

Αρχικά, προστίθενται οι μεταβλητές x_{ij1}, x_{ij2} και x_{ij3} ώστε να δημιουργηθεί ο δυαδικός πίνακας χρήσης του δικτύου.



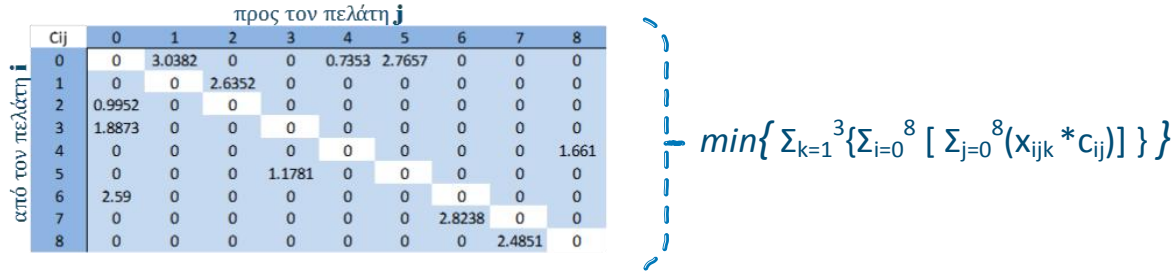
Σχήμα 4.4 Πρώτο βήμα υπολογισμών παραδείγματος της δεύτερης αντικειμενικής συνάρτησης

Στη συνέχεια πολλαπλασιάζεται με τον πίνακα κόστους διαδρομής c_{ij} ανά στοιχείο.



Σχήμα 4.5 Δεύτερο βήμα υπολογισμών παραδείγματος της δεύτερης αντικειμενικής συνάρτησης

Τέλος, προθέτονται τα στοιχεία του τελικού πίνακα ώστε να προκύψει η ποσότητα προς ελαχιστοποίηση.



Σχήμα 4.6 Τρίτο βήμα υπολογισμών παραδείγματος της δεύτερης αντικειμενικής συνάρτησης

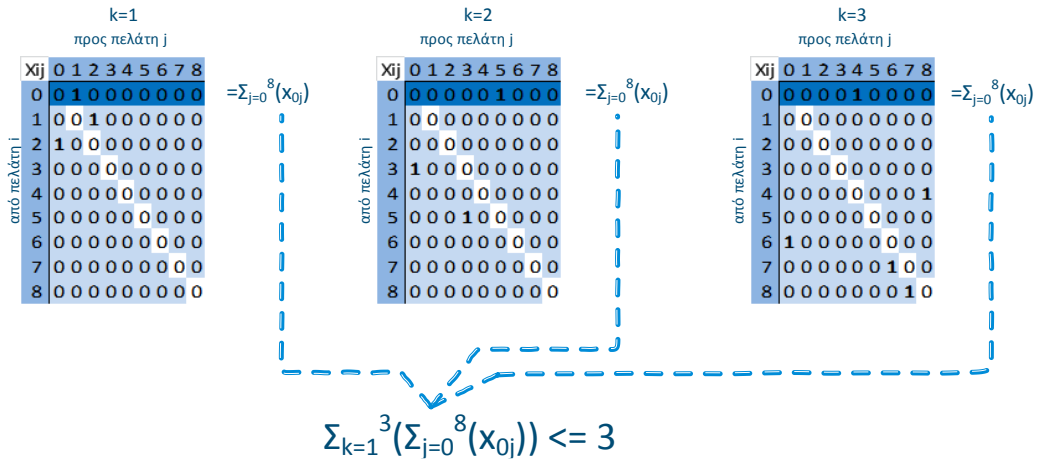
4.3 Περιορισμοί

Παρουσιάζεται μια σύντομη λεκτική περιγραφή των περιορισμών. Επίσης, παρουσιάζονται αναλυτικά οι πίνακες των μεταβλητών ενός παραδείγματος του VRPTW οκτώ πελατών και τριών οχημάτων για την καλύτερη κατανόηση των ανισώσεων των περιορισμών.

- ο αριθμός των οχημάτων που θα αναχωρήσουν από την κεντρική αποθήκη ο οποίος δεν πρέπει να ξεπερνά το συνολικό αριθμό των διαθέσιμων οχημάτων K.

$$\sum_{k=1}^{K}\sum_{j=0}^{N} x_{hjk} \leq K, για\ h = 0 \qquad (4.3)$$

Ο αριθμός των οχημάτων που χρησιμοποιούνται στη λύση του προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) ισούται με το σύνολο τον οχημάτων που αναχωρούν από την κεντρική αποθήκη, δηλαδή το άθροισμα της πρώτης γραμμής του πίνακα των μεταβλητών x_{ijk} για κάθε όχημα, το οποίο και περιορίζεται όπως παρουσιάζεται στο επόμενο σχήμα.



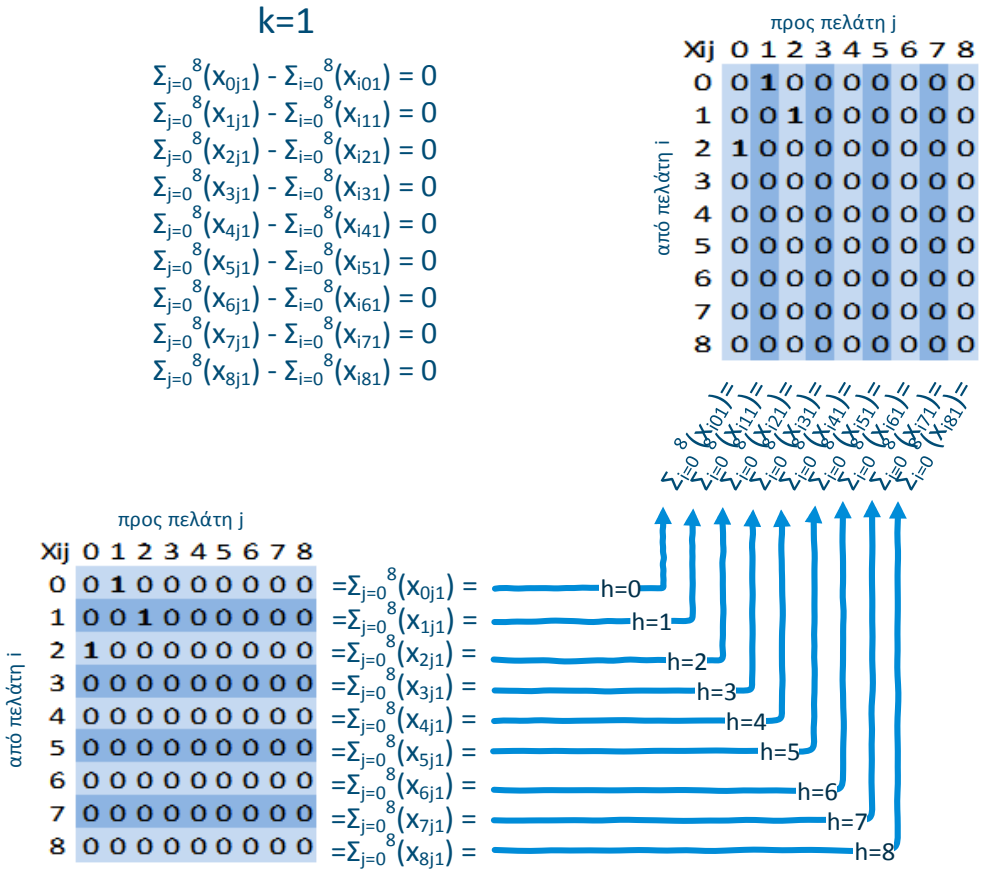
Σχήμα 4.7 Υπολογισμοί παραδείγματος του περιορισμού (4.3)

- αν κάποιο όχημα k, από οποιονδήποτε πελάτη, μεταβεί σε κάποιον πελάτη h τότε θα αναχωρήσει και από τον πελάτη h προς οποιονδήποτε πελάτη

$$\sum_{i=1}^{N} x_{ihk} - \sum_{j=1}^{N} x_{hjk} = 0\ , για\ κάθε\ h \in \{0, \dots, N\}\ και\ για\ κάθε\ k \in \{1, \dots, K\} \qquad (4.4)$$

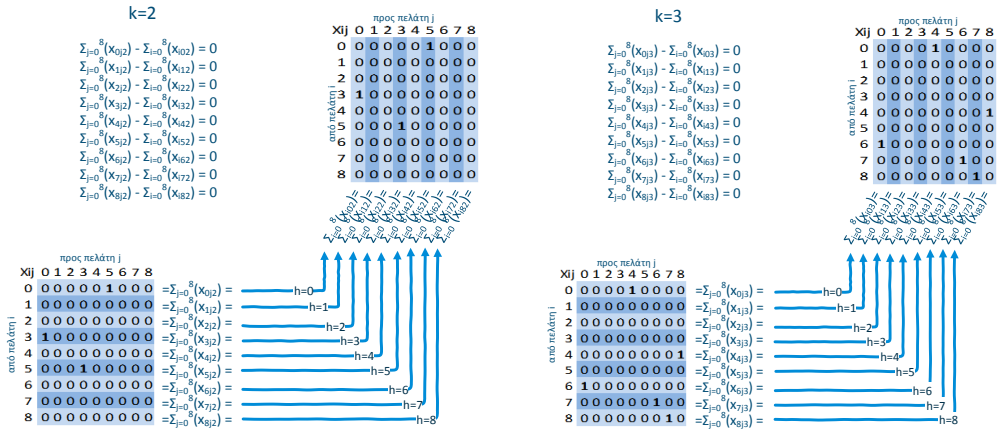
Αν κάποιο όχημα καταφθάσει σε κάποιον πελάτη i, τότε η στήλη i του πίνακα μεταβλητών x_{ijk} του οχήματος θα έχει άθροισμα ίσο με 1, διαφορετικά θα έχει άθροισμα 0. Αν κάποιο όχημα αναχωρήσει από κάποιον πελάτη i, τότε η γραμμή i του πίνακα μεταβλητών x_{ijk} του οχήματος θα έχει άθροισμα ίσο με 1, διαφορετικά θα έχει άθροισμα 0. Συνεπώς, για να ικανοποιηθεί ο εν λόγω περιορισμός πρέπει για τον πίνακα των μεταβλητών x_{ijk} του κάθε οχήματος, το άθροισμα κάθε στήλης i να ισούται με το άθροισμα της αντίστοιχης σειράς i. Παρουσιάζεται μία εξίσωση για κάθε στήλη (ή γραμμή) του πίνακα, και για κάθε όχημα, συνεπώς συνολικά $(N + 1) * K$ εξισώσεις.

Αρχικά, παρουσιάζονται οι εξισώσεις που αφορούν το πρώτο όχημα προς δρομολόγηση.



Σχήμα 4.8 Υπολογισμοί παραδείγματος του περιορισμού (4.4) βήμα α'

Αντίστοιχα, παρουσιάζονται οι εξισώσεις για τις γραμμές και τις στήλες των πινάκων των μεταβλητών x_{ijk} του δεύτερου και του τρίτου οχήματος.



Σχήμα 4.9 Υπολογισμοί παραδείγματος του περιορισμού (4.4) βήμα β'

Τέλος, παρουσιάζονται συνοπτικά οι εξισώσεις του περιορισμού.

$$\Sigma_{j=0}^{8}(x_{hjk}) - \Sigma_{i=0}^{8}(x_{ihk}) = 0,$$

για κάθε πελάτη h

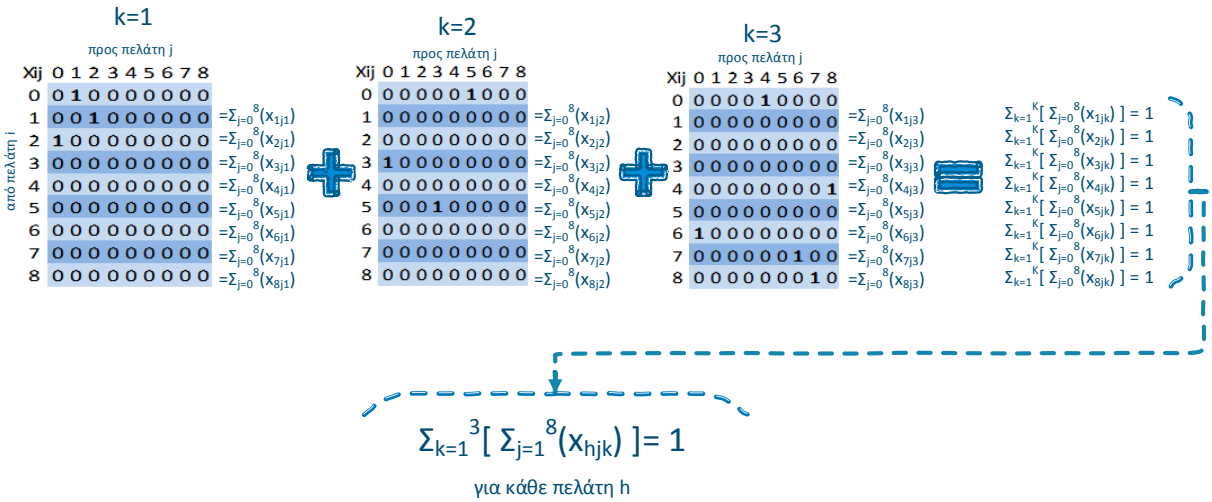
για κάθε πελάτη k

Σχήμα 4.10 Υπολογισμοί παραδείγματος του περιορισμού (4.4) βήμα γ'

- πρέπει ο κάθε πελάτης να εξυπηρετείται από ένα και μόνο όχημα, συνεπώς από σε κάθε πελάτη h θα πρέπει να αναχωρεί ένα και μόνο όχημα προς οποιονδήποτε πελάτη

$$\sum_{k=1}^{K}\sum_{j=0}^{N} x_{hjk} = 1\,, για\ κάθε\ h \in \{1, \dots, N\} \qquad (4.5)$$

Το άθροισμα της σειράς i, αθροιστικά για όλα τα οχήματα πρέπει να είναι ίσο με 1, επειδή αν ένα όχημα αναχωρήσει από κάποιον πελάτη, τότε τα υπόλοιπα οχήματα δεν πρέπει να αναχωρήσουν από τον ίδιο πελάτη. Εξαίρεση αποτελεί η πρώτη σειρά που αντιπροσωπεύει την κεντρική αποθήκη από την οποία αναχωρούν όλα τα οχήματα. Ο περιορισμός αποτελείται συνολικά από N εξισώσεις.

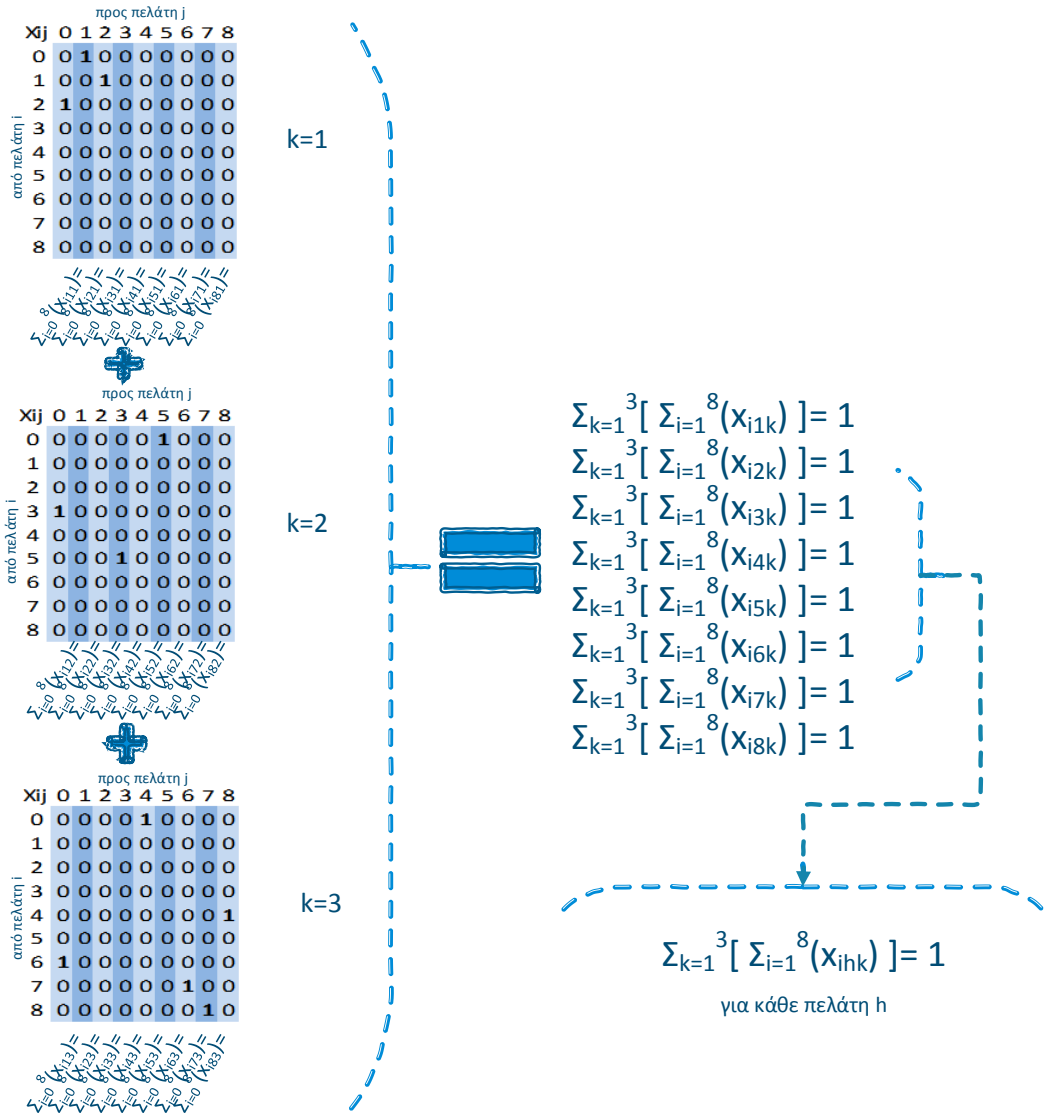


Σχήμα 4.11 Υπολογισμοί παραδείγματος του περιορισμού (4.5)

- και προς κάθε πελάτη h πρέπει να καταφθάνει ένα και μόνο όχημα από οποιονδήποτε πελάτη

$$\sum_{k=1}^{K}\sum_{i=0}^{N} x_{ihk} = 1\,, για\ κάθε\ h \in \{1,\dots,N\} \qquad (4.6)$$

Το άθροισμα της στήλης i, αθροιστικά για όλα τα οχήματα πρέπει να είναι ίσο με 1, επειδή αν ένα όχημα καταφθάσει σε κάποιον πελάτη, τότε τα υπόλοιπα οχήματα δεν πρέπει να καταφθάσουν στον ίδιο πελάτη. Εξαίρεση αποτελεί η πρώτη σειρά που αντιπροσωπεύει την κεντρική αποθήκη στην οποία καταφθάνουν όλα τα οχήματα. Ο περιορισμός αποτελείται συνολικά από N εξισώσεις.

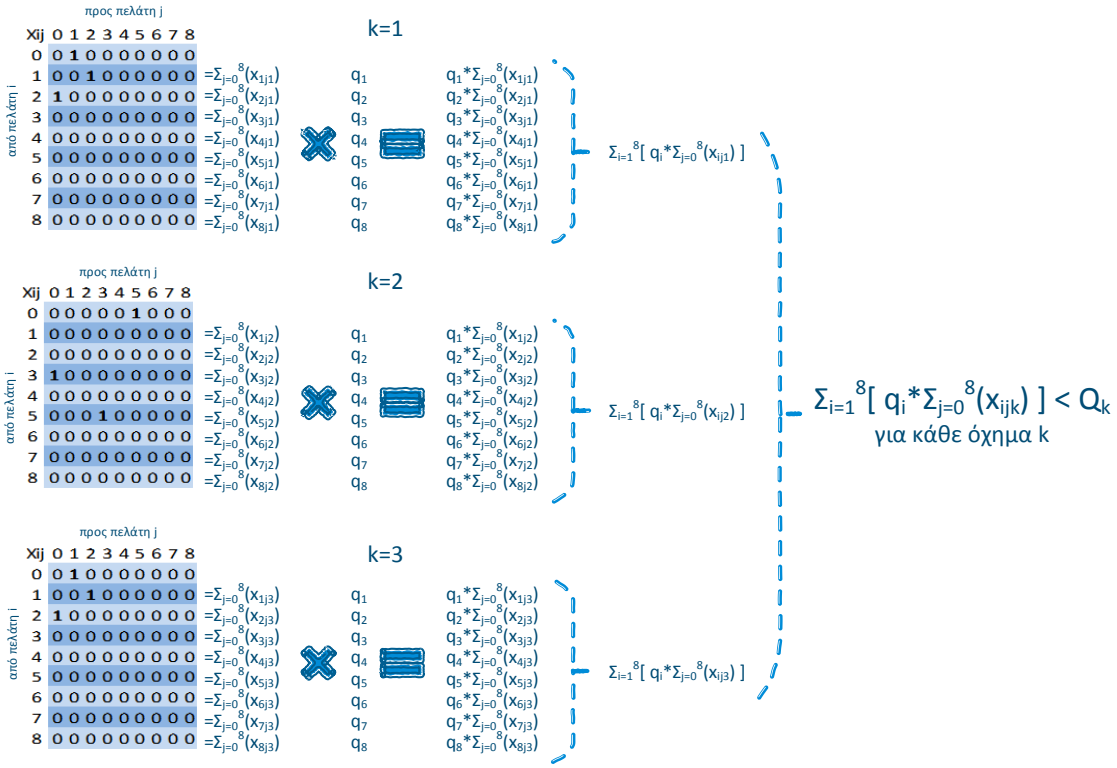


Σχήμα 4.12 Υπολογισμοί παραδείγματος του περιορισμού (4.6)

- το άθροισμα της ζήτησης των πελατών που εξυπηρετείται από το κάθε όχημα δεν πρέπει να ξεπερνά τη χωρητικότητα του οχήματος

$$\sum_{i=1}^{N} q_i \sum_{j=0}^{N} x_{ijk} \leq Q_k, \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ k \in \{1, \dots, K\} \qquad \textbf{(4.7)}$$

Η ζήτηση των πελατών που εξυπηρετεί κάποιο όχημα υπολογίζεται πολλαπλασιάζοντας κατά στοιχείο τις ζητήσεις των πελατών q_i με το άθροισμα της αντίστοιχης σειράς του πίνακα x_{ijk}, και τέλος αθροίζοντας τα γινόμενα που προκύπτουν, για κάθε όχημα. Ο περιορισμός αυτός αποτελείται από K ανισώσεις.

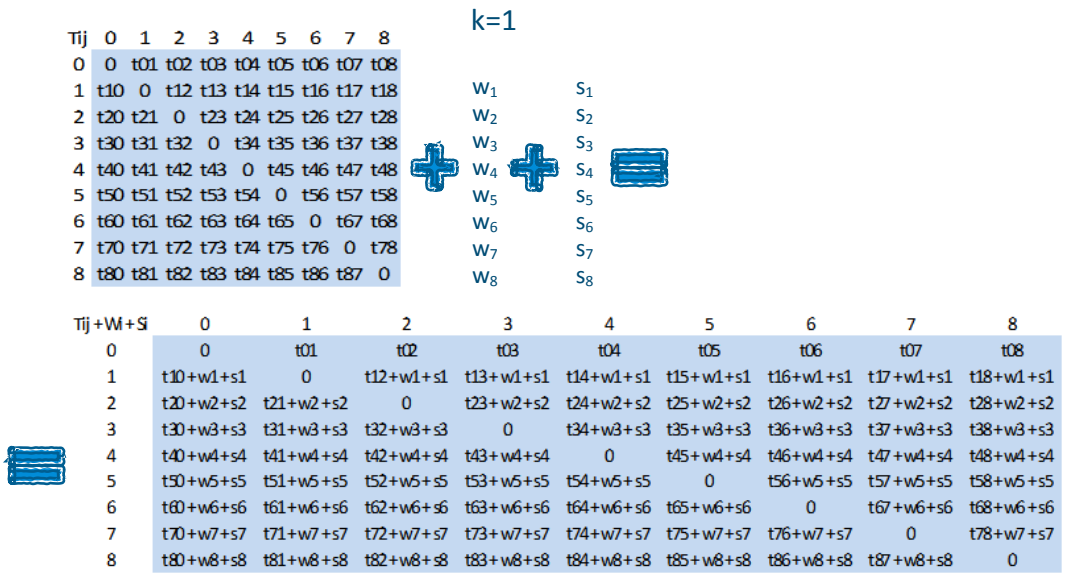


Σχήμα 4.13 Υπολογισμοί παραδείγματος του περιορισμού (4.7)

- η συνολική διάρκεια της διαδρομής του κάθε οχήματος δεν πρέπει να ξεπερνά την αργότερη δυνατή άφιξη οχήματος στην κεντρική αποθήκη

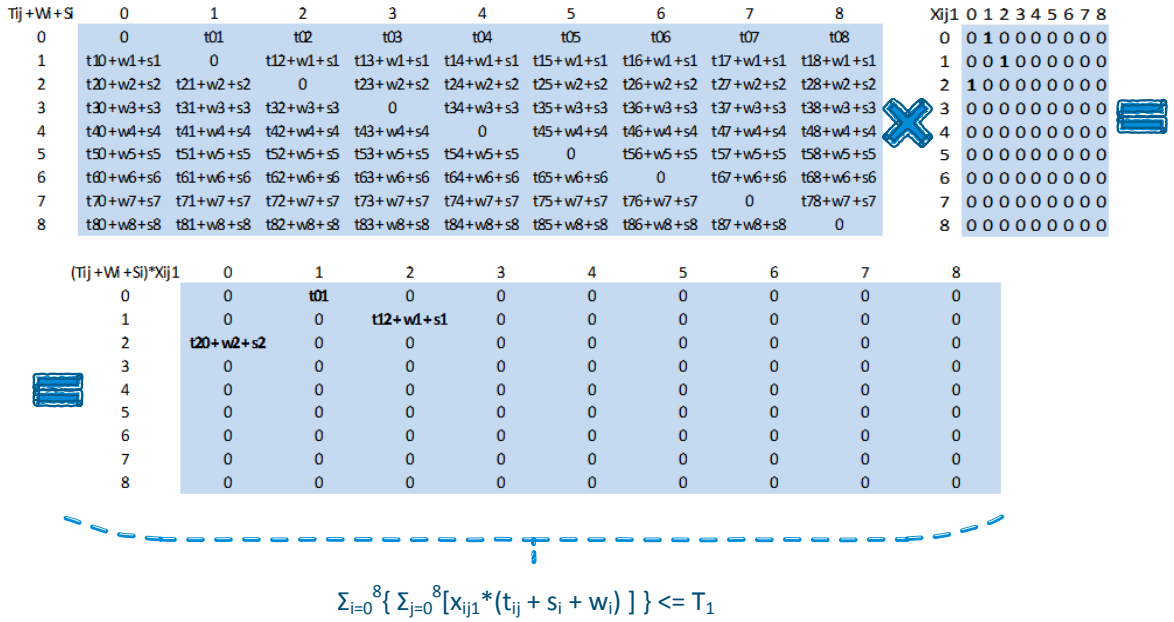
$$\sum_{i=0}^{N} \sum_{j=0}^{N} x_{ijk}\left(t_{ij} + s_i + w_i\right) \leq T_k, \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ k \in \{1, \dots, K\} \qquad \textbf{(4.8)}$$

Αρχικά, προστίθενται οι χρόνοι εξυπηρέτησης και ο χρόνος παραμονής του πρώτου οχήματος στον εκάστοτε πελάτη i, σε όλα τα στοιχεία της αντίστοιχης σειράς του πίνακα των μεταβλητών t_{ij}.



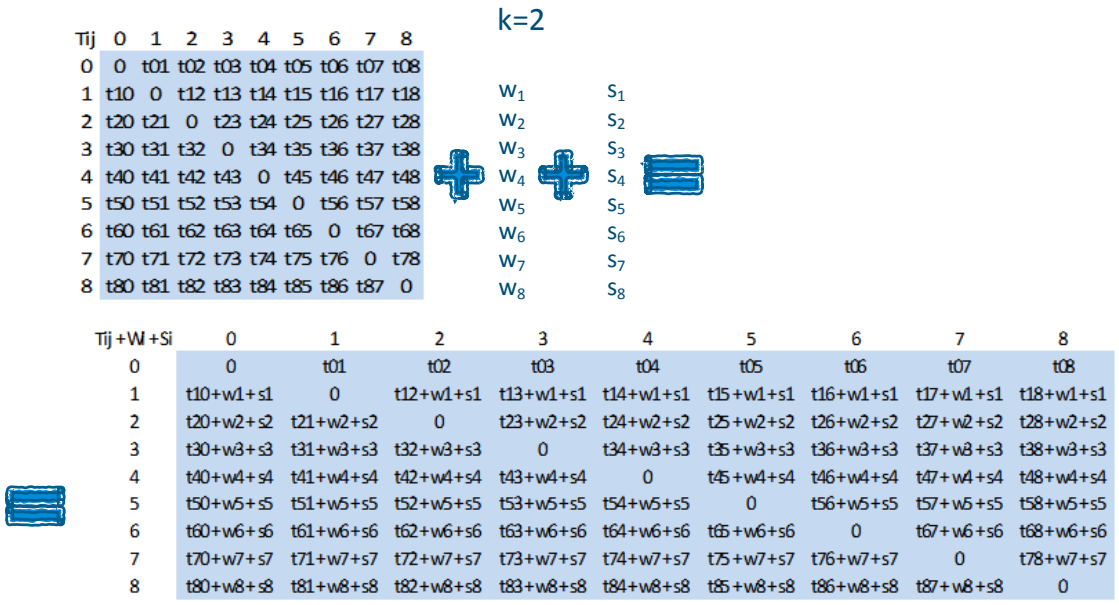
Σχήμα 4.14 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα α'

Ο πίνακας που προκύπτει πολλαπλασιάζεται κατά στοιχείο με τον πίνακα των μεταβλητών απόφασης x_{ijk} του πρώτου οχήματος και περιορίζεται το άθροισμα όλων των στοιχείων του πίνακα που προκύπτει.

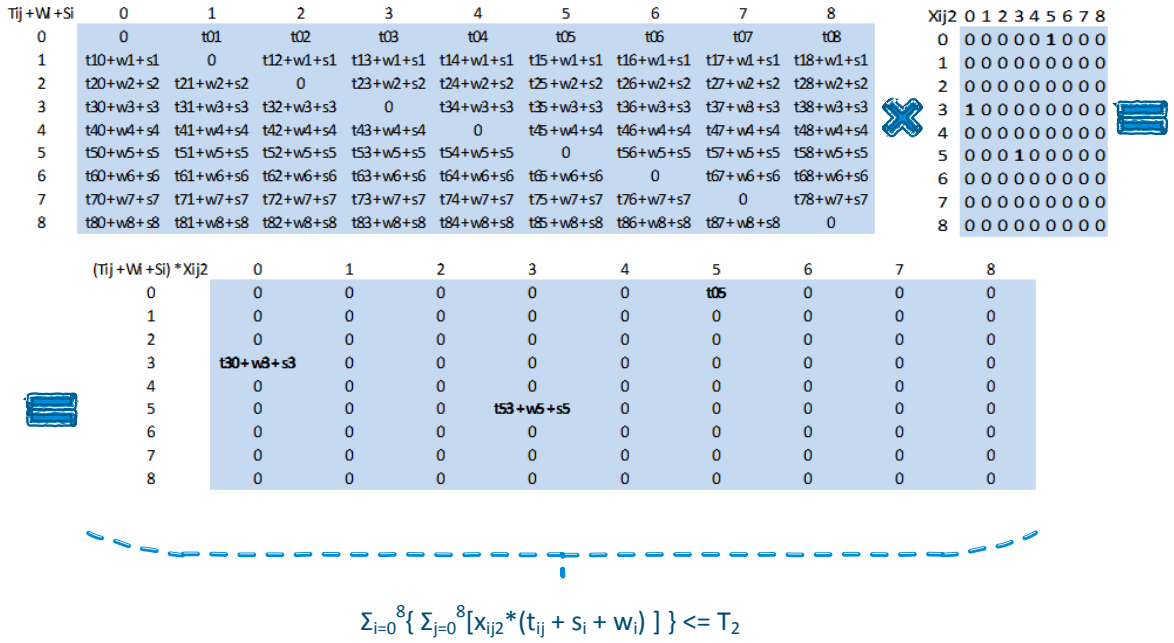


Σχήμα 4.15 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα β'

Παρουσιάζεται η αντίστοιχη διαδικασία για το δεύτερο όχημα.



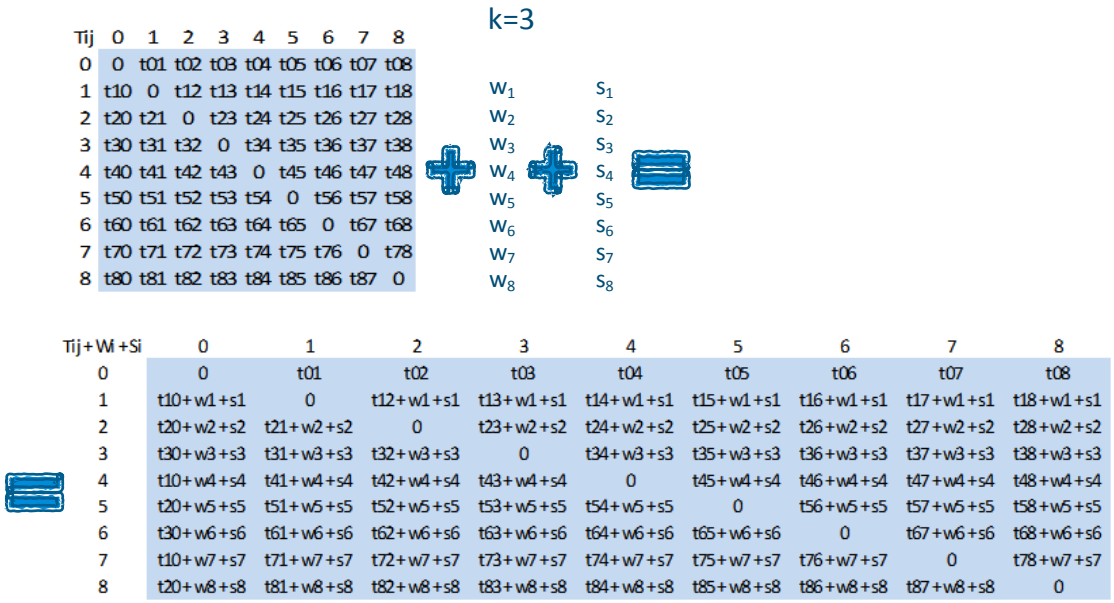
Σχήμα 4.16 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα γ'



$$\Sigma_{i=0}^{8}\{\ \Sigma_{j=0}^{8}[x_{ij2}*(t_{ij} + s_i + w_i)\ ]\ \} <= T_2$$

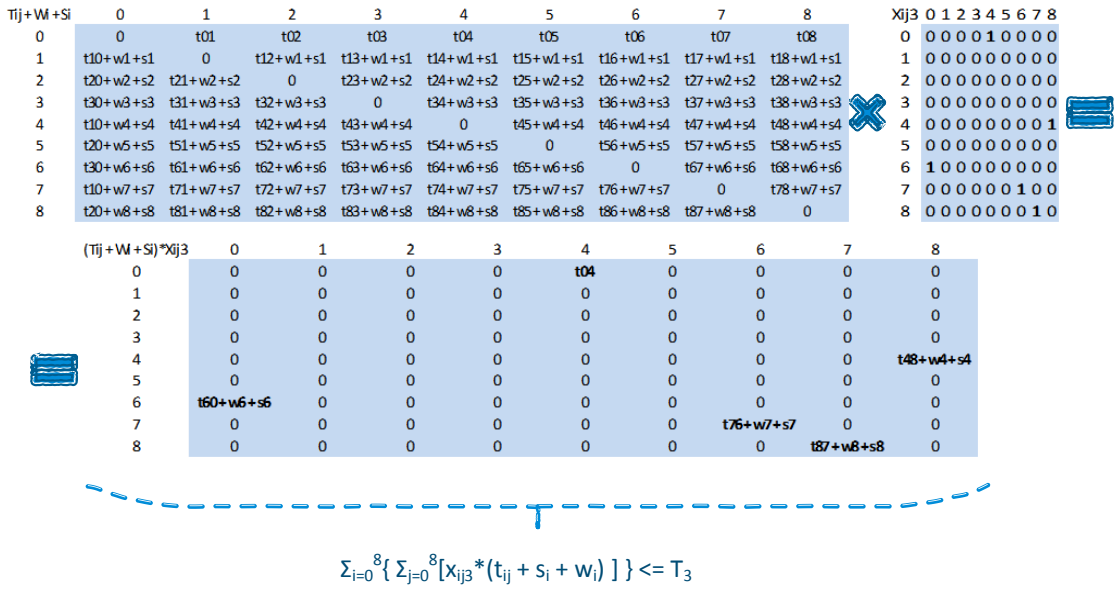
Σχήμα 4.17 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα δ'

Παρουσιάζεται η αντίστοιχη διαδικασία για το τρίτο όχημα.



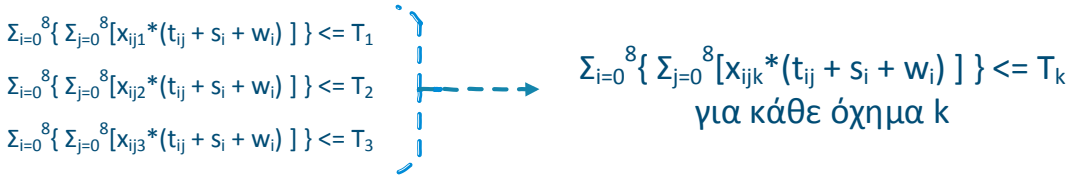
Tij+Wi+Si	0	1	2	3	4	5	6	7	8
0	0	t01	t02	t03	t04	t05	t06	t07	t08
1	t10+w1+s1	0	t12+w1+s1	t13+w1+s1	t14+w1+s1	t15+w1+s1	t16+w1+s1	t17+w1+s1	t18+w1+s1
2	t20+w2+s2	t21+w2+s2	0	t23+w2+s2	t24+w2+s2	t25+w2+s2	t26+w2+s2	t27+w2+s2	t28+w2+s2
3	t30+w3+s3	t31+w3+s3	t32+w3+s3	0	t34+w3+s3	t35+w3+s3	t36+w3+s3	t37+w3+s3	t38+w3+s3
4	t10+w4+s4	t41+w4+s4	t42+w4+s4	t43+w4+s4	0	t45+w4+s4	t46+w4+s4	t47+w4+s4	t48+w4+s4
5	t20+w5+s5	t51+w5+s5	t52+w5+s5	t53+w5+s5	t54+w5+s5	0	t56+w5+s5	t57+w5+s5	t58+w5+s5
6	t30+w6+s6	t61+w6+s6	t62+w6+s6	t63+w6+s6	t64+w6+s6	t65+w6+s6	0	t67+w6+s6	t68+w6+s6
7	t10+w7+s7	t71+w7+s7	t72+w7+s7	t73+w7+s7	t74+w7+s7	t75+w7+s7	t76+w7+s7	0	t78+w7+s7
8	t20+w8+s8	t81+w8+s8	t82+w8+s8	t83+w8+s8	t84+w8+s8	t85+w8+s8	t86+w8+s8	t87+w8+s8	0

Σχήμα 4.18 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα ε'



Σχήμα 4.19 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα στ'

Τέλος, συνοψίζονται οι περιορισμοί για το κάθε όχημα.

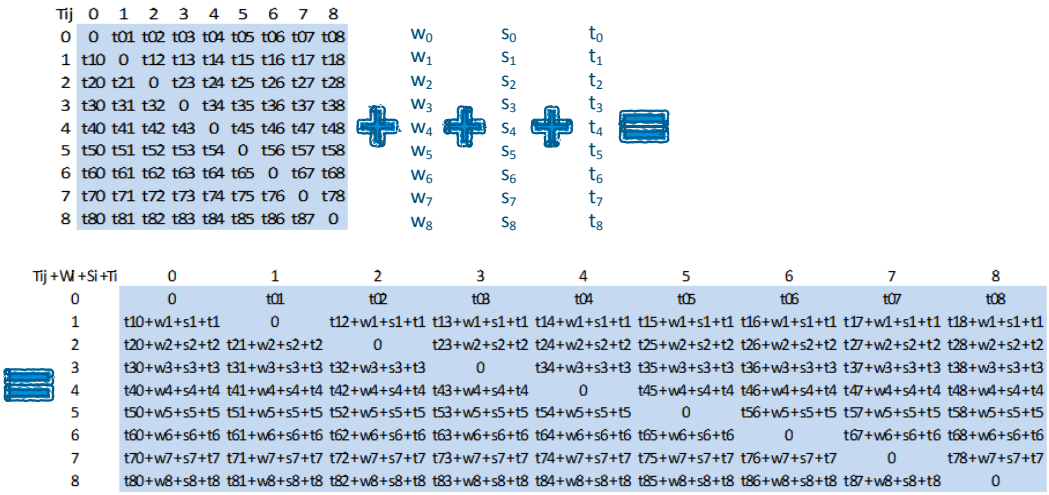


Σχήμα 4.20 Υπολογισμοί παραδείγματος του περιορισμού (4.8) βήμα ζ'

- πρέπει να οριστεί η μεταβλητή απόφασης t_i που αναπαριστά την χρονική στιγμή στην οποία εξυπηρετείται ο κάθε πελάτης i

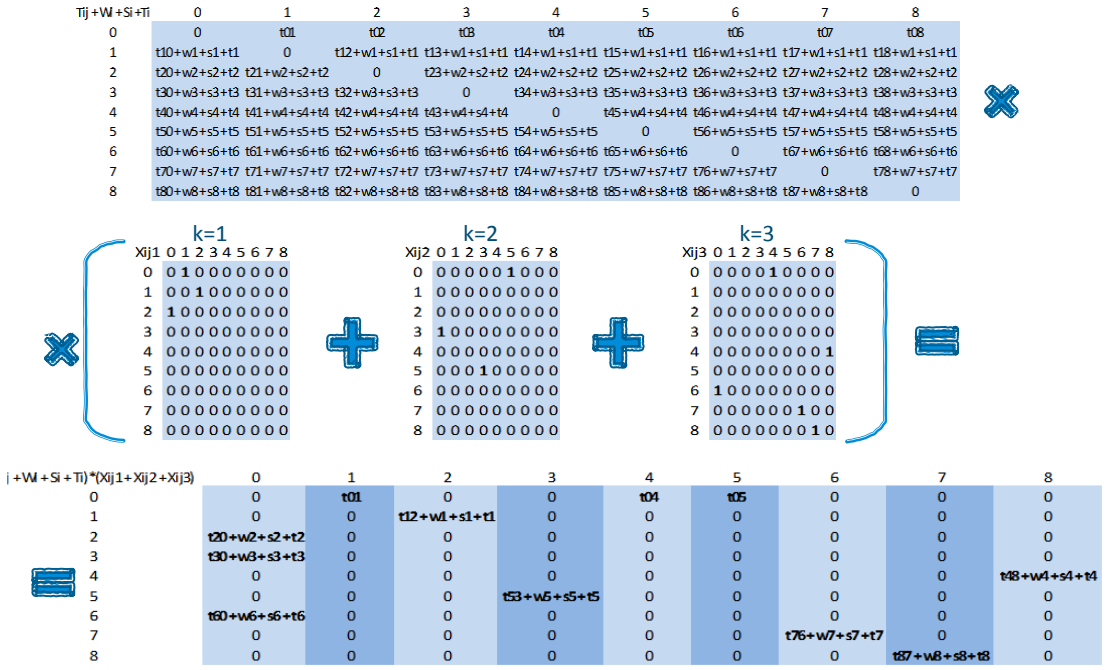
$$\sum_{k=1}^{K}\sum_{i=0}^{N} x_{ihk}(t_i + t_{ih} + s_i + w_i) = t_h, \quad \text{για κάθε } h \in \{1, \dots, N\} \tag{4.9}$$

Αρχικά προσθέτονται κατά σειρά οι πίνακες των μεταβλητών t_i, t_{ih}, s_i και w_i.



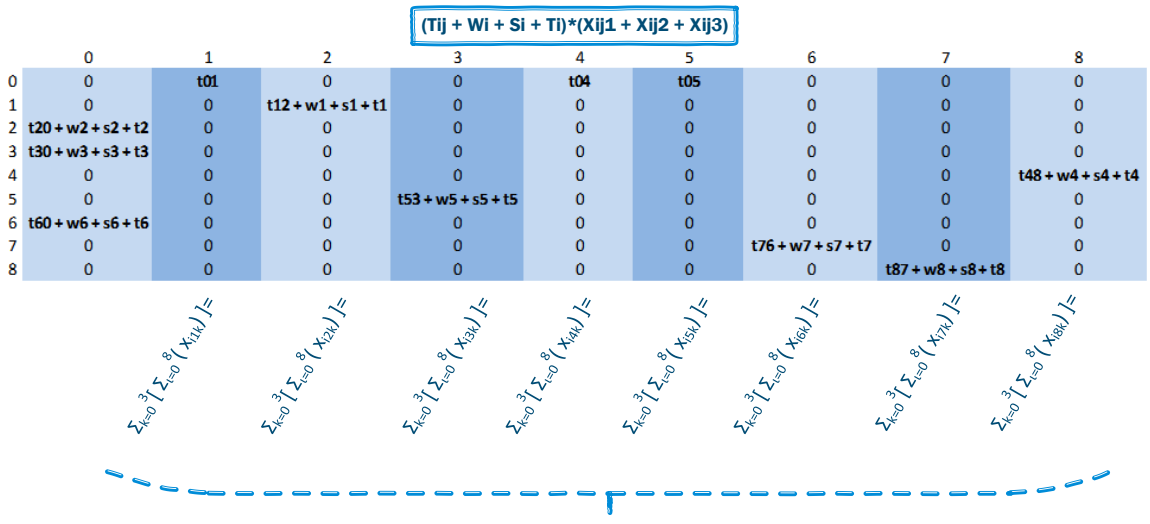
Σχήμα 4.21 Υπολογισμοί παραδείγματος του περιορισμού (4.9) βήμα α'

Στη συνέχεια, ο πίνακας που προκύπτει πολλαπλασιάζεται με το άθροισμα κατά στοιχείο των πινάκων των μεταβλητών x_{ihk} του κάθε οχήματος, ώστε να αφαιρεθούν οι χρόνοι των διαδρομών που δε χρησιμοποιούνται από τα οχήματα.



Σχήμα 4.22 Υπολογισμοί παραδείγματος του περιορισμού (4.9) βήμα β'

Το άθροισμα της κάθε στήλης του πίνακα που προκύπτει αναπαριστά τη στιγμή που κάποιο όχημα καταφθάνει στον αντίστοιχο πελάτη t_h. Παρουσιάζονται συνοπτικά οι εξισώσεις των περιορισμών.



$$\Sigma_{k=1}^{3}\{\ \Sigma_{i=0}^{8}[x_{ihk}*(t_i + t_{ih} + s_i + w_i)\ ]\ \} <= t_h$$

για κάθε πελάτη h

Σχήμα 4.23 Υπολογισμοί παραδείγματος του περιορισμού (4.9) βήμα γ'

- η χρονική στιγμή που εξυπηρετείται ο κάθε πελάτης h πρέπει να είναι εντός του χρονικού παραθύρου του εκάστοτε πελάτη h

$$e_h \leq (t_h + w_h) \leq l_h, για\ κάθε\ h \in \{0, \dots, N\} \quad \textbf{(4.10)}$$

4.4 Μαθηματική Μοντελοποίηση

$$min\left\{\sum_{k=1}^{K}\sum_{j=1}^{N} x_{hjk}\right\}, για\ h = 0 \tag{4.11}$$

$$min\left\{\sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} c_{ij}x_{ijk}\right\} \tag{4.12}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{N} x_{hjk} \leq K, για\ h = 0 \tag{4.13}$$

$$\sum_{i=1}^{N} x_{ihk} - \sum_{j=1}^{N} x_{hjk} = 0\ , για\ κάθε\ h \in \{0, \dots, N\} \tag{4.14}$$

$$και\ για\ κάθε\ k \in \{1, \dots, K\}$$

$$\sum_{k=1}^{K}\sum_{j=0}^{N} x_{hjk} = 1\ , για\ κάθε\ h \in \{1, \dots, N\} \tag{4.15}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{N} x_{ihk} = 1\ , για\ κάθε\ h \in \{1, \dots, N\} \tag{4.16}$$

$$\sum_{i=1}^{N} q_i \sum_{j=0}^{N} x_{ijk} \leq Q_k, για\ κάθε\ k \in \{1, \dots, K\} \tag{4.17}$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} x_{ijk}\left(t_{ij} + s_i + w_i\right) \leq T_k, για\ κάθε\ k \in \{1, \dots, K\} \tag{4.18}$$

$$t_0 = w_0 = s_0 = 0 \tag{4.19}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{N} x_{ihk}(t_i + t_{ih} + s_i + w_i) = t_h, για\ κάθε\ h \in \{1, \dots, N\} \tag{4.20}$$

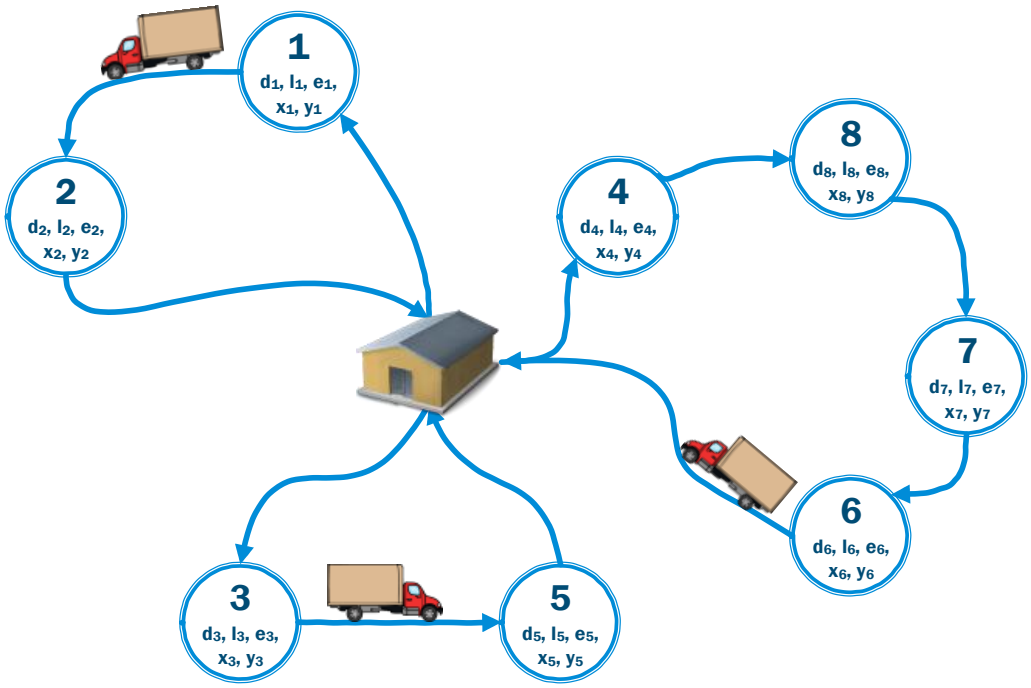
$$e_h \leq (t_h + w_h) \leq l_h, για\ κάθε\ h \in \{0, \dots, N\} \tag{4.21}$$

Οι εξισώσεις (4.11) και (4.12) αποτελούν τις αντικειμενικές συναρτήσεις και οι εξισώσεις (4.13) - (4.21) αποτελούν τους περιορισμούς του προβλήματος. Η αντικειμενική συνάρτηση (4.11) ορίζει πως ο πρωτεύων στόχος βελτιστοποίησης είναι η ελαχιστοποίηση του αριθμού των χρησιμοποιούμενων οχημάτων. Η αντικειμενική συνάρτηση (4.12) ορίζει πως ο δευτερεύον στόχος βελτιστοποίησης είναι η ελαχιστοποίηση της συνολικής διανυόμενης απόστασης των οχημάτων. Ο περιορισμός (4.13) ορίζει πως πρέπει να αναχωρούν το πολύ K οχήματα από την κεντρική αποθήκη. Ο περιορισμός (4.14) ορίζει πως αν ένα όχημα

επισκεφτεί οποιονδήποτε κόμβο (πελάτες ή αποθήκη), τότε θα αναχωρήσει από αυτόν προς οποιονδήποτε προορισμό. Οι περιορισμοί (4.15) και (4.16) ορίζουν πως κάθε πελάτης πρέπει να εξυπηρετηθεί από ένα και μόνο όχημα. Ο περιορισμός (4.17) ορίζει πως το φορτίο του κάθε οχήματος δεν ξεπερνά τη χωρητικότητά του. Ο περιορισμός (4.18) ορίζει το χρονικό όριο επιστροφής του κάθε οχήματος στην κεντρική αποθήκη. Οι περιορισμοί (4.19), (4.20) και (4.21) ορίζουν τα παράθυρα χρόνου. Κάθε λύση που ικανοποιεί τους παραπάνω περιορισμούς αποτελεί λύση για το συγκεκριμένο VRPTW.

5 Προτεινόμενη Μέθοδος Επίλυσης

5.1 Περιγραφή του συστήματος



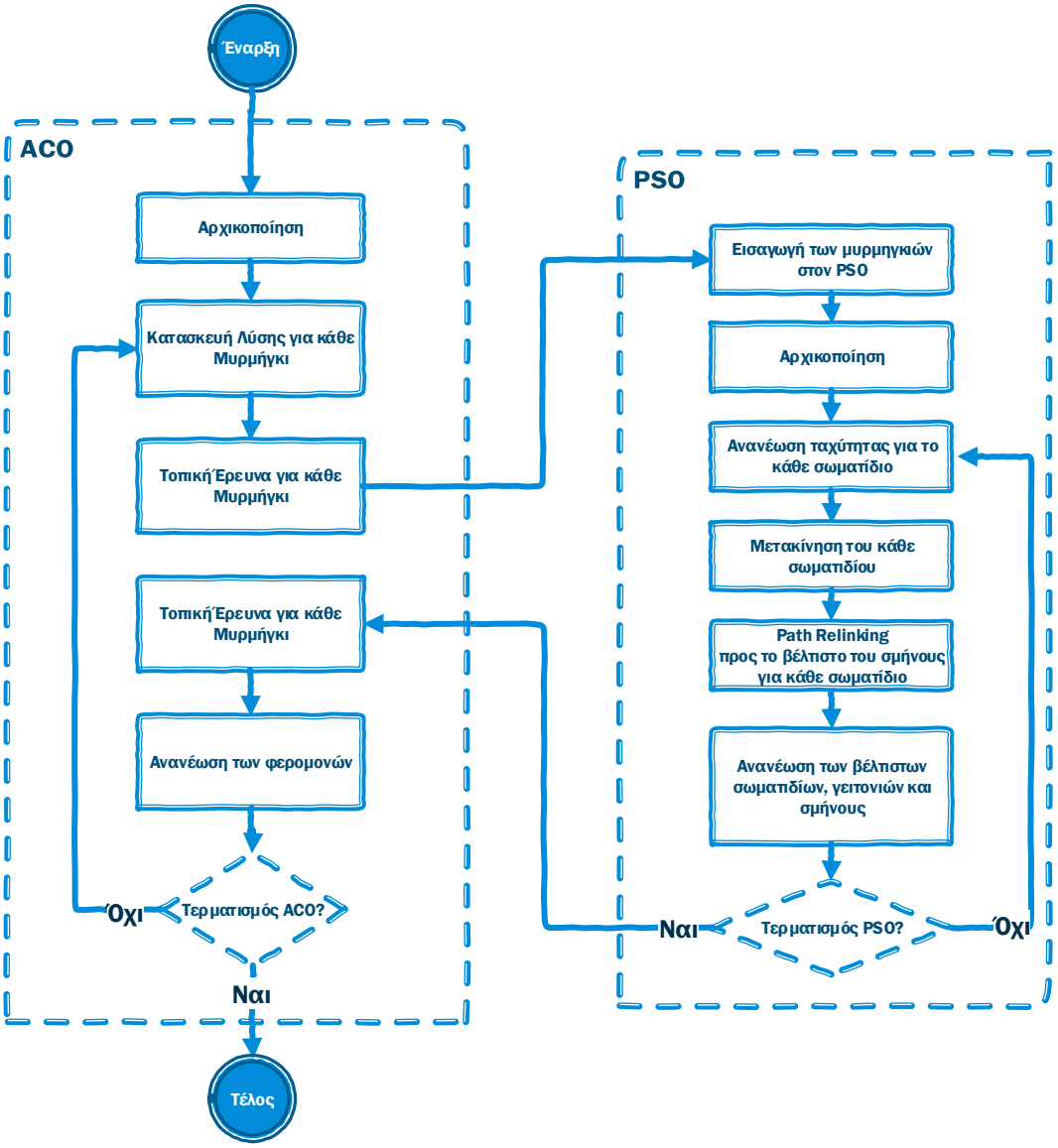
Αναπτύσσεται σύστημα επίλυσης προβλήματος δρομολόγησης οχημάτων προς εξυπηρέτηση ενός συνόλου πελατών με παράθυρα χρόνου. Το σύστημα έχει ως

- Είσοδο: τα χαρακτηριστικά των πελατών και των οχημάτων είτε από το χρήστη μηχανής είτε ως αρχείο της μορφής που έχουν τα πρότυπα προβλήματα δρομολόγησης οχημάτων με παράθυρα χρόνου του Marius M. Solomon (http://w.cba.neu.edu/~msolomon/problems.htm)
- Έξοδο: ένα αρχείο excel που περιλαμβάνει τα δρομολόγια των οχημάτων, δηλαδή το ποια οχήματα εξυπηρετούν ποιούς πελάτες καθώς και με ποιά σειρά, καθώς και γραφική αναπαράσταση της λύσης

5.2 Προτεινόμενος Αλγόριθμος

Ο βασικός άξονας που κινείται ο προτεινόμενος υβριδικός αλγόριθμος είναι η εμφωλευμένη χρήση του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO) από τον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών (ACO). Συνοπτικά, ο προτεινόμενος υβριδικός αλγόριθμος ξεκινά έναν αλγόριθμο ACO, ο οποίος σε κάθε του επανάληψη, μετά την κατασκευή των λύσεων για κάθε μυρμήγκι, μετατρέπει τον τρέχον πληθυσμό των μυρμηγκιών ως πληθυσμό σωματιδίων και καλεί τον αλγόριθμο PSO. Μετά το πέρας της εκτέλεσης του αλγορίθμου PSO, εφαρμόζεται αντίστροφη μετατροπή στον πληθυσμό των σωματιδίων σε μυρμήγκια, και συνεχίζεται η λειτουργία του αλγορίθμου ACO.

5.2.1 Διάγραμμα ροής του προτεινόμενου αλγορίθμου



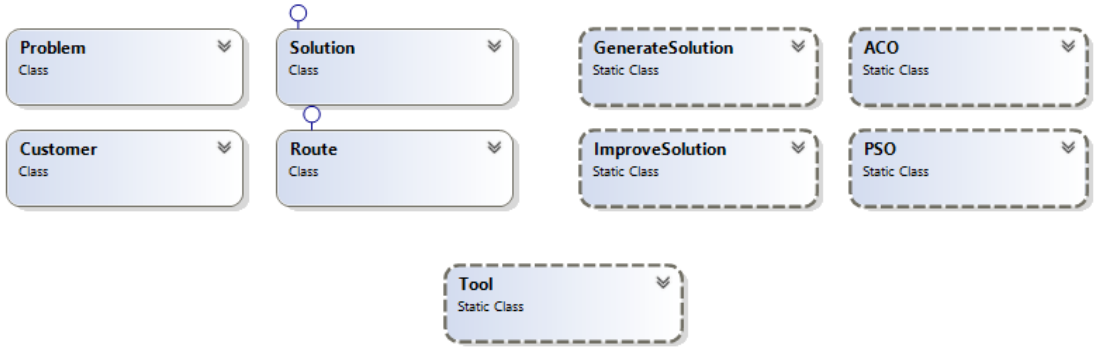
Σχήμα 5.1 : Διάγραμμα Ροής του προτεινόμενου αλγορίθμου

- Αρχικά, γίνεται αρχικοποίηση των δεδομένων του προβλήματος δηλαδή δημιουργούνται οι πίνακες των μεταβλητών που περιγράφουν το πρόβλημα προς επίλυση, και των παραμέτρων του προσαρμοσμένου αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών (ACO).
- Κατασκευάζεται μία λύση του προβλήματος για κάθε μυρμήγκι σύμφωνα με έναν από τους διαθέσιμους κατασκευαστικούς αλγορίθμους για το πρόβλημα δρομολόγησης οχημάτων με παράθυρα χρόνου και εκτελείται μία από τις διαθέσιμες μεθόδους τοπικής αναζήτησης για την λύση του κάθε μυρμηγκιού, ώστε να βρεθεί το τοπικό βέλτιστο.
- Ο πληθυσμός των μυρμηγκιών εισάγεται στον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων, κατά την αρχικοποίηση του οποίου, η λύση του προβλήματος κάθε μυρμηγκιού κωδικοποιείται ως ένα πολυδιάστατο διάνυσμα, ώστε τα διανύσματα αυτά να χρησιμοποιηθούν ως σωματίδια στον PSO, ενώ παράγονται τυχαίες ταχύτητες για την κάθε διάσταση του κάθε σωματιδίου. Εντοπίζεται το βέλτιστο σωματίδιο του πληθυσμού.
- Ανανεώνονται οι ταχύτητες ώστε τα σωματίδια να τείνουν προς το μέχρι στιγμής ολικά βέλτιστο σωματίδιο.
- Εκτελείται μετατόπιση των σωματιδίων ανάλογα με την ταχύτητά τους
- Εκτελείται η μέθοδος Path Relinking για κάθε σωματίδιο με λύση-στόχο αυτήν του βέλτιστου σωματιδίου. Σε περίπτωση εύρεσης καλύτερης θέσης από τη μέχρι στιγμής βέλτιστη του σωματιδίου, η μέθοδος σταματά και το σωματίδιο παίρνει τη βελτιωμένη αυτή θέση.
- Εντοπίζεται το μέχρι στιγμής βέλτιστο του σμήνους, και αν είναι καλύτερο από το ολικό βέλτιστο, αυτό ανανεώνεται
- Ελέγχονται οι συνθήκες τερματισμού του αλγορίθμου PSO. Στην περίπτωση που δεν ικανοποιούνται ο αλγόριθμος επιστρέφει στο βήμα ανανέωσης των ταχυτήτων των σωματιδίων, ενώ στην περίπτωση που ικανοποιούνται οι συνθήκες τερματισμού του αλγορίθμου PSO, τα πολυδιάστατα διανύσματα που αποτελούν τον πληθυσμό σωματιδίων αποκωδικοποιείται σε λύσεις του προβλήματος VRPTW και επιστρέφονται στον αλγόριθμο ACO ως πληθυσμός μυρμηγκιών.
- Εκτελείται μία από τις διαθέσιμες μεθόδους τοπικής αναζήτησης για την λύση κάθε μυρμηγκιού, ώστε να βρεθεί το τοπικό βέλτιστο για κάθε ένα από αυτά και ανανεώνονται οι φερομόνες ανάλογα με την ποιότητα της λύσης του κάθε μυρμηγκιού του πληθυσμού και την παράμετρο εξάτμισης των φερομονών.
- Τέλος, ελέγχονται οι συνθήκες τερματισμού του αλγορίθμου ACO. Στην περίπτωση που δεν ικανοποιούνται, ο αλγόριθμος κατασκευάζει νέες λύσεις του προβλήματος για κάθε μυρμήγκι και συνεχίζει τις επαναλήψεις, ενώ σε διαφορετική περίπτωση τερματίζει τη λειτουργία του.

5.2.2 Δομικά στοιχεία του προτεινόμενου αλγορίθμου

Πρώτο βήμα είναι η ανάλυση του συστήματος προς υλοποίηση, δηλαδή καθορίστηκαν τι κλάσεις θα χρησιμοποιηθούν καθώς και τα χαρακτηριστικά της κάθε μίας. Υλοποιήθηκαν οι

συναρτήσεις για την εισαγωγή δεδομένων του προβλήματος στο σύστημα και μια αρχική μορφή του αλγόριθμου βελτιστοποίησης αποικίας μυρμηγκιών στην C#, έτσι είχαμε μια πρώτη επίλυση του προβλήματος VRPTW. Να σημειωθεί πως σε αυτό το σημείο μας ενδιέφερε μια πρώτη απλή προσέγγιση του προβλήματος και η σωστή κατάστρωση των αντικειμένων, των εισόδων και των εξόδων που θα χρησιμοποιηθούν. Παρουσιάζονται σχηματικά οι κλάσεις που χρησιμοποιήθηκαν.



Σχήμα 5.2 : Σχηματική απεικόνιση των κλάσεων που χρησιμοποιήθηκαν

- Η κλάση Problem περιλαμβάνει τα δεδομένα του προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW), όπως τον πίνακα αποστάσεων και χρόνου μετάβασης από και προς τον κάθε πελάτη και την κεντρική αποθήκη, το πλήθος και τη χωρητικότητα των οχημάτων και το σύνολο των πελατών και των χαρακτηριστικών τους ως λίστα είδους Customer.
- Η κλάση Customer αναπαριστά έναν πελάτη ή την κεντρική αποθήκη και περιλαμβάνει τα χαρακτηριστικά του/της, όπως τις συντεταγμένες του, τη ζήτησή του, τον απαιτούμενο χρόνο για την εξυπηρέτησή του και τα παράθυρα χρόνου που τον διέπουν.
- Η κλάση Solution αποτελεί αναπαράσταση μιας λύσης του προβλήματος, συνεπώς περιλαμβάνει ένα σύνολο από διαδρομές, ως λίστα είδους Route, κάθε μία από τις οποίες θα εκτελέσει ένα όχημα.
- Η κλάση Route αποτελεί αναπαράσταση μιας διαδρομής που θα εκτελέσει κάποιο όχημα και περιλαμβάνει το σύνολο από τους πελάτες που θα εξυπηρετηθούν καθώς και τη σειρά προτεραιότητας που θα ακολουθηθεί.
- Η κλάση GenerateSolution περιλαμβάνει τις σειριακές και παράλληλες κατασκευαστικές μεθόδους που δημιουργούν λύσεις για το εκάστοτε πρόβλημα VRPTW. Οι μέθοδοι αυτές παίρνουν ως είσοδο τα χαρακτηριστικά του προβλήματος και ένα σύνολο παραμέτρων και επιστρέφουν μία λύση του προβλήματος σε μορφή κλάσης Solution.
- Η κλάση ImproveSolution περιλαμβάνει τις μεθόδους βελτίωσης λύσης (μέθοδοι τοπικής αναζήτησης), τύπου έσω-διαδρομής και έξω-διαδρομής, για το εκάστοτε πρόβλημα VRPTW. Οι μέθοδοι αυτές παίρνουν ως είσοδο ένα σύνολο παραμέτρων, τα χαρακτηριστικά του προβλήματος και μία λύση του και επιστρέφουν μία λύση του προβλήματος σε μορφή κλάσης Solution που είναι η ίδια, ή πιο βέλτιστη από την αρχική λύση που δόθηκε ως είσοδος.

- Οι κλάσεις ACO και PSO περιλαμβάνουν τις μεθόδους των δύο μεταευρετικών αλγορίθμων Προσομοίωσης Αποικίας Μυρμηγκιών και Βελτιστοποίησης Σμήνους Σωματιδίων αντίστοιχα. Δέχονται ως είσοδο ένα σύνολο παραμέτρων και ένα πρόβλημα προς βελτιστοποίηση και επιστρέφουν μία λύση σε μορφή κλάσης Solution.
- Η κλάση Tool περιλαμβάνει ποικίλα βοηθητικά εργαλεία που αναπτύχθηκαν κατά τη διάρκεια υλοποίησης του αλγορίθμου. Περιλαμβάνει μεθόδους διαχείρισης εισόδων και προβολής εξόδων του λογισμικού, διαχείρισης δυσδιάστατων και τρισδιάστατων πινάκων, εισαγωγής δεδομένων και εξαγωγής στατιστικών χρήσης των αλγορίθμων, καθώς και μεθόδους για τη δοκιμή των αλγορίθμων που υλοποιήθηκαν.

5.3 Υλοποίηση προτεινόμενου αλγορίθμου

5.3.1 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.

5.3.2 Βασικοί Παράμετροι προτεινόμενου αλγορίθμου

Παρουσιάζονται οι βασικές παράμετροι του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών (ACO) και του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO) οι οποίες προσδιορίζουν τα στοιχεία του προτεινόμενου υβριδικού αλγορίθμου επηρεάζοντας σημαντικά την ταχύτητα σύγκλισής του, το χρόνο εκτέλεσης καθώς και την ποιότητα της λύσης που επιστρέφει.

5.3.2.1 ACO

Παρουσιάζονται οι έξι βασικές παράμετροι του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών (ACO). Οι παράμετροι α και β αντιπροσοπεύουν την επιρροή των φερομονών και της ορατότητας μεταξύ των πελατών στην κατασκευή λύσης από τις κατασκευαστικές μεθόδους, συγκεκριμένα είναι η δύναμη στην οποία υψώνονται οι φερομόνες και η ορατότητα μεταξύ των πελατών αντίστοιχα. Η παράμετρος ρ παίρνει τιμή μεταξύ του 0 και του 1 και πολλαπλασιάζεται με την τιμή των φερομονών σε κάθε επανάληψη του αλγορίθμου ACO, ώστε να τις μειώνονται ποσοστιαία και να προσομοιώνεται η εξάτμιση των φερομονών που συμβαίνει στη φύση. Η παράμετρος a/c αναπαριστά τον λόγο του πλήθους των μυρμηγκιών προς το πλήθος των πελατών και χρησιμοποιείται για τον προσδιορισμό του

αριθμού των μυρμηγκιών που δημιουργούνται ανάλογα με τον αριθμό των πελατών του εκάστοτε προβλήματος. Η παράμετρος Q αντιπροσωπεύει την σταθερή ποσότητα φερομονών που προσθέτονται σε κάθε ανανέωση των φερομονών από την αντίστοιχη μέθοδο. Η παράμετρος EA αντιπροσωπεύει τον αριθμό των επαναλήψεων που εκτελούνται μέχρι το πέρας των υπολογισμών.

Πίνακας 5.1 Παράμετροι Αλγορίθμου Βελτιστοποίησης Αποικίας Μυρμηγκιών

a	δύναμη στην οποία υψώνονται οι φερομόνες
b	δύναμη στην οποία υψώνεται η ορατότητα μεταξύ των πελατών
ρ	πολλαπλασιαστής μείωσης των φερομονών λόγω εξάτμισης
a/c	ο λόγος πλήθους μυρμηγκιών προς πλήθος πελατών
Q	σταθερός αριθμός που προστίθεται σε κάθε ανανέωση των φερομονών
EA	αριθμός επαναλήψεων του αλγορίθμου ACO

5.3.2.2 PSO

Παρουσιάζονται οι πέντε βασικές παράμετροι του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO). Η παράμετρος w ονομάζεται αδρανειακό βάρος των σωματιδίων (inertia weight) και αντιπροσωπεύει τη διατήρηση της ταχύτητας των σωματιδίων, καθώς κατά την ανανέωση των ταχυτήτων των σωματιδίων, η ταχύτητά τους εξαρτάται από το γινόμενο της ταχύτητας του σωματιδίου κατά την προηγούμενη επανάληψη με την παράμετρο w. Η παράμετρος α_1 αντιπροσωπεύει την τάση σύγκλισης του κάθε σωματιδίου από τη βέλτιστη θέση που έχει περάσει το ίδιο το σωματίδιο. Η παράμετρος α_2 αντιπροσωπεύει την τάση σύγκλισης του κάθε σωματιδίου προς το ολικό βέλτιστο, δηλαδή τη βέλτιστη θέση που έχει περάσει οποιοδήποτε σωματίδιο του σμήνους σωματιδίων. Η παράμετρος $A\Sigma$ αντιπροσωπεύει το πλήθος των σωματιδίων που δημιουργεί ο αλγόριθμος PSO και η παράμετρος $E\Pi$ αντιπροσωπεύει τον αριθμό των επαναλήψεων του αλγορίθμου PSO που εκτελούνται σε κάθε κλήση του, δηλαδή σε κάθε επανάληψη του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών ACO.

Πίνακας 5.2 Παράμετροι Αλγορίθμου Βελτιστοποίησης σμήνους Σωματιδίων

w	αδράνεια των σωματιδίων
a_1	παράμετρος σύγκλισης σωματιδίων προς το δικό τους βέλτιστο
α_2	παράμετρος σύγκλισης σωματιδίων προς το ολικό βέλτιστο
$A\Sigma$	πλήθος σωματιδίων
$E\Pi$	αριθμός επαναλήψεων του αλγορίθμου PSO

5.3.3 Βασικές μέθοδοι που υλοποιήθηκαν

Παρουσιάζονται οι μέθοδοι που αποτελούν τον προτεινόμενο υβριδικό αλγόριθμο. Κατά την υλοποίηση των μεθόδων που αναπτύχθηκαν δόθηκε ιδιαίτερη έμφαση στην επεκτασιμότητα του συστήματος, ώστε να δύναται η περεταίρω εξέλιξή του με την προσθήκη επιπρόσθετων μεθόδων, είτε κατασκευαστικών είτε τοπικής αναζήτησης είτε διαφορετικού τύπου.

5.3.3.1 Μέθοδοι βελτιστοποίησης αποικίας μυρμηγκιών

Παρουσιάζονται οι μέθοδοι που αποτελούν την υλοποίηση του τροποποιημένου αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών.

5.3.3.1.1 MasterACO

Η μέθοδος MasterACO συνιστά τον πρωτεύον αλγόριθμο που χρησιμοποιείται από το προτεινόμενο σύστημα και αποτελεί μία τροποποιημένη υλοποίηση του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών (ACO) για την επίλυση προβλημάτων δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) με την χρήση του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO). Δέχεται ως είσοδο τα δεδομένα που καθορίζουν το προς επίλυση πρόβλημα και ένα σύνολο παραμέτρων απαραίτητες για τον προσδιορισμό της λειτουργίας της μεθόδου. Μετά το πέρας των υπολογισμών, επιστρέφει μία λύση του δοσμένου προβλήματος VRPTW.

Η μέθοδος εκκινεί αρχικοποιώντας τις φερομόνες μέσω της μεθόδου InitializePheromones (παράγραφος 5.3.3.1.2) και δημιουργώντας τον πληθυσμό των μυρμηγκιών κατασκευάζοντας μία λύση του προβλήματος VRPTW για το καθένα μέσω της κάποιας από τις διαθέσιμες κατασκευαστικές μεθόδους. Στη συνέχεια, εκτελεί μία από τις διαθέσιμες μεθόδους τοπικής αναζήτησης σε κάθε μυρμήγκι και εισάγει τον πληθυσμό τον μυρμηγκιών στην μέθοδο βελτιστοποίησης σμήνους σωματιδίων MasterPSOforVRPTW (παράγραφος 5.3.3.4.1) όπου και μετατρέπεται σε πληθυσμό σωματιδίων. Μετά το πέρας των υπολογισμών του αλγορίθμου PSO, το ένα τρίτο του πληθυσμού σωματιδίων μετατρέπεται ξανά σε πληθυσμό μυρμηγκιών, διαγράφονται τα χειρότερα μυρμήγκια ώστε να διατηρείται ο αριθμός των μυρμηγκιών σταθερός και εκτελείται μέθοδος τοπικής αναζήτησης σε κάθε μυρμήγκι. Τέλος, ανανεώνονται οι φερομόνες του δικτύου μετάβασης του προβλήματος

VRPTW μέσω της μεθόδου PheromoneUpdate (παράγραφος 5.3.3.1.4), εκτελείται εξάτμιση των φερομονών μέσω της μεθόδου EvaporatePheromones (παράγραφος 5.3.3.1.3) και τερματίζεται η τρέχουσα επανάληψη της μεθόδου. Οι επαναλήψεις της μεθόδου συνεχίζονται μέχρι ένα προκαθορισμένο από τις παραμέτρους αριθμό ή έως ότου η τρέχουσα βέλτιστη λύση έχει αποδεκτή ποιότητα, σημείο στο οποίο η μέθοδος τερματίζει και επιστρέφει την τρέχουσα βέλτιστη λύση.

Παρουσιάζεται ο κώδικας της μεθόδου MasterACO.

```
          public   static   Solution   MasterACO(Problem   problem,ACO.Parameters   pars,Random
randomizer)
              {
                  PSO_2.Parameters psopars = new PSO_2.Parameters();
                  psopars.iterations = 0;
                  return MasterACO(problem, pars, psopars, randomizer);
              }
              public static Solution MasterACO(Problem problem,Parameters pars,PSO_2.Parameters
psopars,Random randomizer)
              {
                  //If pheromone min and max is not initialized, set max = Q/Σ(d0i) , min =
Q/Σ(2*d0i)
                  if ((pars.pheromoneMin == 0) & (pars.pheromoneMax == 0))
                  {
                      pars.pheromoneMin = Tool.GetPherMinFromQ(problem, pars);
                      pars.pheromoneMax = Tool.GetPherMaxFromQ(problem, pars);
                  }
                  //if iteration parameter is zero, make it one
                  if (pars.iterations <= 0)
                      pars.iterations = 1;

                  Stopwatch superWatch = new Stopwatch();
                  superWatch.Start();
                  int antNumber = Convert.ToInt16(problem.n*pars.antsToCustomersRatio);
                  if (antNumber < 1)
                      antNumber = 1;
                  ants = new Solution[antNumber];

                  BestFound = new Solution(problem);
                  BestFound.ttlDistance = 0;

                  InitializePheromones(problem,pars);
                  int iter;
                  int BestFoundIter = -1;
                  for (iter = 0; iter < pars.iterations; iter++)
                  {
                      //Console.WriteLine("iteration: " + iter);
                      double[,] phersOfCurrentIter = Tool.Make2D(pheromones, iter);
                      Stopwatch timeToConstruct = new Stopwatch();
                      timeToConstruct.Start();
                      for (int ant = 0; ant < ants.Length; ant++)
                      {
                          //generate ants
                          ants[ant]     =     GenerateSolution.pickerForACO(pars,     problem,
phersOfCurrentIter, randomizer);
                          if (ants[ant].isComplete == false)
                              throw new Exception("every constructor should return a valid
solution.");
                          //improve ants
                          ants[ant] = ImproveSolution.Intraroute.TwoOpt1(ants[ant], problem);
                      }
                      //PSO Diagnostics: keep ants' stats
                      double mean1 = ants.Sum(x => Tool.ObjectiveFunc(x, problem)) / ants.Length;
                      double sssbest1 = Tool.ObjectiveFunc(ants.Min(), problem);

                      //HYBRID IMPLEMENTATION:
                      //insert ants into PSO_2
                      List<Solution>   PSO_result   =   PSO_2.MasterPSOforVRPTW(problem,   ants,
psopars, randomizer);
                      //local search PSO results
                      for (int i = 0; i < PSO_result.Count(); i++)
                          PSO_result[i]                                                    =
ImproveSolution.Intraroute.TwoOpt1(PSO_result[i],problem);
                      //make a list of ants and particles
                      List<Solution> AntsNParts = new List<Solution>();
                      AntsNParts = ants.ToList();
                      AntsNParts.AddRange(PSO_result);
                      //sort hybrid swarm by objective function
                      AntsNParts.Sort((x,          y)          =>          Tool.ObjectiveFunc(x,
problem).CompareTo(Tool.ObjectiveFunc(y, problem)));
                      //keep the best solutions so that ant number stays the same
                      for (int i = 0; i < PSO_result.Count(); i++)
                          AntsNParts.Remove(AntsNParts.Last());
                      //update ant array
                      ants = AntsNParts.ToArray();

                      //improve ants
                      for (int ant = 0; ant < ants.Length; ant++)
```

```
                            ants[ant] = ImproveSolution.Intraroute.TwoOpt1(ants[ant], problem);

                        //improve and save iteration best
                        Solution CurrIterationBest = ants.Min();
                        CurrIterationBest = ImproveSolution.Intraroute.TwoOpt1(CurrIterationBest,
problem);

                        if      ((CurrIterationBest.CompareTo(BestFound)      ==      -1)      |
(BestFound.ttlDistance == 0))
                        {
                            BestFound   =   ImproveSolution.Intraroute.TwoOpt1(CurrIterationBest,
problem);
                            BestFoundIter = iter; //keep the iteration ACO found the best solution.
                        }

                        EvaporatorPicker(pars, iter);

                        UpdatorPicker(problem, pars, ants, iter);

                        //IF the bestFound is pars.stopACOifNoImprovementForIter iterations old,
                        //stop ACO (works for 100-customer problems)
                        if (
                            (
                                (problem.bestKnownDistance  *  (1.0  +  pars.stopIterationAt)  >
BestFound.ttlDistance) &
                                (problem.n == 101) &
                                (iter > 3)
                            ) |
                                (iter - BestFoundIter > pars.stopACOifNoImprovementForIter)
                            )
                        {
                            Console.WriteLine("Breaking ACO at: " + iter + " iter. (BestFound is
" + (iter - BestFoundIter) + " iters old)." );
                            pheromones = Tool.ReduceThirdDimension(pheromones, iter+1);
                            break;
                        }
                    }
                    if (iter == pars.iterations)
                        iter--;
                    superWatch.Stop();

                    //Save diagnostics:

                    BestFound.timeElapsed = superWatch.Elapsed;
                    BestFound.iterrationFound = BestFoundIter;

                    double[, ,] pherWithNoDepot = Tool.RemoveDepotFromPhers(pheromones); //remove
pheromones of arcs that include the depot
                    var pheromonesWithOutDiagonal = pherWithNoDepot.Cast<double>().Where(x => (x
> 0)); //remove the diagonal of the pheromones

                    //Stagnation:
                    BestFound.simpleStagnation = Tool.GetSimpleStagnation(ants, BestFound);

                    BestFound.pherMax = pheromonesWithOutDiagonal.Max();
                    BestFound.pherMin = pheromonesWithOutDiagonal.Min();
                    double[,] lastPhers = Tool.Make2D(pherWithNoDepot, iter);
                    BestFound.lastPherMean    =    lastPhers.Cast<double>().Where(x    =>    (x    >
0)).Average(); //remove diagonal and get average.
                    foreach (double d in lastPhers)
                    {
                        if (d <= pars.pheromoneMin*1.1)
                            BestFound.lastPhersNearMin++;
                        if (d >= pars.pheromoneMax * 0.9)
                            BestFound.lastPhersNearMax++;
                    }
                    BestFound.lastPhersNearMin     *=     1     /     (pheromones.GetLength(0)     *
pheromones.GetLength(1));
                    BestFound.lastPhersNearMax     *=     1     /     (pheromones.GetLength(0)     *
pheromones.GetLength(1));
                    BestFound.allPheromones = pheromones;
                    return BestFound;
                }
```

5.3.3.1.2 InitializePheromones

Η μέθοδος InitializePheromones δημιουργεί και αρχικοποιεί τον πίνακα των φερομονών ανάλογα με τις παραμέτρους του εκάστοτε προβλήματος.

Παρουσιάζεται ο κώδικας της μεθόδου InitializePheromones.

```
        public static void InitializePheromones(Problem problem, Parameters pars)
                {
                    pheromones = new double[problem.n, problem.n, pars.iterations];
                    for (int i = 0; i < problem.n; i++)
                        for (int j = 0; j < problem.n; j++)
                        {
                            pheromones[i, j, 0] = pars.pheromoneInit;
```

```
                            if (pheromones[i, j, 0] > pars.pheromoneMax)
                                pheromones[i, j, 0] = pars.pheromoneMax;
                            else if (pheromones[i, j, 0] < pars.pheromoneMin)
                                pheromones[i, j, 0] = pars.pheromoneMin;

                            if (i == j)
                                pheromones[i, j, 0] = 0;
                        }
                }
```

5.3.3.1.3 EvaporatePheromones

Η μέθοδος EvaporatePheromones αποτελεί το μηχανισμό εξάτμισης των φερομονών για τον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών. Δέχεται ως είσοδο τον τρέχων πίνακα φερομονών και την παράμετρο εξάτμισης και επιστρέφει τον πίνακα των φερομονών μετά την εξάτμισή τους. Η μέθοδος αυτή, καλείται αποκλειστικά από τη μέθοδο EvaporatorPicker η οποία επιλέγει μέθοδο εξάτμισης των φερομονών ανάλογα με τις παραμέτρους του προτεινόμενου αλγορίθμου, ώστε να δύναται η επέκταση του αλγορίθμου με την υλοποίηση περισσότερων μεθόδων εξάτμισης των φερομονών.

Παρουσιάζεται ο κώδικας της μεθόδου EvaporatorPicker και EvaporatePheromones.

```
        public static void EvaporatorPicker(Parameters pars, int iter)
                {
                    if (pars.pickPheromoneEvaporator == 1)
                        EvaporatePheromones(pars, iter);
                    else
                        throw new Exception(String.Format(pars.pickPheromoneEvaporator + " is not
a valid pickPheromoneEvaporator parameter."));
                }
                public static void EvaporatePheromones(Parameters pars,int iteration)
                {
                    if (iteration < pheromones.GetLength(2)-1)
                    {
                        for (int i = 0; i < pheromones.GetLength(0); i++)
                        {
                            for (int j = 0; j < pheromones.GetLength(1); j++)
                            {
                                pheromones[i, j, iteration + 1] = pheromones[i, j, iteration] * (1
- pars.pheromoneEvaporation);

                                //Pher Min Max:
                                if (pheromones[i, j, iteration + 1] > pars.pheromoneMax)
                                    pheromones[i, j, iteration + 1] = pars.pheromoneMax;
                                else if (pheromones[i, j, iteration + 1] < pars.pheromoneMin)
                                    pheromones[i, j, iteration + 1] = pars.pheromoneMin;

                                if (i == j)
                                    pheromones[i, j, iteration + 1] = 0;
                            }
                        }
                    }
                }
```

5.3.3.1.4 Pheromone Update

Η μέθοδος UpdatePheromones αποτελεί το μηχανισμό ανανέωσης των φερομονών του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών. Δέχεται ως είσοδο το προς επίλυση πρόβλημα δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW), τις τρέχουσες φερομόνες του αλγορίθμου ACO και την τρέχουσα λύση του προβλήματος για το κάθε μυρμήγκι και επιστρέφει τον ανανεωμένο πίνακα φερομονών του αλγορίθμου. Η μέθοδος αυτή καλείται από τη μέθοδο UpdatorPicker, η οποία επιλέγει μέθοδο ανανέωσης των φερομονών ανάλογα με τις παραμέτρου του αλγορίθμου, ώστε να είναι δυνατή η επέκταση του αλγορίθμου με την υλοποίηση περισσότερων μεθόδων ανανέωσης των φερομονών.

Η ανανέωση των φερομονών υλοποιείται για την λύση του κάθε μυρμηγκιού ξεχωριστά. Συγκεκριμένα, προστίθενται φερομόνες στα τόξα μετάβασης του δικτύου, ανάλογα με την ποιότητα της λύσης στην οποία χρησιμοποιούνται. Συνεπώς, προστίθενται περισσότερες

φερομόνες στα τόξα μετάβασης που χρησιμοποιούνται περισσότερο και από ποιοτικότερες λύσεις μυρμηγκιών.

Παρουσιάζεται ο κώδικας της μεθόδου UpdatePheromones και UpdatorPicker.

```
       public static void UpdatorPicker(Problem problem, Parameters pars, Solution[] ants, int
iter)
               {
                   if (pars.pickPheromoneUpdator == 1)
                       UpdatePheromones(problem, ants, pars, iter);
                   else
                       throw new Exception(String.Format(pars.pickPheromoneUpdator + " is not a
valid pickPheromoneUpdator parameter."));
               }
               public static void UpdatePheromones(Problem p, Solution[] ants, Parameters pars,
int iteration)
               {
                   //Pheromone update depending on the total travel distance of each ant.
                   if (iteration < pheromones.GetLength(2)-1)
                   {
                       foreach (Solution ant in ants)
                       {
                           foreach (Route route in ant.routes)
                           {
                               for (int i = 1; i < route.customers.Count; i++)
                               {
                                   int fromID = route.customers[i - 1].ID;
                                   int toID = route.customers[i].ID;
                                   pheromones[fromID, toID, iteration + 1] = pheromones[fromID,
toID, iteration] + pars.Q / ant.ttlDistance;

                                   //Pher Min Max:
                                   if    (pheromones[fromID,    toID,    iteration    +    1]    >
pars.pheromoneMax)
                                       pheromones[fromID,    toID,    iteration    +    1]    =
pars.pheromoneMax;
                                   else   if   (pheromones[fromID,   toID,   iteration   +   1]   <
pars.pheromoneMin)
                                       pheromones[fromID,    toID,    iteration    +    1]    =
pars.pheromoneMin;

                                   if (pars.MakePheromonesSymmetric == true)
                                       pheromones[toID,    fromID,    iteration    +    1]    =
pheromones[fromID, toID, iteration + 1];
                                   if (fromID == toID)
                                       throw new Exception("Infeasible route. Vehicle cannot go
from " + fromID + " to " + toID + " .");
                               }
                           }
                       }
                   }
               }
```

5.3.3.2 Μέθοδοι κατασκευής λύσης

Παρουσιάζονται οι σειριακές και οι παράλληλες κατασκευαστικές μέθοδοι που χρησιμοποιήθηκαν. Στόχος των κατασκευαστικών μεθόδων είναι να δημιουργήσουν μία λύση ενός προβλήματος VRPTW σύμφωνα με προκαθορισμένες παραμέτρους. Συνεπώς, παίρνουν ως είσοδο τα χαρακτηριστικά ενός προβλήματος VRPTW και τον τρέχων πίνακα φερομονών του αλγορίθμου προσομοίωσης αποικίας μυρμηγκιών και επιστρέφουν μία λύση του προβλήματος που περιλαμβάνει το πλήθος των χρησιμοποιούμενων οχημάτων και το δρομολόγιο του καθενός.

5.3.3.2.1 G1_NNAACO

Η μέθοδος αυτή είναι μια εκδοχή της σειριακής κατασκευαστικής μεθόδου του Πλησιέστερου Γείτονα προσαρμοσμένη για να χρησιμοποιηθεί στον αλγόριθμο προσομοίωσης αποικίας μυρμηγκιών. Συγκεκριμένα, αντί το όχημα να επιλέγει τον επόμενο πελάτη προς εξυπηρέτηση με άξονα την μικρότερη απόσταση ντετερμινιστικά, επιλέγει σύμφωνα με ένα μηχανισμό επιλογής ανάλογα με την καταλληλόλητά του (Fitness Proportionate Selection). Συγκεκριμένα, η καταλληλόλητα των πελατών είναι το γινόμενο του

αντιστρόφου της απόστασης από την τρέχουσα θέση του οχήματος με το ποσό των φερομονών που βρίσκονται στη διαδρομή από την τρέχουσα θέση του οχήματος μέχρι την θέση του κάθε πελάτη.

Έστω ότι το όχημα βρίσκεται στην κεντρική αποθήκη (Πελάτης 0), η απόσταση από την κεντρική αποθήκη προς κάποιον πελάτη j συμβολίζεται ως d_{0j} και το ποσό της φερομόνης της συγκεκριμένης διαδρομής συμβολίζεται ως τ_{0j}. Παρουσιάζεται η καταλληλότητα της μετάβασης του οχήματος που βρίσκεται στην κεντρική αποθήκη στον πελάτη j.

$$\frac{1}{d_{0j}} \times t_{0j} \tag{5.1}$$

Παρουσιάζεται ο κώδικας της μεθόδου G1_NNAACO.

```
       public static Solution G1_ACONNA(Problem problem, double[,] pheromones, ACO.Parameters
pars, Random randomizer)
                    {
                        if   ((problem.distances.GetLength(0)   !=   pheromones.GetLength(0))   |
(problem.distances.GetLength(1) != pheromones.GetLength(1)))
                            throw new Exception("Problem distance matrix and pheromones matrix do
not match.");

                        Stopwatch watch = new Stopwatch();
                        watch.Start();
                        Solution solution = new Solution(problem);

                        if (pars.SolutionGeneratorPars.NV <= 0)
                            throw new Exception("??");

                        bool restart = true;
                        int iterrations = 0;
                        while (restart == true)
                        {
                            restart = false;
                            iterrations++;
                            if (iterrations > stopAtIterationNumber)
                            {
                                //IF  ITERATIONS  ARE  OVER  stopAtIterationNumber  RETURN  AN  EMPTY
SOLUTION.
                                return new Solution(problem);
                            }

                            solution = new Solution(problem);
                            while  ((solution.isComplete  ==  false)  &&  (problem.unsolvable  ==
false))
                            {
                                //if the BestKnownNV is reached, restart
                                if (solution.routes.Count == pars.SolutionGeneratorPars.NV)
                                {
                                    restart = true;
                                    break;
                                }

                                //Next Route:
                                //Console.WriteLine("Unvisited: " + solution.unvisited.Count);
                                Route route = new Route(problem);

                                while  ((route.isComplete  ==  false)  &&  (problem.unsolvable  ==
false))
                                {
                                    //Next Customer:

                                    //Make list of suitable customers:
                                    List<Customer>                  suitables                  =
route.getSuitableFromLast(solution, problem);

                                    if (suitables.Count > 0)
                                    {
                                        //pick the next customer depending on a complicated rule

                                        Customer nextCust = problem.customers[0];

                                        //Build weights for each suitable customer:
                                        //the weight of each customer is (pheromone/distance) from
the last customer

                                        List<double> weights = new List<double>();

                                        foreach (Customer suitable in suitables)
                                        {
                                            //if it's the first customer, give favor to the more
distant customers (reverse visibility)
                                            double visiFL;
```

```
                                        if (route.customers.Count == 1)
                                            visiFL                                          =
Math.Pow(problem.distances[route.customers.Last().ID, suitable.ID], pars.a);
                                        else
                                            visiFL          =          Math.Pow(1          /
(problem.distances[route.customers.Last().ID, suitable.ID]), pars.a);
                                        double                          pherFL                  =
Math.Pow(pheromones[route.customers.Last().ID, suitable.ID], pars.b);
                                        weights.Add(pherFL * visiFL);
                                    }

                                    //Pick a customer from suitables
                                    int picker = Tool.WeightedPick(weights, randomizer);
                                    nextCust = suitables[picker];
                                    route.AddCustomer(nextCust, problem);
                                    //Mark customer as visited in the solution
                                    solution.RemoveFromUnvisited(nextCust);
                                }
                                else if (route.customers.Last().ID == 0) //if there are no
suitable customers and the vehicle just left the depot, problem is unsolvable.
                                {
                                    problem.unsolvable = true;
                                    route.Disp();
                                    Console.WriteLine();
                                    throw     new     Exception(String.Format("Problem     is
unsolvable."));
                                }
                                else
                                { //if there is no suitable customer add the depot to end the
loop.
                                    //Console.WriteLine("Route enters Solution");
                                    route.AddCustomer(problem.customers[0], problem);
                                    solution.AddRoute(route, problem);
                                }
                            }
                        }
                    }
                    //Console.WriteLine(iterrations);
                    watch.Stop();
                    solution.timeElapsed = watch.Elapsed;
                    return solution;
                }
```

5.3.3.2.2 G5_SortedByLA

Η μέθοδος αυτή αποτελεί μια παράλληλη βασική κατασκευαστική μέθοδος για το πρόβλημα δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) η οποία χρησιμοποιεί δεδομένα από τον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών (ACO). Εκτενέστερα, δημιουργούνται ταυτόχρονα ένα πλήθος (A) κενών διαδρομών και μία λίστα των (N) πελατών η οποία ταξινομείται ανάλογα με την αργότερη δυνατή εξυπηρέτησή τους. Στη συνέχεια, ξεκινά μία επαναληπτική διαδικασία (N) επαναλήψεων, στην οποία σε κάθε επανάληψη επιλέγεται ένας πελάτης προς δρομολόγηση, χρησιμοποιώντας ένα μηχανισμό τυχαίας επιλογής, δίνοντας μεγαλύτερη πιθανότητα στην επιλογή των πελατών με τη νωρίτερη χρονική στιγμή αργότερης δυνατής εξυπηρέτησης. Ο πελάτης που επιλέγεται, προστίθεται στην καταλληλότερη διαδρομή. Η καταλληλότητα του πελάτη για να προστεθεί σε κάποια διαδρομή εξαρτάται από την απόσταση και τις φερομόνες που χαρακτηρίζουν τη μετάβαση από τον τελευταίο πελάτη της διαδρομής προς τον πελάτη προς δρομολόγηση. Στην περίπτωση που κάποιος πελάτης δεν δύναται να προστεθεί σε καμία διαδρομή λόγω παραβίασης του περιορισμού της χωρητικότητας ή του περιορισμού χρονικών παραθύρων, τότε η μέθοδος αρχικοποιεί τις μεταβλητές που χρησιμοποιεί και επανεκκινείται. Στην περίπτωση που το πλήθος των επανεκκινήσεων υπερβεί ένα συγκεκριμένο όριο, το πλήθος (A) των αρχικών διαδρομών-οχημάτων αυξάνεται κατά ένα και συνεχίζεται η εκτέλεση της μεθόδου.

Παρουσιάζεται ο κώδικας της μεθόδου G5_SortedByLA.

```
        public static Solution G5_ACOSortedByLA(Problem problem, double[,] pheromones,
ACO.Parameters pars, Random randomizer)
                    {
                        Stopwatch watch = new Stopwatch();
                        watch.Start();
                        Solution sol = new Solution(problem);
                        Customer nextCust;
                        List<double> weights;
                        int k;

                        if ((problem.distances.GetLength(0) != pheromones.GetLength(0)) |
(problem.distances.GetLength(1) != pheromones.GetLength(1)))
                            throw new Exception("Problem distance matrix and pheromones matrix do
not match.");

                        k = pars.SolutionGeneratorPars.NV;
                        if (k == 0)
                        {
                            if (problem.problemTitle == null)
                                k = 2;
                            else if ((problem.problemTitle.StartsWith("C2")) |
                                    (problem.problemTitle.StartsWith("R2")) |
                                    (problem.problemTitle.StartsWith("RC2")))
                                k = 2;
                            else
                                k = 2;
                        }

                        bool restart = true;
                        int itersneeded = 0;
                        while (restart == true)
                        {
                            itersneeded++;
                            restart = false;
                            if (itersneeded > stopAtIterationNumber)
                            {
                                //IF ITERATIONS ARE OVER stopAtIterationNumber RETURN AN EMPTY
SOLUTION.
                                return new Solution(problem);
                            }

                            sol = new Solution(problem);

                            //make a list with all the customers (unvisited list)
                            sol = new Solution(problem);

                            //sort unvisited list by latest arrival
                            sol.unvisited = sol.unvisited.OrderBy(s => s.latestArrival).ToList();

                            //make a list of (k) routes
                            for (int i = 0; i < k; i++)
                                sol.routes.Add(new Route(problem));

                            //for each customer in unvisited
                            for (int customeri = 0; customeri < sol.unvisited.Count; customeri++)
                            {
                                nextCust = sol.unvisited[customeri];
                                //make weights for each route (last route customer -> unvisited
customer i)
                                weights = new List<double>();
                                foreach (Route toRoute in sol.routes)
                                {
                                    //if nextCust is suitable for toRoute, build weight.
                                    if (toRoute.IsSuitableAfterLast(nextCust, problem) == 1)
                                    {
                                        double visiFL = Math.Pow(1 /
(problem.distances[toRoute.customers.Last().ID, nextCust.ID]), pars.a);
                                        double pherFL =
Math.Pow(pheromones[toRoute.customers.Last().ID, nextCust.ID], pars.b);
                                        weights.Add(pherFL * visiFL);
                                    }
                                    else //if nextCust is not suitable to be added make his weight
zero.
                                    {
                                        weights.Add(0);
                                    }
                                }
                                //check if no route is suitable for the customer.
                                if (weights.All(x => x == 0))
                                {
                                    //restart SortedByLA with one more vehicle (or the same number
of vehicles)
                                    //if (randomizer.NextDouble() <
pars.SolutionGeneratorPars.G5_chanceToIncreaseNV)
                                    //    k++;

                                    restart = true;
                                    break;
                                }
                                //pick a route
                                int picker = Tool.WeightedPick(weights, randomizer);

                                //add nextCust to the chosen route
                                sol.routes[picker].AddCustomer(nextCust, problem);
                            }
```

```
                        if (restart == false)
                        {
                            //Console.Write(itersneeded + "  (");
                            //add the depot at the end of each route and remove empty routes
                            for (int i = 0; i < sol.routes.Count; i++)
                            {
                                if (sol.routes[i].customers.Count > 1)
                                    sol.routes[i].AddCustomer(problem.customers[0], problem);
                                else if (sol.routes[i].customers.Count == 1)
                                {
                                    sol.RemoveRoute(sol.routes[i], problem);
                                    i--;
                                }
                            }
                            sol.UpdatePropsFromRoutes(problem);
                            watch.Stop();
                            sol.timeElapsed += watch.Elapsed;
                            //Console.WriteLine(watch.Elapsed + ")");
                            return sol;
                        }
                    }
                    throw new Exception("how did it end here?");
                }
```

5.3.3.2.3 G7_SavingsForTW

Η μέθοδος αυτή αποτελεί μία προσαρμογή της παράλληλης κατασκευαστικής μεθόδου Savings για το πρόβλημα δρομολόγησης οχημάτων (VRP) στο πρόβλημα δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) και στον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών (ACO). Αρχικά, δημιουργείται μία διαδρομή για κάθε έναν πελάτη του προβλήματος ανεξάρτητα από το διαθέσιμο πλήθος των οχημάτων. Στη συνέχεια, επιλέγονται διαδρομές προς συνένωση, με προτεραιότητα των καταλληλότερων διαδρομών. Η καταλληλότητα της συνένωσης δύο διαδρομών εξαρτάται από την μείωση του συνολικού κόστους που προκύπτει και από τις φερομόνες που χαρακτηρίζουν τη μετάβαση οχήματος από τον τελευταίο πελάτη της πρώτης διαδρομής προς συνένωση προς τον πρώτο πελάτη της δεύτερης διαδρομής προς συνένωση. Η μέθοδος τερματίζεται όταν δεν δύναται να πραγματοποιηθούν περεταίρω συνενώσεις διαδρομών λόγω των περιορισμών χωρητικότητας ή χρονικών παραθύρων.

Παρουσιάζεται ο κώδικας της μεθόδου G7_SavingsForTW.

```
        public static Solution G7_Savings(Problem p, double[,] pheromones, ACO.Parameters pars,
Random randomizer)
                    {
                        if (pars.SolutionGeneratorPars.NV <= 0)
                            throw new Exception("??");

                        Stopwatch watch = new Stopwatch();
                        Solution solution = new Solution(p);
                        bool restart = true;
                        int iterrations = 0;

                        watch.Start();

                        while (restart == true)
                        {
                            restart = false;
                            iterrations++;
                            if (iterrations > stopAtIterationNumber)
                            {
                                //IF ITERATIONS ARE OVER stopAtIterationNumber RETURN AN EMPTY
SOLUTION.
                                return new Solution(p);
                            }

                            //SAVING ALGORITHM:
                            solution = new Solution(p);

                            //make N routes of 1 customer
                            foreach (Customer c in solution.unvisited)
                            {
                                Route newRoute = new Route(p);
                                newRoute.AddCustomer(c,p);
                                newRoute.AddCustomer(p.customers[0], p);
                                solution.AddRoute(newRoute, p);
                            }
```

```
                            solution.unvisited.Clear();

                            //repeat for p.n-1 times or until no feasible combination
                            for (int i = 0; i < p.n-1; i++)
                            {
                                //make a list of feasible route combinations
                                List<RouteCombination>   FeasibleRouteCombinationsList   =   new
List<RouteCombination>();
                                foreach (Route r1 in solution.routes)
                                {
                                    foreach (Route r2 in solution.routes)
                                    {
                                        if (Object.ReferenceEquals(r1, r2) == false)
                                        {
                                            RouteCombination rc = new RouteCombination(r1, r2, p);

                                            //if the combination is feasible add it to the list
                                            if (rc.isFeasible == 0)
                                                FeasibleRouteCombinationsList.Add(rc);
                                        }
                                    }
                                }

                                //if there are no feasible combinations stop the algorithm
                                if (FeasibleRouteCombinationsList.Count() == 0)
                                    break;

                                //build  weights  for  each  route  combination  on  savings  and
pheromones of r1.last - r2.first
                                RouteCombination pickedCombo;
                                List<double> weightsOfCombos = new List<double>();
                                foreach (RouteCombination rc in FeasibleRouteCombinationsList)
                                {
                                    //get phers of r1 last customer TO r2 first customer
                                    Customer                  lastCustomerOfr1                =
rc.r1.customers[rc.r1.customers.Count() -2];
                                    Customer firstCustomerOfr2 = rc.r2.customers[1];
                                    double      pher      =      pheromones[lastCustomerOfr1.ID,
firstCustomerOfr2.ID];


weightsOfCombos.Add(Math.Pow(rc.saving,pars.SolutionGeneratorPars.G7_savingPower)              *
Math.Pow(pher,pars.b));
                                }

                                //some savings may be negative. To make them all and non-zero,
                                //do weights += max - 2 * min
                                //ATTENTION: this convert affects the picker chances A LOT!
                                double minimumWeight = weightsOfCombos.Min();
                                double maximumWeight = weightsOfCombos.Max();
                                weightsOfCombos   =   weightsOfCombos.ConvertAll(x   =>   x   -
minimumWeight + pars.SolutionGeneratorPars.G7_weightMinCoef * (- minimumWeight + maximumWeight));

                                minimumWeight = weightsOfCombos.Min();
                                maximumWeight = weightsOfCombos.Max();

                                //actually pick the route
                                int picker = Tool.WeightedPick(weightsOfCombos, randomizer);
                                pickedCombo = FeasibleRouteCombinationsList[picker];

                                //delete the two routes that were combined and add the combined
route
                                solution.RemoveRoute(pickedCombo.r1, p);
                                solution.RemoveRoute(pickedCombo.r2, p);
                                solution.AddRoute(pickedCombo.combinedRoute, p);
                                foreach (Customer c in pickedCombo.combinedRoute.customers)
                                    if (c.ID != 0)
                                        solution.RemoveFromUnvisited(c);
                            }
                            //solution.Disp();
                            if (solution.routes.Count() > pars.SolutionGeneratorPars.NV)
                                restart = true;
                        }

                        solution.timeElapsed = watch.Elapsed;
                        return solution;
                    }
                }
```

5.3.3.3 Μέθοδοι τοπικής αναζήτησης

Παρουσιάζεται η μέθοδος τοπικής αναζήτησης που χρησιμοποιήθηκε. Στόχος των μεθόδων τοπικής αναζήτησης είναι να βελτιώσουν μία προυπάρχουσα λύση ενός προβλήματος VRPTW σύμφωνα με προκαθορισμένες παραμέτρους. Συνεπώς, δέχονται ως είσοδο τα χαρακτηριστικά ενός προβλήματος VRPTW και μία λύση του και επιστρέφουν μία κοντινή στην αρχική λύση του προβλήματος που είναι πιο βέλτιστη αυτής. Στην περίπτωση που δεν

μπορεί να βρεθεί πιο βέλτιστη γειτονική λύση της αρχικής, τότε η μέθοδοι τοπικής αναζήτησης επιστρέφουν την ίδια την αρχική λύση. Οι μέθοδοι τοπικής αναζήτησης δημιουργούν κοντινές ή γειτονικές λύσης της αρχικής εναλλάσσοντας πελάτες μεταξύ τους και χωρίζονται σε έσω-διαδρομής και έξω-διαδρομής ανάλογα με την απαγόρευση ή όχι της εναλλαγής πελατών που ανήκουν σε διαφορετικές διαδρομές. Οι μέθοδοι τοπικής αναζήτησης δύναται να χρησιμοποιηθούν τόσο από τον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών όσο και από τον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων.

5.3.3.3.1 2-opt

Η μέθοδος 2-opt είναι μία από τις πιο διαδεδομένες μεθόδους τοπικής αναζήτησης έσω-διαδρομής, η οποία δημιουργεί γειτονικές λύσεις εναλλάσσοντας δύο πελάτες οι οποίοι εξυπηρετούνται σε αλληλουχία. Στην παρούσα υλοποίηση της μεθόδου 2-opt η κύρια μέθοδος δέχεται ως είσοδο μία λύση του προβλήματος VRPTW και καλεί μία δευτερεύουσα μέθοδο για κάθε μία διαδρομή της λύσης αυτής. Η δευτερεύουσα μέθοδος αυτή δημιουργεί ένα πλήθος γειτόνων για τη διαδρομή αυτή ανάλογα με τις προκαθορισμένες παραμέτρους και επιστρέφει την πιο βέλτιστη διαδρομή από την αρχική και τις γειτονικές διαδρομές.

Παρουσιάζεται ο κώδικας της μεθόδου 2-opt.

```
        public static Solution TwoOpt1(Solution s, Problem p)
                    {
                        Solution optedSolution = new Solution(p);
                        //for each route in the solution
                        foreach (Route r in s.routes)
                        {
                            optedSolution.AddRoute(TwoOpt1(r,p), p);
                        }
                        return optedSolution;
                    }
                    public static Route TwoOpt1(Route r, Problem p)
                    {
                        //if there's only one customer do nothing.
                        if (r.customers.Count() == 3)
                            return r;

                        //CREATE THE NEIGHBORHOOD LIST by exchanging 2 customers
                        List<Route> neighborhood = GetNeighborhood(r, p);

                        if (r.customers.Count() > 5)
                        {
                            //enlarge neighborhood list by adding the neighbors of the neighbors
                            int neighSize = neighborhood.Count();
                            for (int oldNeighborhoodi = 0; oldNeighborhoodi < neighSize;
oldNeighborhoodi++)

neighborhood.AddRange(GetNeighborhood(neighborhood[oldNeighborhoodi], p));
                        }


                        //update neighborhood routes <for debugging>
                        foreach (Route r23 in neighborhood)
                            r23.Update(p);

                        //remove infeasible route exchanges
                        neighborhood.RemoveAll(x => (x.IsFeasible(p) != 0));

                        //if there are no feasible route exchanges return the initial route
                        if (neighborhood.Count() == 0)
                            return r;
                        else //else pick the one with the smallest r.ttlDistance (Route inherites
IComparable)
                        {
                            Route bestNeighbor = neighborhood.Min();

                            //if it's better than the starting route, return the opted route
                            if (bestNeighbor.ttlDistance < r.ttlDistance)
                                return bestNeighbor;
                            else
                                return r;
                        }

                    }

                    private static List<Route> GetNeighborhood(Route r, Problem p)
```

```
                {
                    //CREATE THE NEIGHBORHOOD LIST by exchanging 2 customers
                    List<Route> neighborhood = new List<Route>();

                    //there will be (r.customers.count)-3 customer combinations,
                    //because the first and the last customer is the depot
                    for (int routei = 1; routei < r.customers.Count() - 2; routei++)
                    {
                        //create the routei and add it to the neighborhood
                        Route neighRoute = new Route(p);
                        for (int custi = 1; custi < r.customers.Count(); custi++)
                        {
                            //when creating route1, exchange 1st and 2nd cust,
                            //when creating route2, exchange 2nd and 3rd cust, etc..
                            if (custi == routei)
                            {
                                neighRoute.AddCustomer(r.customers[custi + 1], p);
                                neighRoute.AddCustomer(r.customers[custi], p);
                                custi++;
                            }
                            else
                                neighRoute.AddCustomer(r.customers[custi], p);
                        }
                        neighborhood.Add(neighRoute);
                    }
                    return neighborhood;
                }
            }
```

5.3.3.4 Μέθοδος βελτιστοποίησης σμήνους σωματιδίων

Παρουσιάζονται οι μέθοδοι που αποτελούν την υλοποίηση του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO).

5.3.3.4.1 MasterPSO

Υλοποιήθηκε ένας αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (PSO) κατά την ανάπτυξη του οποίου δόθηκε έμφαση στην γενικότητα του παραγόμενου κώδικα και στην επεκτασιμότητά του, ώστε να είναι δυνατή η χρήση του αλγορίθμου σε οποιοδήποτε πρόβλημα βελτιστοποίησης του οποίου η λύση δύναται να αναπαρασταθεί ως ένα πολυδιάστατο διάνυσμα. Συνεπώς, ο κώδικας αλγορίθμου PSO που αναπτύχθηκε δέχεται ως είσοδο ένα σύνολο οποιουδήποτε πλήθους (A) αρχικών διανυσμάτων θέσης, οποιουδήποτε αριθμού διαστάσεων (N) , μία αντικειμενική συνάρτηση υπολογισμού κόστος και προαιρετικά ένα σύνολο πλήθους A αρχικών διανυσμάτων ταχύτητας N διαστάσεων. Στην περίπτωση που ο χρήστης δεν εισάγει ως είσοδο διανύσματα ταχύτητας, τότε ο αλγόριθμος παράγει ένα σύνολο από μηδενικά αρχικά διανύσματα ταχύτητας πλήθους (A) και διαστάσεων (N).

Πίνακας 5.3 Είσοδοι του αλγορίθμου PSO

Είσοδος	Σημασία	Περιγραφή
Διανύσματα θέσης	Απαραίτητη είσοδος	Διανύσματα πλήθους (A) και διαστάσεων (N) που αναπαριστούν την αρχική θέση των σωματιδίων στον N-διάστατο χώρο

Διανύσματα ταχύτητας	Προαιρετική είσοδος	Διανύσματα πλήθους (A) και διαστάσεων (N) που αναπαριστούν την αρχική ταχύτητα των σωματιδίων στον N-διάστατο χώρο
Αντικειμενική συνάρτηση	Απαραίτητη είσοδος	Μέθοδος που δέχεται ως είσοδο ένα διάνυσμα διάστασης (N) και επιστρέφει το κόστος που του αντιστοιχεί

Ο αλγόριθμος χρησιμοποιεί την αντικειμενική συνάρτηση που εισάγεται για να αξιολογήσει το διάνυσμα θέσης του κάθε σωματιδίου ώστε να εντοπίσει τα βέλτιστα και να συγκλίνει προς αυτά προσαρμόζοντας την ταχύτητα των σωματιδίων και επιστρέφει το σύνολο των λύσεων του σμήνους σωματιδίων και τη βέλτιστη λύση τους σύμφωνα με την προκαθορισμένη αντικειμενική συνάρτηση.

Για τη χρήση του γενικού αυτού κώδικα του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO) στην επίλυση του προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) απαιτείται η κωδικοποίηση της λύσης ενός VRPTW ώστε να αναπαρασταθεί σε ένα πολυδιάστατο διάνυσμα και η στρατηγική εισαγωγής των περιορισμών, είτε κατά την κωδικοποίηση των λύσεων, τακτική η οποία χρησιμοποιήθηκε στον προτεινόμενο αλγόριθμο, είτε στην αντικειμενική συνάρτηση ως ποινές.

Η μέθοδος MasterPSO καλείται από την μέθοδο MasterPSOforVRPTW η οποία δέχεται ως είσοδο ένα σύνολο λύσεων για το VRPTW, τις κωδικοποιεί ως πολυδιάστατα διανύσματα χρησιμοποιώντας τη μέθοδο GetPSOForm2 (Κεφάλαιο 5.3.3.4.5), αρχικοποιεί τις ταχύτητες των σωματιδίων και ορίζει ότι η μέθοδος CalculateCost θα ταυτίζεται με την ObjectiveFunc που αποτελεί την αντικειμενική συνάρτηση για την αξιολόγηση των σωματιδίων. Σε κάθε επανάληψη της μεθόδου MasterPSO, μετά την μετατόπιση του κάθε σωματιδίου εκτελείται η μέθοδος PathRelinking με αρχική λύση τη θέση του σωματιδίου και τελική λύση την τρέχουσα ολικά βέλτιστη και αν κατά την εκτέλεση της μεθόδου βρεθεί καλύτερη λύση, ανανεώνεται η θέση του σωματιδίου.

Παρουσιάζεται ο κώδικας της μεθόδου MasterPSO και της μεθόδου MasterPSOforVRPTW.

```
        public       static       List<Solution>       MasterPSOforVRPTW(Problem       p,Solution[]
initialSolutions,PSO_2.Parameters pars,Random randomizer)
                {
                    List<Solution> listSols = initialSolutions.ToList();
                    return MasterPSOforVRPTW(p, listSols, pars, randomizer);
                }
                public   static   List<Solution>   MasterPSOforVRPTW(Problem   p,   List<Solution>
initialSolutions, PSO_2.Parameters pars, Random randomizer)
                {

                    List<double[]> initialPositions = new List<double[]>();

                    for (int i = 0; i < initialSolutions.Count; i++)
                    {
                        initialSolutions[i].UpdatePSOformFromRoutes2();
                        initialPositions.Add(initialSolutions[i].PSOform2);
                    }
                    Func<double[], double> objectiveFunction = new Func<double[], double>(x =>
Tool.ObjectiveFunc(x,p));
```

```
                List<Solution> results = new List<Solution>();
                List<double[]> resultsPSOform2 = new List<double[]>();

                resultsPSOform2   =   MasterPSO(initialPositions,   objectiveFunction,   pars,
randomizer, p);

                foreach( double[] psoform in resultsPSOform2)
                {
                    results.Add(new Solution(p));
                    results.Last().PSOform2 = psoform;
                    results.Last().UpdateRoutesFromPSOform2(p);
                }

                return results;
            }
            public    static    List<double[]>    MasterPSO(List<double[]>    initialPositions,
Func<double[], double> calculateCost, PSO_2.Parameters pars, Random randomizer,Problem prob)
                {


                    //Initialization:
                    List<double[]> initialVelocities = new List<double[]>();
                    numberOfDimensions = initialPositions[0].GetLength(0);
                    numberOfParticles = initialPositions.Count;
                    particles = new List<particle>();
                    bestParticleIndex = 0;


                    //Set initial velocities equal zero
                    initialVelocities       =       initialPositions.ConvertAll(x       =>
Array.ConvertAll(x,y => y));
                    for (int particlei = 0; particlei < numberOfParticles; particlei++)
                        for (int d = 0; d < numberOfDimensions; d++)
                            initialVelocities[particlei][d] = 0;

                    //Create particles from positions and velocities
                    for (int particlei = 0; particlei < numberOfParticles; particlei++)
                    {
                        particles.Add(new           particle(initialPositions[particlei],
initialVelocities[particlei], calculateCost));
                    }

                    //Update gBest:
                    UpdateGBest(calculateCost);

                    //MAIN PSO LOOP:
                    //for a number of iterations
                    for (int iteration = 0; iteration < pars.iterations; iteration++)
                    {
                        //for each particle
                        foreach (particle p in particles)
                        {
                            //for each dimension
                            for (int d = 0; d < numberOfDimensions; d++)
                            {
                                //calculate distance from pBest and gBest
                                double x = p.position[d];

                                double distFromPBest = p.bestPosition[d] - x;
                                double distFromGBest = bestParticle.bestPosition[d] - x;

                                //update particle's dimension's velocity
                                double v = p.velocity[d];
                                double r1 = randomizer.NextDouble();
                                double r2 = randomizer.NextDouble();

                                p.velocity[d] = pars.w * v;
                                p.velocity[d] += pars.a1 * r1 * distFromPBest;
                                p.velocity[d] += pars.a2 * r2 * distFromGBest;

                                //update particle's dimension's position
                                p.position[d] = x + v;

                            } //end dimensions

                            double[] temp = Array.ConvertAll(p.position, x => x);

                            //make position of current particle an integer array
                            //keep indices
                            int[] positionIndices = new int[p.position.Length];
                            for (int i = 0; i < p.position.Length; i++)
                                positionIndices[i] = i;

                            //set the priority of depot to be the lowest priority-number
                            //(actually  the  lowest  priority-number  is  the  highest
priority)
                            p.position[0] = p.position.Min() - 1;

                            //sort both arrays on PSOform2
                            Array.Sort(p.position, positionIndices);

                            //build the new PSOform2 from the now sorted PSOform2indices
                            for (int i = 0; i < positionIndices.Length; i++)
```

```
                                    p.position[positionIndices[i]] = i;

                                //Path Relinking
                                //From   Marinakis   (2010),   this   swaps   customers   of
StartingPosition until it makes it the same
                                //as TargetPosition. In the process, if it finds a better
position than the particles best,
                                //it sets it as the current position of the particle.
                                p.position          =          Tool.PathRelinking(p.position,
bestParticle.bestPosition, calculateCost, p.bestPositionCost, randomizer);

                                if (new Solution(prob, p.position, true).IsFeasible(prob) !=
0)
                                    throw new Exception();

                                //Calculate particle cost:
                                p.updtCost();

                                //Update bestPosition of particle:
                                if
(FirstParticleIsBetter(p.position,p.bestPosition,calculateCost))
                                {
                                    p.setCurrentPositionAsBest();
                                }
                            } //end particles

                            //Update gBest
                            UpdateGBest(calculateCost);

                            //if more than 70% of the particles have the same bestPositionCost
as the bestParticle.bestPositionCost
                            //BREAK iterations.
                            if     (particles.FindAll(x     =>     x.bestPositionCost     ==
bestParticle.bestPositionCost).Count() > 0.7 * numberOfParticles)
                            {
                                break;
                            }
                        } //end iterations

                        //sort the particles on bestPositionCost
                        particles.Sort((x,                          y)                         =>
x.bestPositionCost.CompareTo(y.bestPositionCost));

                        //return the best one third particles
                        List<double[]> results = new List<double[]>();
                        for (int i = 0; i < particles.Count()/3; i++)
                        {
                            particle part = particles[i];
                            results.Add(part.bestPosition);
                        }

                        return results;
                    }
```

5.3.3.4.2 UpdateGBest

Η μέθοδος UpdateGBest περιλαμβάνει τον κώδικα του αλγορίθμου που ανανεώνει τη βέλτιστη μέχρι στιγμής λύση του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων. Η αξιολόγηση των λύσεων-σωματιδίων γίνεται με χρήση της μεθόδου FirstParticleIsBetter η οποία καλεί τη μέθοδο ObjectiveFunc κατά τη σύγκριση των λύσεων.

Παρουσιάζεται ο κώδικας του αλγορίθμου UpdateGbest.

```
public static void UpdateGBest(Func<double[],double> calcCost)
                    {
                        for (int particlei = 0; particlei < numberOfParticles; particlei++)
                        {
                            particle p = particles[particlei];
                            if
(FirstParticleIsBetter(p.bestPosition,bestParticle.bestPosition, calcCost))
                            {
                                bestParticleIndex = particlei;
                            }
                        }
                    }
```

5.3.3.4.3 PathRelinking

Η μέθοδος PathRelinking δέχεται ως είσοδο μία αρχική θέση σωματιδίου, μία θέση-στόχο και μία κρίσιμη ποσότητα της αντικειμενικής συνάρτησης. Εκτελεί διαδοχικές μεταθέσεις στις

διαστάσεις της αρχικής θέσης μέχρις ότου καταλήξει να ταυτιστεί με την θέση-στόχο, ή έως ότου το αντικειμενικό κόστος της ενδιάμεσης θέσης του σωματιδίου είναι μικρότερο της προκαθορισμένης κρίσιμης ποσότητας. Στην πρώτη περίπτωση, επιστρέφεται η αρχική θέση ή η θέση-στόχος με ίσες πιθανότητες, ενώ στη δεύτερη περίπτωση επιστρέφεται η ενδιάμεση βέλτιστη θέση του σωματιδίου.

Παρουσιάζεται ο κώδικας της μεθόδου PathRelinking.

```
        public static double[] PathRelinking(double[] StartingPosition, double[] TargetPosition,
Func<double[],double> CostOf, double bestCost, Random rand)
                {
                    //NEW ADDITION: Path Relinking for PSO Solution Representation for VRP
                    //From Marinakis (2010), this does swaps customers of StartingPosition until
it makes it the same
                    //as TargetPosition. In the process, if it finds a better position than the
particles best one,
                    //it sets it as the current position of the particle.

                    //returns the current position if its cost is better than bestCost
                    //if not it returns the initialPosition.

                    //create a copy of starting position
                    double[] currentPosition = Array.ConvertAll(StartingPosition, x => x);

                    //make a list of the indices of targetPosition
                    List<int> indices = new List<int>();
                    for (int i = 0; i < TargetPosition.Length; i++)
                        indices.Add(i);

                    while (indices.Count() > 0)
                    {
                        //pick a random index d and remove him from the list
                        int d = indices[rand.Next(indices.Count())];
                        indices.Remove(d);

                        //find d in currentPosition and swap it with its position in
targetPosition
                        int dindex = Array.FindIndex(currentPosition, x => x ==
TargetPosition[d]);

                        //swap the two
                        double temp = currentPosition[d];
                        currentPosition[d] = TargetPosition[d];
                        currentPosition[dindex] = temp;

                        //if the currentPosition's cost is less than the TargetCost, return
currentPosition
                        if (CostOf(currentPosition) < bestCost)
                            return currentPosition;
                    }
                    double picker = rand.NextDouble();
                    if (picker < 0.5)
                        return StartingPosition;
                    else
                        return TargetPosition;

                }
```

5.3.3.4.4 ObjectiveFunc

Για την αξιολόγηση των λύσεων, χρησιμοποιείται η μέθοδος ObjectiveFunc η οποία δέχεται ως είσοδο μία λύση του προβλήματος VRPTW, είτε σε κωδικοποιημένη μορφή προς χρήση για τον αλγόριθμο PSO είτε σε απλή μορφή κλάσης Solution, και επιστρέφει την τιμή της αντικειμενικής συνάρτησης για τη λύση αυτή. Χρησιμοποιήθηκε η αντικειμενική συνάρτηση του Gong (2012) ώστε να επιτευχθεί η κατασκευή μίας ποσότητας προς ελαχιστοποίηση που να περιλαμβάνει τους δύο αντικειμενικούς στόχους του προβλήματος VRPTW

$$ObjectiveFunc(x) = NVof(x) + Normalize(x.ttlDistance) \quad (5.2)$$

$$Normalize(y) = arctan(y) * 2 / pi \quad (5.3)$$

Η κατασκευή μίας αντικειμενικής συνάρτησης για δύο αντικειμενικούς στόχους είναι δυνατή διότι ο πρωτεύον αντικειμενικός στόχος, το πλήθος των οχημάτων προς ελαχιστοποίηση, είναι ακέραιος αριθμός. Συνεπώς, χρησιμοποιώντας την (5.3) η οποία είναι αύξουσα και φραγμένη στο διάστημα $(0,1)$, κανονικοποιούμε το δεύτερο αντικειμενικό στόχο, τη συνολικά διανυόμενη απόσταση από το σύνολο των οχημάτων και η ποσότητα αυτή προστίθεται στον αριθμό των οχημάτων εν χρήσει. Για την κανονικοποίηση της διανυόμενης απόστασης δύναται να χρησιμοποιηθεί οποιαδήποτε αύξουσα συνάρτηση, που είναι κάτω φραγμένη στο 0 και άνω φραγμένη στο 1, για την οποία ισχύει

$$lim \lim_{y \to \infty} Normalize(y) = 1. \quad \textbf{(5.4)}$$

5.3.3.4.5 PSOform2, UpdateRoutesFromPSOform2 & UpdatePSOformFromRoutes2

Η παράμετρος PSOform2 της κλάσης τύπου Solution αποτελεί μία κωδικοποίηση μίας λύσης ενός προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW) σε ένα πολυδιάστατο διάνυσμα, ώστε να δύναται να χρησιμοποιηθεί από τον κώδικα του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO). Η παράμετρος αυτή ανανεώνεται από τα χαρακτηριστικά της λύσης μέσω της μεθόδου UpdatePSOformFromRoutes2, ενώ τα χαρακτηριστικά της λύσης ανανεώνονται από την παράμετρο PSOform2 μέσω της μεθόδου UpdateRoutesFromPSOform2. Η μέθοδος κωδικοποίησης (UpdatePSOformFromRoutes2) και αποκωδικοποίησή (UpdateRoutesFromPSOform2) της λύσης του προβλήματος είναι βασισμένες στην προσέγγιση του Gong et al. (2012) που παρουσιάζεται αναλυτικά στο Κεφάλαιο 2.6.1.3.

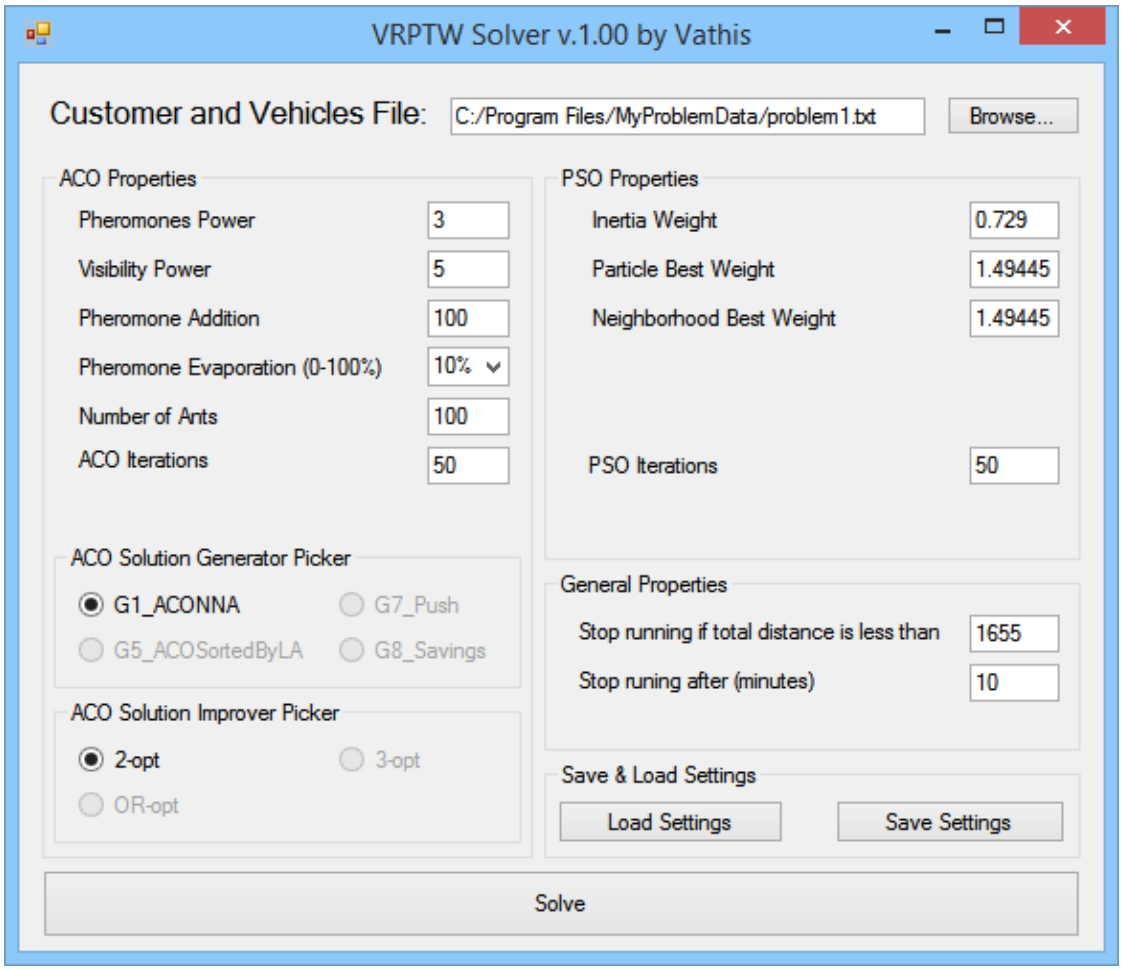
Παρουσιάζεται ο κώδικας της μεθόδου UpdatePSOformFromRoutes2 και της μεθόδου UpdateRoutesFromPSOform2.

```
        public void UpdateRoutesFromPSOform2(Problem p)
                {
                    //Clear routes
                    routes.Clear();

                    //initialize one route
                    routes.Add(new Route(p));

                    //for all customers, beginning with customer with smaller priority (priority
== 1), route them
                    for (double priorityi = 1; priorityi < p.n; priorityi++)
                    {
                        //find index of customer with priorityi
                        int customerID = Array.FindIndex(PSOform2, x => x == priorityi);
                        Customer customerToRoute = p.customers[customerID];

                        //if he's suitable to be added to the current route, route him
                        if (routes.Last().IsSuitableAfterLast(customerToRoute,p) == 1)
                        {
                            routes.Last().AddCustomer(customerToRoute,p);
                        }
                        else //else close the current route, create a new route and add
customerToRoute to the new route
                        {
                            routes.Last().AddCustomer(p.customers[0], p);
                            routes.Add(new Route(p));
                            routes.Last().AddCustomer(customerToRoute, p);
                        }
                    }
        public void UpdatePSOformFromRoutes2() //new way. PSO can only produce feasible solutions
with this.
                {
                    //careful: the first double, PSOform2[0] stays null/zero
                    int Priority = 0;
                    for (int ri =0; ri < routes.Count; ri++)
                    {
                        Route currentRoute = routes[ri];
                        for (int ci = 0; ci < currentRoute.customers.Count; ci++)
```

```
                    {
                        Customer currentCustomer = currentRoute.customers[ci];
                        if (currentCustomer.ID != 0)
                        {
                            Priority++;
                            PSOform2[currentCustomer.ID] = Priority;
                        }
                    }
                }
            }
```

5.4 Διεπαφή χρήστη μηχανής

Σχεδιάστηκε η διεπαφή του χρήστη μηχανής ώστε να δύναται η εισαγωγή των χαρακτηριστικών των πελατών προς εξυπηρέτηση και των διαθέσιμων οχημάτων, των παραμέτρων του αλγόριθμου Βελτιστοποίησης Αποικίας Μυρμηγκιών και του αλγορίθμου Βελτιστοποίησης σμήνους σωματιδίων.



Εικόνα 5.1 : Διεπαφή Χρήστη Μηχανής

Αρχικά, ο χρήστης του λογισμικού ορίζει το αρχείο με τα δεδομένα του προς επίλυση προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (VRPTW).

Στη συνέχεια, εισάγει τις απαραίτητες παραμέτρους των δύο αλγορίθμων. Στο αριστερό μέρος της εφαρμογής εμφανίζονται οι παράμετροι του αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών (ACO) καθώς και η επιλογή των μεθόδων κατασκευής λύσης και τοπικής αναζήτησης. Στο δεξί μέρος της εφαρμογής εμφανίζονται οι παράμετροι του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων (PSO). Στο δεξί μέρος της εφαρμογής εμφανίζεται υπό τον τίτλο «General Properties» η επιλογή διακοπής της εκτέλεσης του υβριδικού αλγορίθμου είτε μετά από προκαθορισμένο χρόνο εκτέλεσης, είτε όταν η τρέχουσα βέλτιστη λύση είναι κάτω από ένα συγκεκριμένο όριο. Επιπρόσθετα, δίνεται η

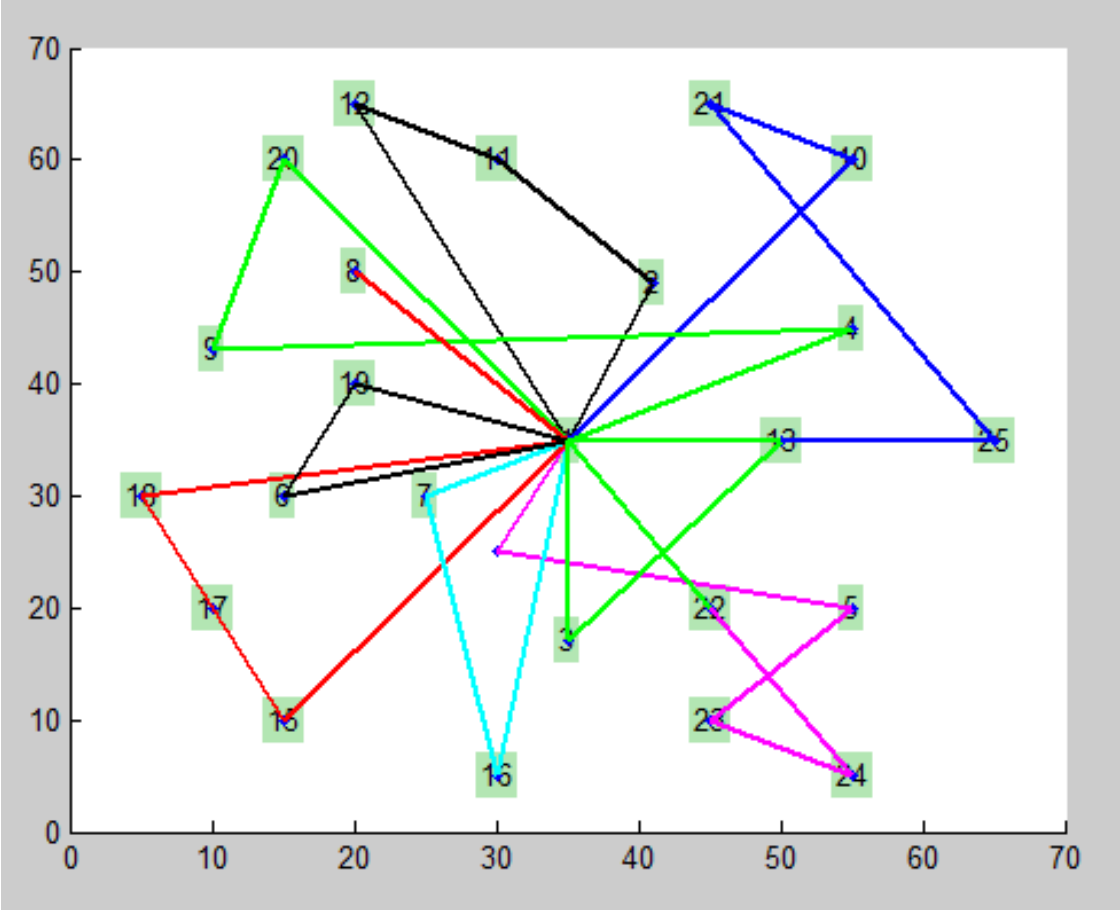
επιλογή αποθήκευσης των ρυθμίσεων των παραμέτρων καθώς και η εύκολη ανάκτησή τους μέσω των επιλογών «Save Settings» και «Load Settings».

Τέλος, ο χρήστης του λογισμικού δίνει το σήμα έναρξης των υπολογισμών για την επίλυση του προκαθορισμένου προβλήματος μέσω της επιλογής «Solve».

Μετά το πέρας των υπολογισμών το σύστημα επιστρέφει στο χρήστη το δρομολόγιο που υπολόγισε τόσο γραφικά, ώστε να δύναται στο χρήστη η συνολική εποπτεία και γρήγορη επαλήθευση της λύσης δρομολόγησης, όσο και σε πίνακα σε μορφή αρχείου Microsoft Excel στον οποίο παρουσιάζονται αναλυτικά οι τιμές των παραμέτρων που χρησιμοποιήθηκαν και τα δρομολόγια του κάθε οχήματος ξεχωριστά.

	A	B	C	D
1	Date and Time	12/8/2014 6:07:53 μμ	**Λύση Προβλήματος:**	
2	Τίτλος Προβλήματος	R101	Κωδικός Λύσης	63543463631156
3	Αριθμός Πελατών Προβλήματος	50	Οχήματα που Χρησιμοποιήθηκαν	12
4	Χωρητικότητα Οχημάτων	200	Συνολική Διανυόμενη Απόσταση	1135,072493
5			Χρόνος Εκτέλεσης	5,0623067
6	**Παράμετροι ACO:**		Επανάληψη Εύρεσης Λύσης	73
7	Δύναμη Φερομονών	3	**Διαδρομές Οχημάτων:**	
8	Δύναμη Ορατότητας	5	Διαδρομή Οχήματος 1	0-21-40-20-0
9	Προσθήκη Φερομονών	100	Διαδρομή Οχήματος 2	0-27-7-8-46-17-0
10	Εξάτμιση Φερομονών	0,05	Διαδρομή Οχήματος 3	0-5-16-37-0
11	Αριθμός Μυρμηγκιών	100	Διαδρομή Οχήματος 4	0-2-18-6-13-0
12	Κατασκευαστής Λύσης	G1_ACONNA	Διαδρομή Οχήματος 5	0-45-11-49-48-0
13	Βελτιωτής Λυσης	2-opt	Διαδρομή Οχήματος 6	0-33-29-3-24-25-0
14	Επαναλήψεις ACO	100	Διαδρομή Οχήματος 7	0-28-12-50-1-0
15	**Παράμετροι PSO:**		Διαδρομή Οχήματος 8	0-39-23-22-26-4-0
16	Βάρος Αδράνειας	0,729	Διαδρομή Οχήματος 9	0-14-15-41-0
17	Βάρος Βέλτιστου Σωματιδίου	1,49445	Διαδρομή Οχήματος 10	0-36-47-19-10-32-0
18	Βάρος Βέλτιστου Γειτονιάς	1,49445	Διαδρομή Οχήματος 11	0-31-30-9-34-35-0
19	Επαναλήψεις PSO	150	Διαδρομή Οχήματος 12	0-42-44-38-43-0

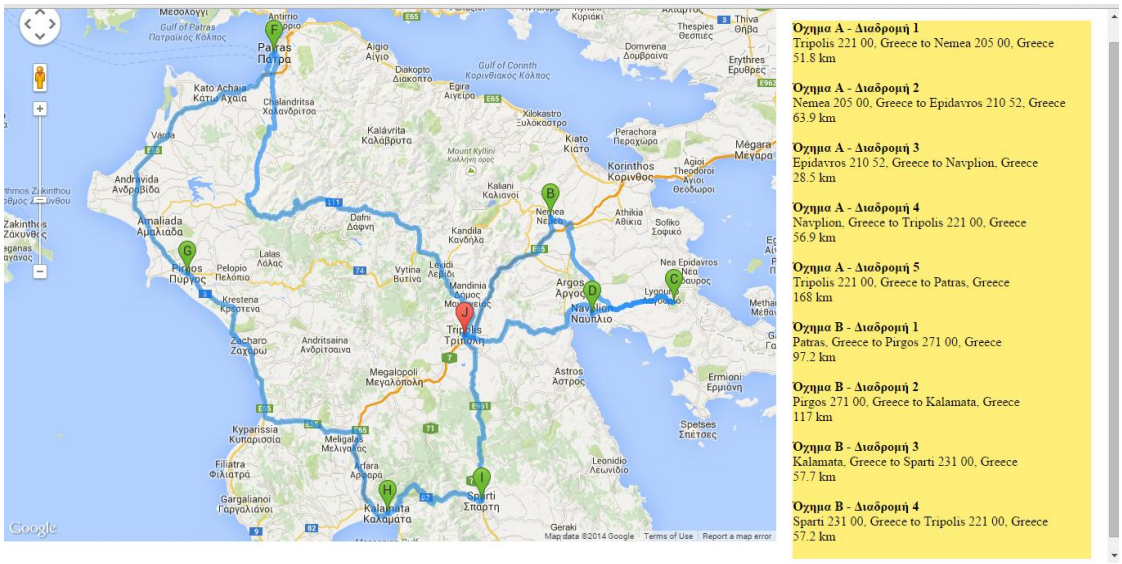
Σχήμα 5.3 : Έξοδος Συστήματος σε μορφή Αρχείου MS Excel



Σχήμα 5.4 : Γραφική Έξοδος Συστήματος

Επιπρόσθετα, στην περίπτωση πραγματικού σεναρίου δρομολόγησης οχημάτων, δύναται η γραφική αναπαράσταση των δρομολογίων σε χάρτη Google. Η δυνατότητα αυτή που παρέχει το λογισμικό βρίσκεται σε δοκιμαστικό στάδιο και στην παρούσα μορφή του παρέχεται για

προβλήματα με περιορισμένο αριθμό πελατών. Παρουσιάζεται παράδειγμα ενός προβλήματος δρομολόγησης επτά πελατών και δύο οχημάτων.



Σχήμα 5.5 : Παράδειγμα Εξόδου Συστήματος Πραγματικών Δεδομένων

6 Αποτελέσματα και Αξιολόγηση

6.1 Παραμετροποίηση του προτεινόμενου αλγορίθμου

Για την παραμετροποίηση του αλγορίθμου πραγματοποιήθηκε έρευνα στη βιβλιογραφία για τις προτεινόμενες τιμές των παραμέτρων των δύο αλγορίθμων ξεχωριστά.

Για την παραμετροποίηση του αλγορίθμου ACO επιλέχθηκαν οι παράμετροι a, b, ρ, a/c και Q επειδή, καθότι φάνηκε μετά από βιβλιογραφική έρευνα, ότι επηρεάζουν περισσότερο την αποδοτικότητα του αλγορίθμου. Παρουσιάζονται οι τιμές αυτών των παραμέτρων που θεωρούνται βέλτιστες από τη βιβλιογραφία.

Πίνακας 6.1 Οι προτεινόμενες από τη βιβλιογραφία παράμετροι του ACO

	ACO	a	b	ρ	a/c	Q
1999	**(Bullnheimer, Hartl, & Strauss, 1999)**	5	5	0,75	1	-
2003	**Machado, Tavares, Pereira, and Costa (2002)**	-	-	-	2	-
2003	**Reimann**			0,95	0,5	-
2004	**Bell**	1	2,3		0,5	-
2004	**Reimann**	5	5	0,95	1	-
2006	**Wei Dong**	1	6	0,7	0,5	-
2007	**Rizzoli**	2	1	0,8	1	1000
2005	**Gaertner**	1	6-12	0,6	1	-
2010	**Niknam**	-	-	0,99	1	-
2008	**Moussouni**	1	1	0,9	1	-
2014	**Chaung Lin**	1	1	0,8	1	1
2014	**Crawford**	1	2	0,98	1	-
2010	**Zhang (Bin 2009)**			0,5	1	100
2010	**Bin Yu (2011)**	2	6	0,99	1	-
1999	**Dorigo (2006)**	1	5	0,5	1	-
2011	**B. Yu**	2	1	0,8	1	1000

- a : η δύναμη στην οποία υψώνονται οι φερομόνες
- b : η δύναμη στην οποία υψώνεται η ορατότητα
- ρ : ο πολλαπλασιαστής μείωσης των φερομονών λόγω εξάτμισης
- a/c : ο λόγος πλήθους μυρμηγκιών προς πλήθος πελατών
- Q : ο σταθερός αριθμός που προστίθεται σε κάθε ανανέωση των φερομονών

Επιλέχθηκαν οι παρακάτω τιμές για την παραμετροποίηση του αλγορίθμου ACO

- $\alpha \in \{ 1,\ 3,\ 6,\ 9 \}$
- $\beta \in \{ 1,\ 3,\ 6,\ 9 \}$
- $\rho \in \{ 0.5\ ,\ 0.75,\ 0.99 \}$
- $Q \in \{ 10,\ 100,\ 1000 \}$
- $a/c \in \{ 0.5,\ 1,\ 1.5 \}$

Πραγματοποιήθηκαν εκτελέσεις του προτεινόμενου αλγορίθμου για κάθε έναν από τους 432 συνδυασμούς των παραπάνω τιμών των παραμέτρων για κάθε ένα από προβλήματα αναφοράς (Κεφάλαιο 6.2) για 25, 50 και 100 πελάτες. Σύμφωνα με τα αποτελέσματα επιλέχθηκαν οι παρακάτω τιμές για τις παραμέτρους του αλγορίθμου προσομοίωσης αποικίας μυρμηγκιών:

- $\alpha = 3$
- $\beta = 6$
- $\rho = 0.9$
- $Q = 100$
- $a/c = 1$

Μετά από διερεύνηση της βιβλιογραφίας επιλέχθηκαν οι παρακάτω τιμές για τις παραμέτρους του αλγορίθμου PSO.

- $w = 0.729$
- $a1 = 1.49445$
- $a2 = 1.49445$

6.2 Παρουσίαση των προβλημάτων αναφοράς

Τα προβλήματα αναφοράς που χρησιμοποιήθηκαν για τον έλεγχο σωστής λειτουργίας, αποδοτικότητας και αποτελεσματικότητας του προτεινόμενου αλγορίθμου είναι αυτά που ανέπτυξε ο Solomon (Marius M Solomon, 1987).

Ανέπτυξε έξι κατηγορίες προβλημάτων (R1, R2, C1, C2, RC1, RC2), που περιλαμβάνουν συνολικά 56 προβλήματα 100 πελατών και ομοιογενούς στόλου οχημάτων, ο σχεδιασμός των οποίων στοχεύει στην επισήμανση των σημαντικότερων παραγόντων που επηρεάζουν τη συμπεριφορά των αλγορίθμων επίλυσης προβλημάτων VRPTW. Αυτοί οι παράγοντες περιλαμβάνουν τα γεωγραφικά χαρακτηριστικά των πελατών και της αποθήκης, το πλήθος των πελατών που εξυπηρετούνται από κάθε όχημα και τα χαρακτηριστικά των παραθύρων χρόνου του κάθε πελάτη όπως το ποσοστό των πελατών που έχουν παράθυρα χρόνου, το μέγεθος του παραθύρου χρόνου και η χρονική του τοποθέτηση.

Συγκεντρωτικά, τα χαρακτηριστικά των πελατών 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
- Τη νωρίτερη δυνατή χρονική στιγμή έναρξης εξυπηρέτησης του πελάτη (έναρξη χρονικού παραθύρου) – για την κεντρική αποθήκη, δηλώνει τη χρονική στιγμή που όλα τα οχήματα ξεκινούν το δρομολόγιό τους από αυτήν
- Την αργότερη δυνατή χρονική στιγμή έναρξης εξυπηρέτησης του πελάτη (λήξη χρονικού παραθύρου) – για την κεντρική αποθήκη, δηλώνει τη χρονική στιγμή που απαιτείται όλα τα οχήματα να έχουν επιστρέψει στην κεντρική αποθήκη
- Την απαιτούμενη χρονική διάρκεια για την εξυπηρέτηση του πελάτη

Τα γεωγραφικά δεδομένα είναι τυχαία και ομοιόμορφα για τα προβλήματα R1 και R2, σε συστάδες για τα προβλήματα τύπου C1 και C2, και μεικτά για τα προβλήματα τύπου RC1 και RC2. Τα προβλήματα τύπου R1, C1 και RC1 έχουν μικρό ορίζοντα προγραμματισμού, ενώ οι περιορισμοί χωρητικότητας οχημάτων και χρονικού ορίου επιστροφής των οχημάτων στην κεντρική αποθήκη επιτρέπουν την εξυπηρέτηση συγκριτικά μικρού αριθμού πελατών από το κάθε όχημα και συνεπώς απαιτούν μεγάλο αριθμό οχημάτων (3 με 19 οχήματα ανάλογα το πρόβλημα). Αντίθετα, τα προβλήματα τύπου R2, C2 και RC2 έχουν μεγάλο ορίζοντα προγραμματισμού και συγκριτικά μεγάλη χωρητικότητα οχημάτων, επιτρέποντας την εξυπηρέτηση μεγάλου αριθμού πελατών από κάθε όχημα και συνεπώς τη χρήση μικρού αριθμού οχημάτων (2 με 9 οχήματα, ανάλογα το πρόβλημα).

Τέλος, να σημειωθεί πως τα προβλήματα κωδικοποιούνται χρησιμοποιώντας αρχικά τον τύπο του προβλήματος (πχ. RC1) στη συνέχεια τον αύξοντα αριθμό (πχ. RC105) και τέλος τον αριθμό των πελατών που χρησιμοποιήθηκαν (πχ. RC105.25, RC105.50, RC105.100).

6.3 Πειραματική Επαλήθευση του Προτεινόμενου Αλγορίθμου και Στατιστική Ανάλυση Αποτελεσμάτων

Για την πειραματική επαλήθευση του αλγορίθμου επιλέχθηκαν τα προβλήματα αναφοράς του (Marius M Solomon, 1987) καθώς είναι τα πιο πολυμελετημένα και τα πιο συχνά συναντούμενα προβλήματα αναφοράς τύπου VRPTW στη βιβλιογραφία. Εκτελέστηκαν 10 τρεξίματα του προτεινόμενου υβριδικού αλγορίθμου σε κάθε ένα από τα 56 επιλεχθέντα προβλήματα αναφοράς.

Στον πίνακα που ακολουθεί παρουσιάζεται ενδεικτικά μέρος των αποτελεσμάτων. Συγκεκριμένα, παρουσιάζεται για κάθε πρόβλημα αναφοράς η μέση απόκλιση των αποτελεσμάτων του προτεινόμενου υβριδικού αλγορίθμου από τα βέλτιστα της βιβλιογραφίας (Gong et al., 2012) για τους δύο αντικειμενικούς στόχους, τον αριθμό των χρησιμοποιούμενων οχημάτων και την συνολική διανυόμενη απόσταση των οχημάτων.

Πίνακας 6.2 Αποτελέσματα Βελτιστοποίησης των προβλημάτων αναφοράς του Solomon

Τίτλος Προβλήματος	NV	BK NV	Απόκλ. από BK NV	Dist	BK Dist	Απόκλ. από BK Dist	CPU time
C101.25	3	3	0.0%	191.8	191.3	0.3%	7.7
C102.25	3	3	0.0%	214.4	190.3	12.7%	9.1
C103.25	3	3	0.0%	192.0	190.3	0.9%	10.4
C104.25	3	3	0.0%	197.8	186.9	5.8%	8.9
C104.50	5	5	0.0%	424.3	358.0	18.5%	104.2
C105.25	3	3	0.0%	289.2	191.3	51.2%	8.1
C106.25	3	3	0.0%	191.8	191.3	0.3%	7.5
C107.25	3	3	0.0%	229.1	191.3	19.7%	9.1
C108.25	3	3	0.0%	191.8	191.3	0.3%	9.1
C109.25	3	3	0.0%	191.8	191.3	0.3%	8.2
C109.50	5	5	0.0%	504.4	362.4	39.2%	70.2
C201.25	2	2	0.0%	215.5	214.7	0.4%	15.1
C202.25	1	1	0.0%	240.3	223.3	7.6%	18.6
C204.50	2	2	0.0%	479.4	350.1	36.9%	132.6
C207.25	1	1	0.0%	274.8	274.8	0.0%	16.2
C207.50	2	2	0.0%	510.7	426.1	19.8%	183.4
C208.25	1	1	0.0%	231.4	229.8	0.7%	16.3
C208.50	2	2	0.0%	429.4	350.5	22.5%	185.3
R101.25	8	8	0.0%	634.3	617.1	2.8%	11.3
R102.25	7	7	0.0%	573.8	547.1	4.9%	8.7
R104.25	4	4	0.0%	469.1	416.9	12.5%	16.0
R107.25	4	4	0.0%	485.0	424.3	14.3%	11.2
R108.25	4	4	0.0%	429.8	397.3	8.2%	9.7
R110.25	4	4	0.0%	479.8	444.1	8.0%	8.4
R111.25	4	4	0,0%	472,64	529,7	-10,8%	9.3
R112.25	4	4	0.0%	413.0	393.0	5.1%	5.5
R201.25	2	2	0.0%	625.8	523.7	19.5%	11.3
R202.25	2	2	0.0%	524.6	455.5	15.2%	13.2
R203.25	2	2	0.0%	448.4	408.9	9.7%	14.4
R204.25	1	1	0.0%	491.9	389.9	26.2%	16.2
R204.50	2	2	0.0%	665.6	518.6	28.4%	242.3
R206.25	1	1	0.0%	446.2	413.2	8.0%	21.1
R206.50	2	2	0.0%	890.7	661.6	34.6%	146.3
R207.50	2	2	0.0%	810.4	594.0	36.4%	180.6
R208.25	1	1	0.0%	367.3	328.2	11.9%	32.0
R208.50	2	2	0.0%	630.6	508.4	24.0%	185.6
R209.25	1	1	0.0%	502.4	438.2	14.6%	14.8
R211.25	1	1	0.0%	411.7	361.7	13.8%	20.7
R211.50	2	2	0.0%	749.6	562.7	33.2%	157.9
RC103.25	3	3	0.0%	336.3	332.8	1.1%	8.8

RC104.25	3	3	0.0%	314.2	306.6	2.5%	9.8
RC105.25	4	4	0.0%	460.1	411.3	11.9%	6.2
RC106.25	3	3	0.0%	346.5	345.5	0.3%	9.9
RC107.25	3	3	0.0%	298.9	298.3	0.2%	8.5
RC108.25	3	3	0.0%	295.0	294.5	0.2%	10.1
RC202.25	2	2	0.0%	456.1	376.1	21.3%	10.5
RC204.25	1	1	0.0%	335.8	327.3	2.6%	22.4
RC204.50	2	2	0.0%	528.9	479.2	10.4%	273.8
RC205.25	2	2	0.0%	468.5	386.2	21.3%	12.6
RC208.25	1	1	0.0%	338.9	309.9	9.4%	23.0
RC208.50	2	2	0.0%	555.9	498.8	11.5%	169.5

Όπου NV είναι ο μέσος αριθμός των οχημάτων που χρησιμοποιούνται για την εξυπηρέτηση των πελατών (Number of Vehicles), $BK\ NV$ είναι ο αριθμός των οχημάτων που χρησιμοποιούνται για την εξυπηρέτηση των πελατών από τη βέλτιστη λύση της βιβλιογραφίας (Best Known Number of Vehicles – BK NV), $Dist$ είναι ο μέσος όρος της συνολικής απόστασης που διανύει ο στόλος οχημάτων σε κάθε τρέξιμο του αλγορίθμου, $BK\ Dist$ είναι η συνολική απόσταση που διανύει ο στόλος οχημάτων στη βέλτιστη λύση κατά τη βιβλιογραφία και τέλος $CPU\ Time$ είναι ο χρόνος εκτέλεσης του υβριδικού αλγορίθμου σε δευτερόλεπτα.

7 Συμπεράσματα

Η δρομολόγηση οχημάτων αφορά την κατάστρωση ενός πλάνου με το οποίο ένας στόλος οχημάτων θα εξυπηρετήσει ένα σύνολο πελατών, με στόχο την επίτευξη του μεγαλύτερου δυνατού κέρδους –ή του μικρότερου δυνατού κόστους. Συγκεκριμένα, το πλάνο καθορίζει τους πελάτες που θα εξυπηρετηθούν από το κάθε όχημα αλλά και τη σειρά εξυπηρέτησης που θα τηρηθεί, ώστε να επιτευχθεί ο αντικειμενικός στόχος αλλά και να ικανοποιηθούν οι κατά περίπτωση περιορισμοί.

Τα προβλήματα δρομολόγησης οχημάτων μελετώνται εκτενώς τα τελευταία εξήντα χρόνια με αυξανόμενο ενδιαφέρον από την επιστημονική κοινότητα, το οποίο πηγάζει από το γεγονός πως η μεταφορά αγαθών ευθύνεται για ένα μεγάλο ποσοστό του κόστους των αγαθών, συνεπώς ακόμη και μικρές βελτιώσεις στη διεκπεραίωσή τους προκαλεί σημαντικές μειώσεις στο συνολικό κόστος των αγαθών, ιδίως μακροπρόθεσμα, καθιστώντας συμφέρουσα την επένδυση στην ανάπτυξη αποδοτικών τεχνικών δρομολόγησης.

Στην παρούσα εργασία μετά τη μελέτη της βιβλιογραφίας για τα είδη προβλημάτων δρομολόγησης επιλέχθηκε προς επίλυση η παραλλαγή του προβλήματος δρομολόγησης οχημάτων με παράθυρα χρόνου (Vehicle Routing Problem with Time Windows – VRPTW) η οποία και πλησιάζει σε μεγάλο βαθμό την πραγματικότητα. Εξετάστηκαν τα μαθηματικά μοντέλα που έχουν χρησιμοποιηθεί από την επιστημονική κοινότητα για την μοντελοποίηση του προβλήματος VRPTW και παρουσιάζεται ένα μαθηματικό μοντέλο βασισμένο στη βιβλιογραφία καθώς και ένα παράδειγμα εφαρμογής του για την καλύτερη κατανόησή του.

Στη συνέχεια, μελετήθηκαν δύο από τους δημοφιλέστερους μεταευρετικούς αλγορίθμους για την επίλυση του επιλεχθέντος προβλήματος, ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization – ACO) και ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization – PSO). Αναπτύχθηκε ένας υβριδικός αλγόριθμος που χρησιμοποιεί αρχές από τους παραπάνω αλγορίθμους, ώστε να εκμεταλευτεί τα πλεονεκτήματα και να εξαλείψει τα μειονεκτήματα του καθενός, και παρουσιάστηκαν τα αποτελέσματα της εκτέλεσής του πάνω στα δημοφιλέστερα προβλήματα αναφοράς για το πρόβλημα VRPTW, καθώς και η σύγκριση των αποτελεσμάτων με τις τρέχουσες βέλτιστες λύσεις της βιβλιογραφίας. Τα αποτελέσματα είναι ενθαρρυντικά για την περεταίρω έρευνα προς την κατεύθυνση του συνδυασμού των δύο αλγορίθμων.

Προτείνονται κάποιες κατευθύνσεις για περεταίρω έρευνα, τόσο απομονωμένα για τον κάθε αλγόριθμο χωριστά, όσο και για τον τρόπο που συνδυάζονται οι δύο αλγόριθμοι. Αρχικά, προτείνεται η περεταίρω μελέτη κατασκευαστικών μεθόδων, μεθόδων τοπικής αναζήτησης και ανανέωσης των φερομονών για τον αλγόριθμο ACO οι οποίες θα πρέπει να μελετηθούν κατά την εκτέλεση του προτεινόμενου υβριδικού αλγορίθμου. Επιπρόσθετα, προτείνεται η μελέτη επιπλέον μεθόδων κωδικοποίησης της λύσης του προβλήματος VRPTW σε πολυδιάστατο διάνυσμα για την χρήση της από τον αλγόριθμο PSO και διαφορετικών εκδοχών του αλγορίθμου PSO που χρησιμοποιούνται για την επίλυση προβλημάτων ακέραιου προγραμματισμού όπως η παρούσα μοντελοποίηση του προβλήματος VRPTW. Επίσης, προτείνεται η συστηματική παραμετροποίηση του προτεινόμενου αλγορίθμου για την εύρεση των βέλτιστων παραμέτρων για την κάθε κατηγορία προβλήματος, αλλά και η

δοκιμή περισσότερων συνδυασμών των δύο αλγορίθμων. Επιπλέον, προτείνεται η επιλογή των βέλτιστων από τις παραπάνω μεθόδους, ή η κατασκευή μεθόδου προσαρμογής του αλγορίθμου ανάλογα με το προς επίλυση πρόβλημα ή κατά τη διάρκεια της εκτέλεσης του αλγορίθμου με χρήση κάποιου μηχανισμού μνήμης.

Τέλος, χρήσιμη κρίνεται η ανάπτυξη εργαλείων ανάκτησης δεδομένων που χαρακτηρίζουν την πορεία της εκτέλεσης του υβριδικού αλγορίθμου, όπως η εξέλιξη των φερομονών και του κόστους της τρέχουσας βέλτιστης λύσης κατά τη διάρκεια της εκτέλεσης των αλγορίθμων.

8 Βιβλιογραφία

Aarts, E. H., & Laarhoven, v. P. (1985). Statistical cooling: A general approach to combinatorial optimization problems. *Philips Journal of Research, 40*(4), 193.

Amberg, A., Domschke, W., & Voß, S. (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., Savelsbergh, M. W., & Speranza, M. G. (2006). Worst-case analysis for split delivery vehicle routing problems. *Transportation Science, 40*(2), 226-234.

Archetti, C., & Speranza, M. G. (2008). The split delivery vehicle routing problem: A survey *The vehicle routing problem: latest advances and new challenges* (pp. 103-122): Springer.

Archetti, C., Speranza, M. G., & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science, 40*(1), 64-73.

Archetti, C., Speranza, M. G., & Savelsbergh, M. W. (2008). An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science, 42*(1), 22-31.

Baldacci, R., & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming, 120*(2), 347-380.

Battarra, M., Monaci, M., & Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research, 36*(11), 3041-3050. doi: http://dx.doi.org/10.1016/j.cor.2009.02.008

Belmecheri, F., Prins, C., Yalaoui, F., & Amodeo, L. (2010). *Particle swarm optimization to solve the vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows.* Paper presented at the Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on.

Belmecheri, F., Prins, C., Yalaoui, F., & Amodeo, L. (2012). 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. doi: 10.1007/s10845-012-0627-8

Bertsimas, D. (1988). *Probabilistic combinatorial optimization problems.* Massachusetts Institute of Technology.

Bertsimas, D. J. (1992). A vehicle routing problem with stochastic demand. *Operations research, 40*(3), 574-585.

Boudia, M., Prins, C., & Reghioui, M. (2007). An effective memetic algorithm with population management for the split delivery vehicle routing problem *Hybrid Metaheuristics* (pp. 16-30): Springer.

Brandão, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research, 38*(1), 140-151.

Brandão, J., & Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research, 100*(1), 180-191.

Bratton, D., & Kennedy, J. (2007). *Defining a standard for particle swarm optimization.* Paper presented at the Swarm Intelligence Symposium, 2007. SIS 2007. IEEE.

Bräysy, O., & Gendreau, M. (2005). Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science, 39*(1), 104-118. doi: 10.1287/trsc.1030.0056

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An improved ant System algorithm for thevehicle Routing Problem. *annals of Operations Research, 89*, 319-328.

Cattaruzza, D., Absi, N., Feillet, D., Guyon, O., & Libeaut, X. (2013). *The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates.* Paper presented at the 10th Metaheuristics International Conference (MIC 2013).

Cattaruzza, D., Absi, N., Feillet, D., & Vidal, T. (2014). A memetic algorithm for the Multi Trip Vehicle Routing Problem. *European Journal of Operational Research, 236*(3), 833-848. doi: http://dx.doi.org/10.1016/j.ejor.2013.06.012

Chen, A.-l., Yang, G.-k., & Wu, Z.-m. (2006). Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University SCIENCE A, 7*(4), 607-614. doi: 10.1631/jzus.2006.A0607

Chen, S., Golden, B., & Wasil, E. (2007). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks, 49*(4), 318-329.

Christiansen, C. H., & Lysgaard, J. (2007). A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters, 35*(6), 773-781.

Christofides, N. (1976). The vehicle routing problem. *RAIRO-Operations Research-Recherche Opérationnelle, 10*(V1), 55-70.

Christofides, N., & Beasley, J. E. (1984). The period routing problem. *Networks, 14*(2), 237-256.

Clarke, G. u., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research, 12*(4), 568-581.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). *Distributed optimization by ant colonies.* Paper presented at the Proceedings of the first European conference on artificial life.

Contardo, C., & Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization, 12*, 129-146.
Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., & Vigo, D. (2006). Vehicle routing. *Transportation, handbooks in operations research and management science, 14*, 367-428.
Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America, 2*(4), 393-410.
Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*(1), 80-91.
Demir, E., Bektaş, T., & Laporte, G. (2014). A review of recent research on green road freight transportation. *European Journal of Operational Research, 237*(3), 775-793. doi: 10.1016/j.ejor.2013.12.033
Dorigo, M., Maniezzo, V., & Colorni, A. (1991). The ant system: An autocatalytic optimizing process: Technical report.
Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 26*(1), 29-41.
Dorigo, M., Maniezzo, V., Colorni, A., & Maniezzo, V. (1991). Positive feedback as a search strategy.
Dror, M., Laporte, G., & Trudeau, P. (1989). Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science, 23*(3), 166-176.
Dror, M., & Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, 141-145.
Dror, M., & Trudeau, P. (1990). Split delivery routing. *Naval Research Logistics (NRL), 37*(3), 383-402.
Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering, 57*(4), 1472-1483. doi: 10.1016/j.cie.2009.05.009
Escobar, J. W., Linfati, R., Toth, P., & Baldoquin, M. G. (2014). A hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem. *Journal of Heuristics*, 1-27.
Faigle, U., & Kern, W. (1992). Some convergence results for probabilistic tabu search. *ORSA Journal on Computing, 4*(1), 32-37.
Ferrucci, F., Bock, S., & Gendreau, M. (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. doi: 10.1016/j.ejor.2012.09.016
Fisher, M. L. (1985). An applications oriented guide to Lagrangian relaxation. *Interfaces, 15*(2), 10-21.
Fleischmann, B. (1990). *The vehicle routing problem with multiple use of the vehicles*.
Fleszar, K., Osman, I. H., & Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research, 195*(3), 803-809. doi: 10.1016/j.ejor.2007.06.064
Foisy, C., & Potvin, J.-Y. (1993). Implementing an insertion heuristic for vehicle routing on parallel hardware. *Computers & Operations Research, 20*(7), 737-745.
Francis, P. M., Smilowitz, K. R., & Tzur, M. (2008). The period vehicle routing problem and its extensions *The vehicle routing problem: latest advances and new challenges* (pp. 73-102): Springer.
Fu, Z., Eglese, R., & Li, L. (2006). A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society, 57*(8), 1018-1018.
Fu, Z., Eglese, R., & Li, L. Y. (2005). A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society, 56*(3), 267-274.
Gajpal, Y., & Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research, 36*(12), 3215-3223.
Gendreau, M., Guertin, F., Potvin, J.-Y., & Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science, 33*(4), 381-390.
Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, É. D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research, 26*(12), 1153-1173. doi: http://dx.doi.org/10.1016/S0305-0548(98)00100-2
Gendreau, M., Laporte, G., & Séguin, R. (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., & Johnson, J. G. (1976). Multi-terminal vehicle-dispatch algorithm. *Omega, 4*(6), 711-718.
Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533-549.
Golden, B. L., Magnanti, T. L., & Nguyen, H. Q. (1975). Implementing vehicle routing algorithms: DTIC Document.
Gong, Y.-J., Zhang, J., Liu, O., Huang, R.-Z., Chung, H.-H., & Shi, Y.-H. (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.
Gulczynski, D., Golden, B., & Wasil, E. (2011). The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering, 61*(3), 794-804.

Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of operations research, 13*(2), 311-329.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press.

Ioannou, G., Kritikos, M., & Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society, 52*(5), 523-537.

Jepsen, M., Petersen, B., Spoorendonk, S., & Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations research, 56*(2), 497-511.

Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization.* Paper presented at the Proceedings of IEEE international conference on neural networks.

Kenyon, A. S., & Morton, D. P. (2003). Stochastic vehicle routing with random travel times. *Transportation Science, 37*(1), 69-82.

Kuo, Y., & Wang, C.-C. (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., Prins, C., & Reghioui, M. (2008). A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations Research, 42*(03), 415-431.

Labadie, N., & Prins, C. (2012). Vehicle routing nowadays: compact review and emerging problems *Production systems and supply chain management in emerging countries: best practices* (pp. 141-166): Springer.

Laporte, G., Louveaux, F., & Mercure, H. (1992). The vehicle routing problem with stochastic travel times. *Transportation Science, 26*(3), 161-170.

Laporte, G., Louveaux, F. V., & Van Hamme, L. (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations research, 50*(3), 415-423.

Laporte, G., Toth, P., & Vigo, D. (2013). Vehicle routing: historical perspective and recent contributions. *EURO Journal on Transportation and Logistics, 2*(1-2), 1-4. doi: 10.1007/s13676-013-0020-6

Lenstra, J. K., & Kan, A. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11*(2), 221-227.

Leung, S. C. H., Zhang, Z., Zhang, D., Hua, X., & Lim, M. K. (2013). A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research, 225*(2), 199-210. doi: http://dx.doi.org/10.1016/j.ejor.2012.09.023

Li, F., Golden, B., & Wasil, E. (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Operations Research, 34*(10), 2918-2930.

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on, 10*(3), 281-295.

Lima, C. M. R. R., Goldbarg, M. C., & Goldbarg, E. F. G. (2004). A Memetic Algorithm for the Heterogeneous Fleet Vehicle Routing Problem. *Electronic Notes in Discrete Mathematics, 18*(0), 171-176. doi: http://dx.doi.org/10.1016/j.endm.2004.06.027

Machado, P., Tavares, J., Pereira, F. B., & Costa, E. (2002). *Vehicle Routing Problem: Doing It The Evolutionary Way.* Paper presented at the GECCO.

Maffioli, F. (2003). The vehicle routing problem: A book review. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1*(2). doi: 10.1007/s10288-003-0013-7

Marinakis, Y., Marinaki, M., & Dounias, G. (2010). A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence, 23*(4), 463-472. doi: 10.1016/j.engappai.2010.02.002

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., & Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research, 37*(11), 1886-1898.

Moghaddam, B. F., Ruiz, R., & Sadjadi, S. J. (2012). Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm. *Computers & Industrial Engineering, 62*(1), 306-317. doi: 10.1016/j.cie.2011.10.001

Montemanni, R., Gambardella, L. M., Rizzoli, A. E., & Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization, 10*(4), 327-343.

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*(4), 724-737. doi: 10.1016/j.cor.2009.06.022

Novoa, C., & Storer, R. (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., & Hanshar, F. T. (2009). Using genetic algorithms for multi-depot vehicle routing *Bio-inspired algorithms for the vehicle routing problem* (pp. 77-99): Springer.

Pang, W., Wang, K.-p., Zhou, C.-g., & Dong, L.-j. (2004). *Fuzzy discrete particle swarm optimization for solving traveling salesman problem.* Paper presented at the Computer and Information Technology, 2004. CIT'04. The Fourth International Conference on.

Park, Y., & Hong, S. (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.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft, 58*(1), 21-51.

Petch, R. J., & Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics, 133*(1–3), 69-92. doi: http://dx.doi.org/10.1016/S0166-218X(03)00434-7

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1-11. doi: 10.1016/j.ejor.2012.08.015

Pillac, V., Guéret, C., & Medaglia, A. (2010). Dynamic vehicle routing: State of the art and prospects: Technical report, École des Mines de Nantes, France.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence, 1*(1), 33-57.

Powell, W. B. (1986). A stochastic model of the dynamic vehicle allocation problem. *Transportation Science, 20*(2), 117-129.

Prins, C. (2002). Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms, 1*(2), 135-150.

Prins, C. (2009). Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence, 22*(6), 916-928.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science, 14*(2), 130-154.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. *annals of Operations Research, 61*(1), 143-164.

Repoussis, P. P., Tarantilis, C. D., Bräysy, O., & Ioannou, G. (2010). A hybrid evolution strategy for the open vehicle routing problem. *Computers & Operations Research, 37*(3), 443-455. doi: 10.1016/j.cor.2008.11.003

Repoussis, P. P., Tarantilis, C. D., & Ioannou, G. (2009). Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. *Evolutionary Computation, IEEE Transactions on, 13*(3), 624-647.

Roberts, D., & Hadjiconstantinou, E. (1998). *A computational approach to the vehicle routing problem with stochastic demands.* Paper presented at the Computational engineering in systems applications: 16th European conference on operational research (EURO XVI), Brussels, Belgium, IEEE.

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

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research, 171*(3), 750-775.

Salhi, S., & Petch, R. (2007). A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms, 6*(4), 591-613.

Sariklis, D., & Powell, S. (2000). A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 564-573.

Savelsbergh, M. W. (1985). Local search in routing problems with time windows. *annals of Operations Research, 4*(1), 285-305.

Şen, A., & Bülbül, K. (2008). A survey on multi trip vehicle routing problem.

Shen, Z., Ordónez, F., & Dessouky, M. M. (2009). The stochastic vehicle routing problem for minimum unmet demand *Optimization and logistics challenges in the enterprise* (pp. 349-371): Springer.

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., Mungwattana, A., Janssens, G. K., & Manisri, T. (2014). *Heterogeneous VRP Review and Conceptual Framework.* Paper presented at the Proceedings of the International MultiConference of Engineers and Computer Scientists.

Subramanian, A., Drummond, L. M. d. A., Bentes, C., Ochi, L. S., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research, 37*(11), 1899-1911.

Subramanian, A., Penna, P. H. V., Uchoa, E., & Ochi, L. S. (2012). A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem. *European Journal of Operational Research, 221*(2), 285-295. doi: http://dx.doi.org/10.1016/j.ejor.2012.03.016

Subramanian, A., Uchoa, E., & Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research, 40*(10), 2519-2531. doi: 10.1016/j.cor.2013.01.013

Sungur, I., Ordónez, F., & Dessouky, M. (2008). A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions, 40*(5), 509-523.

Taillard, É. D., Gambardella, L. M., Gendreau, M., & Potvin, J.-Y. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research, 135*(1), 1-16.

Tillman, F. A. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science, 3*(3), 192-204.

Tillman, F. A., & Cain, T. M. (1972). An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science, 18*(11), 664-682.

van Hemert, J. I., & La Poutré, J. A. (2004). *Dynamic routing problems with fruitful regions: Models and evolutionary computation.* Paper presented at the Parallel Problem Solving from Nature-PPSN VIII.

Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., & Shapiro, A. (2003). The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications, 24*(2-3), 289-333.

Wang, K.-P., Huang, L., Zhou, C.-G., & Pang, W. (2003). *Particle swarm optimization for traveling salesman problem.* Paper presented at the Machine Learning and Cybernetics, 2003 International Conference on.

Wang, W., Wu, B., Zhao, Y., & Feng, D. (2006). Particle swarm optimization for open vehicle routing problem *Computational Intelligence* (pp. 999-1007): Springer.

Waters, C. D. J. (1989). Vehicle-scheduling problems with uncertainty and omitted customers. *Journal of the Operational Research Society*, 1099-1108.

Wilson, N. H., & Colvin, N. J. (1977). *Computer control of the Rochester dial-a-ride system*: Massachusetts Institute of Technology, Center for Transportation Studies.

Wren, A., & Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 333-344.

Yanik, S., Bozkaya, B., & deKervenoael, R. (2014). A new VRPPD model and a hybrid heuristic solution approach for e-tailing. *European Journal of Operational Research, 236*(3), 879-890. doi: 10.1016/j.ejor.2013.05.023

Yu, B., & Yang, Z. Z. (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., & Kiranoudis, C. T. (2010). An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. *Computers & Operations Research, 37*(4), 712-723.

Zachariadis, E. E., & Kiranoudis, C. T. (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.

Zhang, L.-p., Yu, H.-j., & Hu, S.-x. (2005). Optimal choice of parameters for particle swarm optimization. *Journal of Zhejiang University SCIENCE, 6A*(6), 528-534. doi: 10.1631/jzus.2005.A0528

Zhong, W.-l., Zhang, J., & Chen, W.-n. (2007). *A novel discrete particle swarm optimization to solve traveling salesman problem.* Paper presented at the Evolutionary Computation, 2007. CEC 2007. IEEE Congress on.