**ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΩΝ** **ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΠΕΡΙΟΡΙΣΜΕΝΗ ΧΩΡΗΤΙΚΌΤΗΤΑ**

**ΓΕΡΟΝΤΖΟΣ ΝΕΚΤΑΡΙΟΣ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

Επιβλέπων Καθηγητής Άγγελος Σιφαλέρας

Εξεταστές Σαμαράς Νικόλαος, Χρήστου - Βαρσακέλης Δημήτριος

Τμήμα Εφαρμοσμένης Πληροφορικής

Πανεπιστήμιο Μακεδονίας

Θεσσαλονίκη

Σεπτέμβριος 2023

Copyright ⓒ ΓΕΡΟΝΤΖΟΣ ΝΕΚΤΑΡΙΟΣ, έτος 2023

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Εφαρμοσμένης Πληροφορικής

του Πανεπιστημίου Μακεδονίας δεν υποδηλώνει απαραιτήτως και αποδοχή των

απόψεων του συγγραφέα εκ μέρους του Τμήματος.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία αποτελεί ένα ιδιαίτερα σημαντικό κομμάτι των προπτυχιακών μου σπουδών, καθώς μου έδωσε τη δυνατότητα να διευρύνω τους πνευματικούς μου ορίζοντες, αλλά και σε γνώσεις πέρα από αυτές που προσκόμισα στο πλαίσιο της ακαδημαϊκής μου πορείας. Για την περάτωση της συγκεκριμένης εργασίας, πέρα από προσωπική προσπάθεια και αφοσίωση ώστε να βγει ένα ολοκληρωμένο αποτέλεσμα, οφείλω να ευχαριστήσω ορισμένους ανθρώπους που στάθηκαν δίπλα μου.

Αρχικά, οφείλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Σιφαλέρα Άγγελο για την αμέριστη υποστήριξη και καθοδήγηση που μου παρείχε όλους αυτούς τους μήνες προσπάθειας, αλλά και καθ’ όλη τη διάρκεια των προπτυχιακών μου σπουδών. Έπειτα, χρωστάω ένα μεγάλο ευχαριστώ στην οικογένεια μου, στην κοπέλα μου και στους φίλους μου, που δεν έπαψαν να με στηρίζουν ώστε να πραγματοποιήσω τους στόχους μου στο μέγιστο δυνατό.

Αφιερώνεται σε όλους τους ανθρώπους που με στήριξαν

και στην

αείμνηστη καθηγήτρια μου Βιβή Σαλτάρη.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία παρουσιάζεται το πρόβλημα δρομολόγησης οχημάτων με περιορισμένη χωρητικότητα (CVRP) και ο τρόπος που τέσσερις διαφορετικοί λύτες προσεγγίζουν το πρόβλημα χρησιμοποιώντας διάφορες μεθόδους επίλυσης και βελτιστοποίησης. Οι λύτες που χρησιμοποιήθηκαν είναι τo Gurobi Optimization, LocalSolver, VRPy και Google OR-Tools, οι οποίοι με τη χρήση αλγορίθμων, ευρετικών, μεταευρετικών μεθόδων αλλά και μαθηματικών μοντέλων καταφέρνουν να επιλύσουν τα διάφορα προβλήματα CVRP σε προκαθορισμένο χρόνο 120 δευτερολέπτων. Οι παραπάνω λύτες αναλύονται ως προς τις συγκεκριμένες μεθόδους που χρησιμοποιούν για την εύρεση λύσεων και τέλος συγκρίνουμε τα αποτελέσματα τους ώστε να βρούμε τον βέλτιστο λύτη για το συγκεκριμένο σύνολο προβλημάτων.

Όλα τα προβλήματα CVRP αντλήθηκαν από την βιβλιοθήκη CVRPLIB, η οποία περιέχει πολλαπλά προβλήματα που διαφέρουν μεταξύ τους ως προς το μέγεθος τους, το πλήθος των οχημάτων που χρησιμοποιούνται, καθώς και η χωρητικότητα των οχημάτων. Πέρα από τα δεδομένα των προβλημάτων, παρέχονται και τα αντίστοιχα αρχεία με τις βέλτιστες λύσης τους, ώστε να μπορούμε να κρίνουμε αν οι δικές μας λύσεις έχουν νόημα και αν τελικά είναι κοντά στην πραγματικότητα.

Τέλος παρουσιάζονται τα συμπεράσματα από τη συγκεκριμένη ανάλυση, καθώς και οι βλέψεις για μελλοντική έρευνα.

ΠΕΡΙΕΧΟΜΕΝΑ

1 Εισαγωγή . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

1.1 Προβλήματα VRP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

1.2 CVRP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

1.3 VRP with time windows . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

1.4 VRP with pickups and deliveries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

1.5 Heterogeneous feet VRP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

1.6 VRP with multiple depots and collaborative VRP . . . . . . . . . . . . . 13

1.7 Green VRP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

1.8 CVRP και Logistics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

1.9 Sustainability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2 Mathematical Formulations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.1 Miller–Tucker–Zemlin (MTZ) . . . . . . . . . . . . . . . . . . . . . . . . 17

2.2 Capacity-Constrained Minimum Spanning Tree (CCMST). . . . . . . . . 18

2.3 The Split Delivery Vehicle Routing Problem (SDVRP). . . . . . . . . . 19

3 Heuristics and Metaheuristics . . . . . . . . . . . . . . . . . . . . . . . . . . . .21

3.1 The Clarke and Wright savings heuristic. . . . . . . . . . . . . . . . . . 22

3.2 Nearest Neighbor. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

3.3 Tabu Search. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

3.4 Genetic Algorithms. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

3.5 Ant Colony Optimization. . . . . . . . . . . . . . . . . . . . . . . 26

3.6 Simulated Annealing . . . . . . . . . . . . . . . . . . . . . . . . . . 27

4 Λύτες προβλημάτων . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

4.1 Google OR-Tools . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

4.2 Gurobi Optimization . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

4.3 VRPy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

4.4 LocalSolver . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

4.5 TSPLIB95 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

5 Συλλογές Μετροπροβλημάτων. . . . . . . . . . . . . . . . . . . . . . . . . 39

5.1 TSPLIB . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

5.2 CVRPLIB . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

5.3 Παράδειγμα αρχείου. . . . . . . . . . . . . . . . . . . . . . . . . . 44

6 Αποτελέσματα. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

6.1 Gurobi Optimization . . . . . . . . . . . . . . . . . . . . . . . . 48

6.2 VRPy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

6.3 Local Solver . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

6.4 Google OR-Tools . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

6.5 Βέλτιστες λύσεις και σύγκριση

7 Συμπεράσματα και μελλοντική έρευνα . . . . . . . . . . . . . . . . . . . 56

8 Βιβλιογραφία . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

**1 Εισαγωγή**

**1.1 Προβλήματα VRP**

Το πρόβλημα δρομολόγησης οχημάτων VRP εμφανίστηκε για πρώτη φορά το 1954 απο τους G. Dantzig, R. Fulkerson και S. Johnson, οι οποίοι ήταν αυτοί που έγραψαν την πρώτη βιβλιογραφία σχετικά με το συγκεκριμένο πρόβλημα και ανέπτυξαν τον πρώτο αλγόριθμο για τη μεταφορά πετρελαίου σε έναν σταθμό. Λίγα χρόνια αργότερα το 1964, οι Clarke και Wright έβαλαν για πρώτη φορά περισσότερα από ένα οχήματα στην επίλυση του προβλήματος και πρότειναν έναν αλγόριθμο αρκετα βελτιωμένο από αυτόν των Dantzig, Fulkerson και Johnson.

Όπως αναφέρει και στην έρευνα του ο μαθηματικός και καθηγητής Νίκος Χριστοφίδης, το πρόβλημα δρομολόγησης οχημάτων είναι μια γενική ονομασία που δίνεται σε μια ολόκληρη κατηγορία προβλημάτων που περιλαμβάνουν την επίσκεψη πελάτη από όχημα. Το πρόβλημα προέρχεται από την παράδοση των αγαθών στους πελάτες μέσω οχημάτων, με σκοπό την εύρεση της βέλτιστης απόστασης ώστε να μειωθεί ο χρόνος παράδοσης, το κόστος μεταφοράς ή να ικανοποιηθούν δοσμένοι περιορισμοί. Το πρόβλημα ωστόσο, δεν περιορίζεται μόνο στην παράδοση αγαθών, αλλά και στη συλλογή αυτών. Μπορεί να αφορά, για παράδειγμα, τη βέλτιστη διαδρομή που πρέπει να κάνει ένα σχολικό λεωφορείο για να πάρει όλα τα παιδιά που θα μεταφερθούν στο σχολείο. Πολλά συστήματα διαχείρισης αποθηκών WMS, έχουν ενσωματώσει αλγορίθμους εύρεσης της βέλτιστης διαδρομής που πρέπει να διανύσει ο υπάλληλος στο χώρο της αποθήκης, ώστε να διανύσει τη μικρότερη δυνατή απόσταση, σε ελάχιστο χρόνο.

Λόγω όλων αυτών των περιορισμών, το πρόβλημα δρομολόγησης έχει χωριστεί σε πολλαπλές κατηγορίες, ανάλογα με τις συνθήκες που καλείται να ικανοποιήσει:

* **Vehicle Routing Problem with Profits (VRPP)**: Στο συγκεκριμένο πρόβλημα, στόχος δεν είναι η επίσκεψη όλων των κόμβων-πελατών, αλλά η επίσκεψη αυτών που θα μεγιστοποιήσουν το συνολικό κέρδος, υπό τον περιορισμό ωστόσο κάποιων χρονικών παραθύρων.
* **Vehicle Routing Problem with Pickup and Delivery (VRPPD)**: Σε αυτή την περίπτωση, ένα συγκεκριμένο πλήθος εμπορευμάτων πρέπει να μεταφερθεί σε μία συγκεκριμένη τοποθεσία, με στόχο την εύρεση της βέλτιστης διαδρομής των οχημάτων που θα συμμετέχουν.
* **Vehicle Routing Problem with Time Windows (VRPTW)**: Όπως αναφέρθηκε παραπάνω, τα οχήματα πολλές φορές οφείλουν να ικανοποιήσουν ορισμένους περιορισμούς, ένας εκ των οποίων είναι χρονικά παράθυρα, δηλαδή τα οχήματα

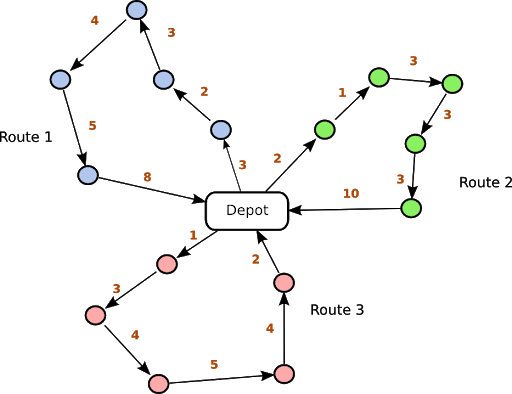
πρέπει να επισκεφθούν τους κόμβους μέσα σε ένα χρονικό περιθώριο, το οποίο όμως μπορεί να μην έχει λύση.

* **Capacitated Vehicle Routing Problem (CVRP)**: Στο συγκεκριμένο τύπο προβλήματος, τα οχήματα έχουν σταθερή χωρητικότητα φορτίου, γεγονός που ίσως αναγκάζει τα οχήματα να επιστρέφουν στην αφετηρία για ανατροφοδότηση.
* **Vehicle Routing Problem with Multiple Trips (VRPMT)**: Τα οχήματα μπορούν να διανύσουν περισσότερες από μία διαδρομές.
* **Open Vehicle Routing Problem (OVRP)**: Τα οχήματα, σε αντίθεση με τις άλλες κατηγορίες, δεν οφείλουν να επιστρέψουν στην αφετηρία.
* **Multi-Depot Vehicle Routing Problem (MDVRP)**: Αρκετά σύνηθες στην πραγματικότητα είναι η περίπτωση που ο στόλος των οχημάτων δεν έχει σταθερή αφετηρία και τέρμα.
* **Vehicle Routing Problem with Transfers (VRPWT)**: Τα αγαθά μπορούν να μεταφέρονται από όχημα σε όχημα πριν φτάσουν στον τελικό προορισμό.

Στην παρούσα εργασία, θα ασχοληθούμε με την κατηγορία CVRP και θα μελετήσουμε τις λύσεις που προκύπτουν με τη χρήση τεσσάρων διαφορετικών solvers. To Gurobi optimization, το Local Solver, το VRPy καθώς και το Google OR-Tools.

**1.2 CVRP**

Το CVRP ανήκει στην κατηγορία του VRP, όπου ένας συγκεκριμένος αριθμός οχημάτων πρέπει να εξυπηρετήσει ένα συγκεκριμένο αριθμό πελατών-κόμβων. Ο σημαντικότερος περιορισμός αυτού του τύπου προβλημάτων είναι πως κάθε όχημα έχει περιορισμένη χωρητικότητα, επομένως ένας κόμβος μπορεί να δεχθεί περισσότερες από μία επισκέψεις. Επίσης, κάνουμε την παραδοχή πως όλα τα οχήματα έχουν μία συγκεκριμένη αφετηρία (depot) και στόχος είναι να βρεθεί ένα σύνολο βέλτιστων διαδρομών ώστε να ικανοποιηθεί μια συγκεκριμένη συνθήκη, όπως η μείωση του κόστους μεταφοράς.



Το CVRP, επομένως, είναι από μόνο του μία δοκιμασία για μοντέρνες εφαρμογές καθώς ανήκει στα NP-hard προβλήματα. Στην πραγματικότητα λοιπόν αυτά τα προβλήματα έχουν μεγάλη έκταση και απαιτούν πολλή υπολογιστική ώρα για να φτάσουν στη βέλτιστη λύση.

**1.3 VRP with time windows**

Ένα από τα επίσης αρκετά μελετημένα προβλήματα δρομολόγησης, είναι το VRP με χρονικά παράθυρα ως περιορισμό για το οποίο έχουν εφαρμοστεί αρκετοί αλγόριθμοι για την επίλυση του. Οι Ghoseiri and Ghannadpour (2010) αντιμετώπισαν το πρόβλημα ως πολυστοχαστικό καθώς τόσο ο στόλος των οχημάτων, όσο και η συνολική διανυόμενη απόσταση πρέπει να μειωθούν. Χρησιμοποίησαν έναν γενετικό αλγόριθμο που εφαρμόζει την κατάταξη Pareto με την προϋπόθεση να βρεθούν οι μη επικρατέστερες λύσεις.

Απο την άλλη, οι Lei et al. (2011) βρήκαν έναν ευρετικό αλγόριθμο μεγάλης γειτονιάς (large neighborhood heuristic), όπου στην αρχή χρησιμοποιείται ένας ευρετικός αλγόριθμος για την εύρεση μιας ιδανικής λύσης. Σε κάθε επανάληψη διάφορες ευρετικές μέθοδοι, αφαιρούν πελάτες (κόμβους) και “καταστρέφουν” την τρέχουσα λύση, πριν οι μέθοδοι εισαγωγής επιδιορθώσουν αυτές τις κατεστραμμένες λύσεις. Η επιλογή των ευρετικών μεθόδων για την αφαίρεση και την διαγραφή γίνεται πιθανοτικά. ενώ ο συνδυασμός τους προσφέρει ευρεία εξερεύνηση στον χώρο των εφικτών λύσεων.

**1.4 VRP with pickups and deliveries**

Οι εταιρείες εφοδιαστικής αλυσίδας, πέρα από τη μεταφορά των προϊόντων οφείλουν να συλλέγουν, πολλές φορές, άλλα προϊόντα προς επιστροφή. Από πολλούς ερευνητές τα reverse logistics μελετώνται είτε θεωρώντας πως η φόρτωση του εμπορεύματος προς επιστροφή γίνεται παράλληλα με την παράδοση, είτε σε δύο διαφορετικούς χρόνους. Ο Çatay (2010) πρότεινε έναν ACO αλγόριθμο που χρησιμοποιεί το Nearest-Neighbor Heuristic (NNH) για την εύρεση της αρχικής λύσης και στη συνέχεια εφαρμόζεται ένας νέος αλγόριθμος με βάση την ελαχιστοποίηση των πόρων.

**1.5 Heterogeneous fleet VRP**

Το HFVRP είναι μία παραλλαγή του κλασικού CVRP με την διαφορά ότι ο στόλος των οχημάτων περιλαμβάνει διαφορετικού τύπου οχήματα. Οι Subramanian et al. (2012) πρότειναν έναν υβριδικό αλγόριθμο που αποτελείται από επαναλαμβανόμενη τοπική αναζήτηση και μία διατύπωση διαχωρισμού συνόλου (SP). Το μοντέλο SP επιλύεται χρησιμοποιώντας έναν λύτη μικτού ακέραιου προγραμματισμού (MIP) που καλεί τον ILS heuristic κατα τη διάρκεια της εκτέλεσης του. Ακόμα, οι Belloso et al. (2019) ανέλυσαν το HFVRP with backhauls και πρότειναν μία λύση βασισμένη σε multi-start biased-randomized heuristic. Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί μία επαναληπτική μέθοδο που βασίζεται σε μικρότερες εκδοχές ενός δεδομένου προβλήματος και στη συνέχεια χρησιμοποιεί αυτές τις λύσεις ως αρχικές-μερικές λύσεις για το αρχικό πρόβλημα.

**1.6 VRP with multiple depots and collaborative VRP**

Το MDVRP είναι μία επέκταση του VRP με την επέκταση πολλαπλών αποθηκών. Η προσέγγιση της επίλυσης απο τους Yu et al. (2011) ήταν η προσθήκη μιας εικονικής κεντρικής αποθήκης, μετατρέποντας το πρόβλημα σε VRP. Την επίλυση του προβλήματος την προσέγγισαν με μια βελτιωμένη “ant colony” optimization διαδικασία, ή αλλιώς αποικίας μυρμηγκιών, η οποία είναι μία πιθανοτική επίλυση που χρησιμοποιείται σε υπολογιστικά προβλήματα για την εύρεση για παράδειγμα βέλτιστων δικτυακών διαδρομών.

Μία ακόμη προσέγγιση του συγκεκριμένου προβλήματος είναι από τους Karakatič και Podgorelec (2015) με τη χρήση γενετικών αλγορίθμων, υποστηρίζοντας πως είναι ισάξιοι με άλλους αλγορίθμους και είναι αποτελεσματικοί σε προβλήματα MDVRP. Κύριο πλεονέκτημα τους σε τέτοια προβλήματα όρισαν πως είναι η γραμμική κλιμάκωση με το μεταβαλλόμενο μέγεθος του προβλήματος και κατ’ επέκταση προτιμάται για την επίλυση μεγάλων NP προβλημάτων, αντί να χρησιμοποιηθούν άλλα, ακριβότερα heuristics.

Τέλος, το 2020 οι Wang et al. για το ίδιο πρόβλημα πρότειναν ένα υβριδικό heuristic μοντέλο, που συνδυάζει τον K-means clustering algorithm, τον Clark-Wright saving algorithm, και τον εκτεταμένο NSGA-II algorithm ο οποίος είναι ένας πολύ στοχαστικός αλγόριθμος. Οι δύο στόχοι που μελετώνται από τους ερευνητές είναι η ελαχιστοποίηση του συνολικού λειτουργικού κόστους, καθώς και η ελαχιστοποίηση του στόλου οχημάτων.

**1.7 Green VRP**

Η κατηγορία των προβλημάτων VRP που σχετίζεται με τη βιωσιμότητα, όπως βλέπουμε και πιο κάτω, το πράσινο και το ηλεκτρικό, ονομάζεται Green VRP. Τα συγκεκριμένα προβλήματα αποτελούνται από οχήματα που χρησιμοποιούν βιώσιμες πηγές ενέργειας, πελάτες-κόμβους προς επίσκεψη και ένα σύνολο σταθμών ανεφοδιασμού με στόχο τόσο να ελαχιστοποιηθεί η συνολική απόσταση, καθώς και η χρήση τους για ανεφοδιασμό. Οι Schneider et al. (2014) αντιμετώπισαν το πρόβλημα με χρήση μικτού στόλου, δηλαδή τόσο με ηλεκτρικά οχήματα όσο και με οχήματα εσωτερικής καύσης. Ως μέθοδο επίλυσης, χρησιμοποίησαν έναν μεικτό ευρετικό αλγόριθμο που εφαρμόζει αλγόριθμο αναζήτησης γειτονιάς (neighborhood search algorithm) με έναν επίσης ευρετικό αλγόριθμο αναζήτησης tabu search.

**1.8 CVRP και Logistics**

Τα CVRP προβλήματα εμφανίζονται κατα κύριο λόγο στην εφοδιαστική αλυσίδα για την μεταφορά των προϊόντων από το εργοστάσιο, στις αποθήκες και στους πελάτες. Στη δρομολόγηση οχημάτων, η χωρητικότητα τους παίζει σημαντικό ρόλο και αποτελεί τον παράγοντα που μελετήθηκε πρώτα από τους ερευνητές, ενώ σε πραγματικά προβλήματα οι περισσότερες εταιρείες διαθέτουν οχήματα με διαφορετική χωρητικότητα για τη βέλτιστη εξυπηρέτηση μιας αποστολής, με το ελάχιστο δυνατό κόστος. Όπως είναι λογικό, τα μικρότερα σε χωρητικότητα οχήματα συνήθως χρησιμοποιούνται για να εξυπηρετούν μέσα σε πόλεις (last mile), ενώ τα μεγαλύτερα χρησιμοποιούνται για μεγαλύτερες αποστάσεις με μεγαλύτερες απαιτήσεις σε χωρητικότητα. Στην πραγματικότητα, πολλές εταιρείες Third-Party Logistics συνεργάζονται με μικρότερες εταιρείες logistics οι οποίες διαχειρίζονται τον στόλο των οχημάτων για αυτούς, με στόχο να επιτύχουν μικρότερο κόστος. Σε αντίθεση με τις περιπτώσεις της συγκεκριμένης εργασίας, τα οχήματα αυτά δεν επιστρέφουν απαραίτητα στο αρχικό depot, αλλά ενδέχεται να έχουν πολλαπλούς τερματικούς σταθμούς και τη συγκεκριμένη εκδοχή του προβλήματος ονομάστηκε Open VRP (OVRP) (Ζαχαριάδης και Κιναρούδης, 2010).

Ωστόσο, η λειτουργία των logistics δεν σταματά στη φάση που τα προϊόντα έχουν παραδοθεί στον πελάτη, αφού στην πράξη συνηθίζεται να υπάρχουν επιστροφές προϊόντων. Η διαδικασία της επιστροφής (VRPB) καθώς και η ταυτόχρονη παράδοση και παραλαβή

(VRPSPD) μελετούν την παράδοση και παραλαβή αγαθών κατα την εκτέλεση των δρομολογίων. Πιο συγκεκριμένα:

* Στην πιο απλή μορφή, στο VRPB η παραλαβή των προϊόντων πραγματοποιείται αφού έχει ολοκληρωθεί πρώτα η παράδοση τους, ενώ στο
* VRPSPD η παραλαβή των επιστροφών πραγματοποιείται ταυτόχρονα με την παράδοση πράγμα μου το καθιστά πιο δύσκολο από το VRPB καθώς πρέπει να μη γίνει υπέρβαση της χωρητικότητας κάθε οχήματος.

**1.9 Sustainability**

Η δημιουργία δικτύων εφοδιασμού, logistics, έχει σημαντικό αντίκτυπο στην παγκόσμια οικονομική δομή των κοινωνιών. Ωστόσο, η συμβατική έμφαση στην εξοικονόμηση κόστους και στη λειτουργική αποτελεσματικότητα έχει συχνά ανεπιθύμητες αρνητικές επιπτώσεις στο περιβάλλον. Η περιβαλλοντική υποβάθμιση και η κλιματική αλλαγή έχουν επιδεινωθεί από τη χρήση ενέργειας, τις εκπομπές και την εκμετάλλευση των πόρων που σχετίζονται με τις δραστηριότητες μεταφοράς και διανομής.

Η υιοθέτηση της βιωσιμότητας στα logistics απαιτεί μια ολοκληρωμένη στρατηγική που στοχεύει στη μείωση των περιβαλλοντικών επιπτώσεων, καλύπτοντας παράλληλα τις ανάγκες μιας δυναμικής αγοράς. Ο τομέας των logistics μπορεί να ελαχιστοποιήσει σημαντικά τις περιβαλλοντικές του επιπτώσεις εφαρμόζοντας πράσινες πρακτικές όπως η βελτιστοποίηση διαδρομής για τη μείωση της κατανάλωσης καυσίμου και των εκπομπών, η χρήση εναλλακτικών πηγών ενέργειας και η εφαρμογή προγραμμάτων κοινής χρήσης αυτοκινήτου.

Η αειφορία στα logistics περιλαμβάνει επίσης ζητήματα εκτός από το περιβάλλον. Περιλαμβάνει την ενθάρρυνση ασφαλών μεθόδων μεταφοράς, την τήρηση της δικαιοσύνης των εργαζομένων και την άσκηση κοινωνικής ευθύνης. Η εφοδιαστική μπορεί να βοηθήσει στη δημιουργία μιας πιο δίκαιης και ηθικά συνειδητοποιημένης κοινωνίας ενθαρρύνοντας επιχειρηματικές πρακτικές χωρίς αποκλεισμούς και ηθικές.

Η πολυπλοκότητα των προβλημάτων δρομολόγησης χωρητικότητας οχημάτων (CVRP) εξετάζεται σε βάθος σε αυτή τη διατριβή και γνωρίζουμε ότι η εργασία μας μπορεί να έχει θετικό αντίκτυπο στις υλικοτεχνικές λειτουργίες στον πραγματικό κόσμο. Εκτός από τη μεγιστοποίηση της χρήσης του στόλου και της εξυπηρέτησης πελατών, θέλουμε να προωθήσουμε ηθικές και υπεύθυνες πρακτικές logistics δημιουργώντας αποτελεσματικούς αλγόριθμους και δημιουργικές λύσεις.

**2 Mathematical Formulations**

Πολλοί λύτες προβλημάτων VRP και των παραλλαγών του συγκεκριμένου προβλήματος, χρησιμοποιούν μαθηματικά μοντέλα τα οποία εφαρμόζοντας τα με συγκεκριμένο τρόπο και συνδέοντας τα με τα δεδομένα του εκάστοτε προβλήματος να βρίσκουν λύση. Είναι απολύτως αναμενόμενο αλλά και επιθυμητό, οι λύσεις ανάμεσα στα μαθηματικά μοντέλα να διαφέρουν τόσο ως προς τον τρόπο που προσεγγίζουν ένα πρόβλημα, αλλά και ως προς την ίδια την λύση καθώς πολλές φορές οι απόκλιση των λύσεων ποικίλει. Τα συγκεκριμένα μαθηματικά μοντέλα χρησιμοποιούνται για να απεικονίσουν σε μαθηματική μορφή προβλήματα σαν το CVRP, αλλά και για άλλα προβλήματα στον χώρο της επιχειρησιακής έρευνας και της συνδυαστικής βελτιστοποίησης. Με τον τρόπο αυτό λοιπόν, δηλαδή μέσω της μαθηματικής απεικόνισης των προβλημάτων, χρησιμοποιούμε τεχνικές μαθηματικής βελτιστοποίησης με στόχο να βρούμε τη βέλτιστη λύση ενός προβλήματος. Στη συγκεκριμένη εργασία, όπως θα δούμε στη συνέχεια, εστιάζουμε στην ελαχιστοποίηση του κόστους από τη μεταφορά αγαθών σε κόμβους πελάτες μέσω

συγκεκριμένου αριθμού στόλου οχημάτων, υπό τον περιορισμό της χωρητικότητας που διαθέτουν τα οχήματα.

**2.1 Miller–Tucker–Zemlin (MTZ)**

Το πρόβλημα δρομολόγησης οχημάτων με χωρητικότητα (CVRP) ορίζεται σε ένα γράφημα G=(V,A), όπου V={v1, …, vn} είναι το σύνολο των κόμβων-πελατών με v1 να απεικονίζει την αποθήκη/depot. Το σύνολο Α απεικονίζει τη σύνδεση μεταξύ των κόμβων για κάθε κόμβο vi∈V⧹{v1} και κάθε τόξο (vi,vj) συνδέεται με ένα κόστος ταξιδιού cij το οποίο στην περίπτωση μας απεικονίζει την απόσταση μεταξύ των κόμβων. Το πρόβλημα ορίζεται από τις παρακάτω συνθήκες:

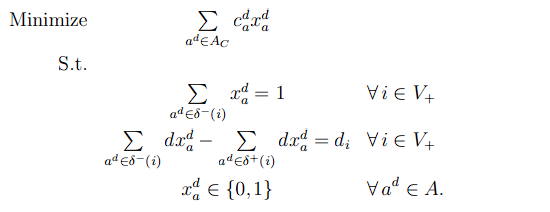
* κάθε όχημα ξεκινά από την αφετηρία και καταλήγει πάλι στην αφετηρία,
* κάθε πελάτης ανήκει σε ακριβώς μία διαδρομή,
* η συνολική ζήτηση μιας διαδρομής δεν υπερβαίνει την χωρητικότητα Q των οχημάτων,
* το συνολικό κόστος των διαδρομών ελαχιστοποιείται.

Αν και υπάρχουν πολλαπλοί τρόποι αναπαράστασης ενός CVRP προβλήματος, στη συγκεκριμένη εκδοχή, το xij είναι μία δυαδική μεταβλητή ίση με 1 αν και μόνο αν το τόξο (vi,vj) ανήκει στην λύση και ένας ακέραιος ui που συνδέεται με κάθε κόμβο ui. Οι μεταβλητές xij ορίζονται αν και μονο αν ισχύει qi+qj⩽Q, με qi να αναπαριστά τη ζήτηση κάθε κόμβου i.

S

**2.2 Capacity-Constrained Minimum Spanning Tree (CCMST)**

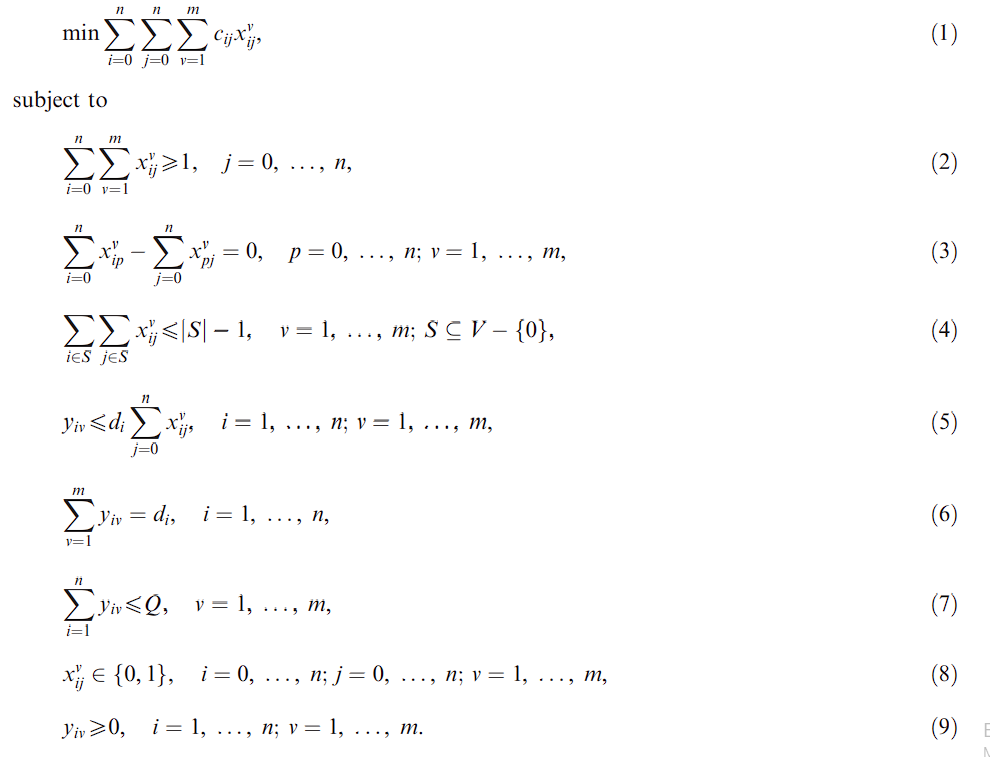
Το “ελάχιστο δέντρο εξάπλωσης” με περιορισμούς χωρητικότητας (Capacity-Constrained Minimum Spanning Tree) αποτελεί μία παραλλαγή του μαθηματικού μοντέλου. Έστω ένας γράφος G = (V,E) με V = {0,1,..., n} κορυφές και m = |E| ακμές. Η σχέση του συγκεκριμένου μοντέλου με τα CVRP προβλήματα είναι η εξής. Η κορυφή 0 αποτελεί τη ρίζα του δέντρου ή αλλιώς την αφετηρία ανάμεσα σε ένα σύνολο κόμβων. Όλες οι υπόλοιπες κορυφές αναπαριστούν τους κόμβους του δικτύου μας και συνδέονται με έναν ακέραιο αριθμό που αναπαριστά τη ζήτηση του συγκεκριμένου κόμβου. Η κορυφή 0 έχει ζήτηση 0, καθώς αποτελεί την αφετηρία. Δεδομένου πως έχουμε μια τιμή C μεγαλύτερη ή ίση της μέγιστης ζήτησης, το CMST στοχεύει στο να βρει ένα υπο δέντρο ,ελάχιστου κόστους, του γράφου G τέτοιο ώστε η συνολική ζήτηση όλων των κορυφών κάθε υποδέντρου να μην υπερβαίνει την τιμή C. Ο Gouveia παρουσίασε την παρακάτω διατύπωση του προβλήματος με περιορισμένη χωρητικότητα, εμπνευσμένος απο το TSP πρόβλημα. Οι δυαδικές τιμές  υποδηλώνουν εάν το τόξο a = (i,j) ανήκει στη βέλτιστη λύση και οτι η συνολική ζήτηση όλων των κορυφών στο υποδέντρο με ρίζα στο j είναι ακριβώς ίση με d. Με απλά λόγια, αν η τιμή αυτή είναι ίση με 1, σημαίνει πως πρέπει να εγκατασταθεί ένας σύνδεσμος χωρητικότητας d, στο τόξο a.



**2.3 The Split Delivery Vehicle Routing Problem (SDVRP)**

Στο πρόβλημα του περιοδεύον πωλητή , ο πωλητής πρέπει να πρέπει να ξεκινήσει τη διαδρομή του από ένα συγκεκριμένο σημείο και να καταλήξει πάλι στο ίδιο σημείο, με το ελάχιστο κόστος. Δεν πρέπει να παραλείψουμε όμως πως σε τέτοια προβλήματα χρησιμοποιείται μόνο ένα όχημα και όχι στόλος οχημάτων. Το SDVRP ορίζεται σε ένα μη κατευθυνόμενο γράφημα G = (V,E) με σύνολο κορυφών V = {0,1,...,n} , όπου το 0 αντιπροσωπεύει την αφετηρία και οι άλλες κορυφές τους πελάτες, με Ε το σύνολο των ακμών. Μία ζήτηση di είναι συνδεδεμένη με κάθε πελάτη i, με τον κόμβο 0 να έχει ζήτηση 0 καθώς αναπαριστά την αφετηρία. Ένα κατώτερο όριο για τον απαραίτητο αριθμό οχημάτων που χρειάζονται για να εξυπηρετήσουν τους πελάτες δίνεται από τη σχέση . Οι Archetti et al. (2009) στην έρευνα τους έδειξαν πως πάντα υπάρχει βέλτιστη λύση που χρησιμοποιεί αριθμό οχημάτων, όχι μεγαλύτερο, από το διπλάσιο της παραπάνω σχέσης, οπότε θεωρούμε πλήθος οχημάτων ίσο με αυτή την τιμή. Η ζήτηση κάθε πελάτη πρέπει να ικανοποιηθεί πλήρως και η συνολική ποσότητα κάθε διαδρομής δεν πρέπει να είναι μεγαλύτερη από το Q, δηλαδή τη συνολική χωρητικότητα κάθε οχήματος.

To SDVRP μπορεί να απεικονιστεί ως εξής:



Οι περιορισμοί (2)-(4) αναπαριστούν τους περιορισμούς δρομολόγησης που έχουμε σε όλα τα προβλήματα. Ο περιορισμός (5) δείχνει πως ο κόμβος i εξυπηρετείται από το όχημα v, αν και μόνο αν το όχημα v επισκέπτεται τον κόμβο i, ο (6) διασφαλίζει πως η ζήτηση κάθε κόμβου εξυπηρετείται πλήρως και ο (7) διασφαλίζει πως η ποσότητα που μεταφέρεται από κάθε όχημα δεν ξεπερνά την ίδια τη χωρητικότητα του οχήματος.

**3 Heuristics and Metaheuristics**

Το Πρόβλημα Δρομολόγησης Οχημάτων (VRP), ένα πρόβλημα συνδυαστικής βελτιστοποίησης, είναι ο εντοπισμός των καλύτερων διαδρομών για να ταξιδέψει ένας στόλος οχημάτων, προκειμένου να εξυπηρετήσει μια ομάδα πελατών, μειώνοντας παράλληλα τα συνολικά έξοδα ή μεγιστοποιώντας την απόδοση του. Λόγω της εγγενούς πολυπλοκότητας του VRP, η εύρεση μιας ακριβούς λύσης μπορεί να είναι δύσκολη υπολογιστικά, ειδικά όταν εμπλέκεται σημαντικός αριθμός καταναλωτών.

Μέσω της γρήγορης ανακάλυψης λύσεων υψηλής ποιότητας, οι ευρετικές μέθοδοι προσφέρουν μια χρήσιμη μέθοδο για την επίλυση του VRP. Οι ευρετικές είναι μέθοδοι για την επίλυση προβλημάτων που βασίζονται στην κοινή λογική, τη διαίσθηση ή την προηγούμενη γνώση για να κατευθύνουν την αναζήτηση απαντήσεων, συχνά ανταλλάσσοντας τη βέλτιστη απόδοση με την αποτελεσματικότητα. Παρέχουν μια αντιστάθμιση μεταξύ της ποιότητας των λύσεων και της υπολογιστικής προσπάθειας, γεγονός που τις καθιστά ιδανικές για την αντιμετώπιση περιπτώσεων VRP μεγάλης κλίμακας.

Η δημιουργία αρχικών λύσεων και η επαναληπτική βελτίωσή τους μέχρι να βρεθεί μια καλή (ή βέλτιστη) λύση είναι ο τρόπος με τον οποίο εφαρμόζονται συνήθως οι ευρετικές μέθοδοι για το VRP. Αυτές οι μέθοδοι επιδιώκουν να βελτιώσουν επαναληπτικά τις διαδρομές αξιοποιώντας τη δομή του προβλήματος και τις μεθόδους τοπικής αναζήτησης, ωστόσο δεν εγγυώνται πως θα βρουν τη βέλτιστη λύση. Για προβλήματα VRP, χρησιμοποιούνται συχνά οι ακόλουθοι ευρετικοί αλγόριθμοι:

* The Clarke and Wright savings heuristic,
* Nearest Neighbor,
* Genetic Algorithms,
* Tabu Search,
* Simulated Annealing,
* Ant Colony Optimization

**3.1 The Clarke and Wright savings heuristic**

Ο αλγόριθμος των Clarke και Wright (Clarke & Wright, 1964) αποτελεί μέχρι και σήμερα από τους πιο γνωστούς και χρησιμοποιείται στην πράξη σε πολλαπλά προβλήματα και χρησιμοποιεί την έννοια της αποταμίευσης. Ο τρόπος επίλυσης είναι ως εξής:

Στην αρχή, ξεκινάμε με ξεχωριστές διαδρομές μεταξύ της αφετηρίας και των πελατών και σε κάθε βήμα συγχωνεύουμε δύο διαδρομές, αν είναι δυνατόν, δημιουργώντας μία ενιαία διαδρομή που τις συνδυάζει. Η απόφαση για συγχώνευση καθορίζεται από την τιμή saving value, η οποία υπολογίζεται αθροίζοντας τις αποστάσεις μεταξύ των πελατών και της αποθήκης σε κάθε διαδρομή και αφαιρώντας παράλληλα την απόσταση μεταξύ των πελατών που εξετάζονται για συγχώνευση. Στην παράλληλη έκδοση του αλγορίθμου, επιλέγουμε τη συγχώνευση, της οποίας το saving value είναι το μεγαλύτερο, ωστόσο στη διαδοχική εκδοχή, συνεχίζεται να επεκτείνεται η δεδομένη διαδρομή έως να μην έχει εφικτή λύση.

**Βήμα 1** (υπολογισμός savings): Υπολογισμός του savings value = + −

για i,j = 1,...,n και i≠j. Δημιουργεί n διαδρομές οχημάτων (0,i,0) για i= 1,...,n

**Βήμα 2** της παράλληλης έκδοσης (καλύτερη δυνατή συγχώνευση): Εκτελούνται τα ακόλουθα, ξεκινώντας από την κορυφή της λίστας αποθήκευσης. Μαθαίνει εάν υπάρχουν δύο μονοπάτια, το ένα από τα οποία έχει τόξο ή ακμή, με ένα save και το άλλο τόξο ή ακμή που περιέχει ποσότητα(i, 0), η οποία μπορεί να συνδυαστεί. Εάν ναι, κάνει αντικατάσταση τα (0,j) και (i, 0) με τα (i, j) για να συνδυαστούν αυτές οι δύο διαδρομές.

**Βήμα 2.1** της διαδοχικής έκδοσης (επέκταση διαδρομής): Παίρνει κάθε πιθανή διαδρομή (0, i,..., j, 0) με τη σειρά. Για να συνδυαστεί η παρούσα διαδρομή με μια άλλη διαδρομή που περιλαμβάνει τόξο ή ακμή (k, 0), ή που περιέχει ακμή (0,), βρίσκει το πρώτο που μπορεί να χρησιμοποιηθεί. Εκτελεί τη συγχώνευση και, στη συνέχεια, επαναλαμβάνει τη διαδικασία για την τρέχουσα διαδρομή.

Ο συγκεκριμένος αλγόριθμος εκτιμάται ιδιαίτερα λόγω της απλότητας και της ταχύτητας του που αναφέραμε παραπάνω, καθώς δεν απαιτεί παραμέτρους και είναι εύκολο να απεικονιστεί με κώδικα, ωστόσο υστερεί σε ακρίβεια καθώς επιτυγχάνει μέτρια βαθμολογία. Κατα τη σύγκριση των αποτελεσμάτων με τις πιο γνωστές τιμές λύσης που προσδιορίστηκαν από τους Taillard και Rochat, η υλοποίηση του αλγορίθμου με την παράλληλη έκδοση ακολουθούμενη από 3-opt (θα το δούμε παρακάτω) έδωσε μέση απόκλιση 6.71%. Το συγκεκριμένο ποσοστό, αν και όχι τέλειο, είναι αποδεκτό.

Συνολικά, ο συγκεκριμένος αλγόριθμος είναι γνωστός για την απλότητα, την ταχύτητα και τη ευκολία του ώστε να αναπαρασταθεί σε μορφή κώδικα. Παρόλο που δεν παράγει βέλτιστες λύσεις, εξακολουθεί να είναι αποδοτικότερος συγκριτικά με άλλες γνωστές λύσεις που προσδιορίζονται σε άλλες μελέτες.

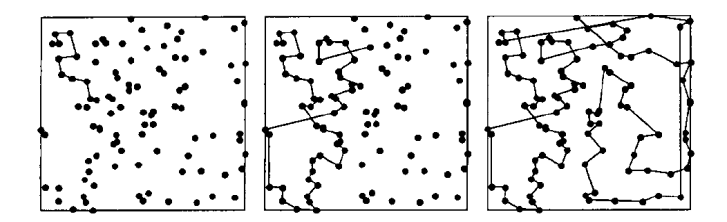
**3.2 Nearest Neighbour**

Ένας από τους πιο γνωστούς αλγορίθμους για την επίλυση προβλημάτων TSP, αλλά και για CVRP σε συνδυασμό με κάποιες προσθήκες, είναι ο αλγόριθμος του πλησιέστερου γείτονα (Nearest Neighbour Algorithm), μελετήθηκε για πρώτη φορά το 1967 από τους Cover και Hart (Cover, T., & Hart, P. , 1967). Λειτουργεί με τον κανόνα να πηγαίνουμε στους διπλανούς πελάτες/κόμβους που δεν έχουμε επισκεφθεί, με τους εξής περιορισμούς:

(i) Ξεκινάμε από την αποθήκη, κάθε κόμβος επισκέπτεται ακριβώς μία φορά και το συνολικό άθροισμα της ζήτησης των επισκεφθέντων κόμβων δεν ξεπερνά τη χωρητικότητα του οχήματος που τους εξυπηρετεί, για την περίπτωση του CVRP.

(ii) Σε περίπτωση που η παραπάνω συνθήκη παραβιαστεί, οφείλουμε να ξεκινήσουμε πάλι από την αρχή και επισκεπτόμαστε διαφορετική διαδρομή από την προηγούμενη.

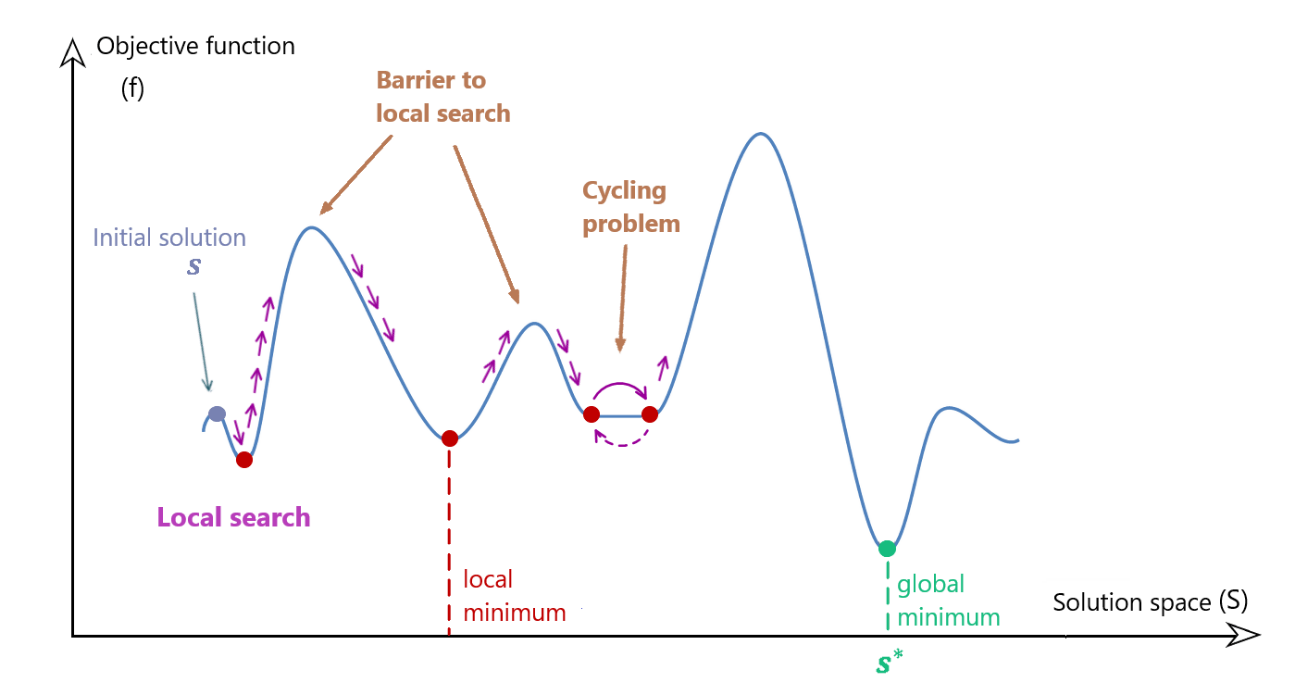
Σε κάθε επανάληψη, οι αναζητήσεις για τον κοντινότερο γείτονα που δεν έχει τοποθετηθεί ακόμα σε διαδρομή, με το που το βρει τον προσθέτει στον τελευταίο του πελάτη, ώστε να δημιουργηθεί η αντίστοιχη διαδρομή. Αυτή η διαδικασία γίνεται ανάμεσα σε όλους τους πελάτες που μπορούν να ανταποκριθούν σε διάφορους περιορισμούς, όπως στη δική μας περίπτωση είναι η χωρητικότητα των οχημάτων.



Πηγή: <https://imada.sdu.dk/u/marco/Teaching/Fall2010/DM204/Slides/dm204-lec31.pdf>

**3.3 Tabu Search**

Ο αλγόριθμος Tabu Search προτάθηκε αρχικά το 1986 από τον Glover (Glover et al. 1986) και αποσκοπεί στην αποφυγή μη αποδεκτών τοπικών βέλτιστων, καθοδηγούμενος από τη μέθοδο τοπικής αναζήτησης. Η μέθοδος αυτή, συνίσταται στη μετάβαση του καλύτερου γείτονα ακόμα και αν αυτό έχει αρνητικές επιπτώσεις στην αντικειμενική συνάρτηση, καθώς αυξάνονται οι πιθανότητες μετακίνησης εκτός τοπικού βέλτιστου. Για να αποφευχθούν τυχόν επαναλήψεις, οι λύσεις που εξετάστηκαν στην αντικειμενική συνάρτηση δηλώνονται tabu για ένα συγκεκριμένο αριθμό επαναλήψεων του αλγορίθμου. Κάθε κίνηση που πραγματοποιείται ονομάζεται tabu μέχρι την επανάληψη (t + θ), όπου θ ο χρόνος που η συγκεκριμένη κίνηση είναι απαγορευμένη. Όλες αυτές οι κινήσεις εισέρχονται σε μία λίστα tabu και κάθε κίνηση που πραγματοποιείται εισέρχεται στο τέλος της λίστας. Στη συνέχεια, με βάση την συγκεκριμένη λύση, δηλαδή το tabu list, ως καλύτερη λύση ορίζεται αυτή με το καλύτερο αποτέλεσμα στην αντικειμενική συνάρτηση. Η διαγραφή στοιχείων από τη λίστα γίνεται συνήθως ακολουθώντας μέθοδο FIFO, όταν ικανοποιηθεί μία δεδομένη συνθήκη. Όταν το μήκος της λίστας είναι μικρό, η αναζήτηση λύσης γίνεται σε μικρές περιοχές του χώρου αναζήτησης, ενώ όταν είναι μεγάλο η αναζήτηση επεκτείνεται σε μεγαλύτερο εύρος.



Πηγή: https://www.baeldung.com/cs/tabu-search

**3.4 Genetic Algorithms**

Πολλές φορές το μέγεθος ενός προβλήματος δεν επιτρέπει σε πιο κλασικές μεθόδους να επιλύσουν ένα τέτοιο πρόβλημα, με αποτέλεσμα να χρειαζόμαστε μία άλλη λύση προκειμένου να πάρουμε κάποιο αποτέλεσμα. Οι γενετικοί αλγόριθμοι, ενδείκνυται

για τέτοιου είδους προβλήματα, ωστόσο δεν μπορούν να εγγυηθούν πως θα βρουν τη βέλτιστη λύση, αλλά μια ικανοποιητική, σε μικρό χρονικό διάστημα.

Τα βασικά σενάρια γενετικών αλγορίθμων αναπτύχθηκαν πρώτη φορά το 1975 από τον Holland, ενώ η πρακτική τους χρήση καταδείχθηκε απο τους De Jong (1975) και Goldberg (1989). Το 1993, οι Blanton και Wainwright εισήγαγαν έναν υβριδικό γενετικό αλγόριθμο με τη χρήση μιας άπληστης ευρετικής μεθόδου. Πιο συγκεκριμένα, ο γενετικός αλγόριθμος αναζητά καλές παραγγελίες πελατών προς εξυπηρέτηση και αφήνει την εύρεση της εφικτής λύσης στον ευρετικό αλγόριθμο. Στόχος του συγκεκριμένου αλγορίθμου είναι να τοποθετήσει τους πελάτες σύμφωνα για παράδειγμα με το χρονικό του παράθυρο, πιο μπροστά στις παραγγελίες σχετικά με άλλους πελάτες που έχουν μεγαλύτερο χρονικό παράθυρο. Ωστόσο, οι συγγραφείς έχουν δοκιμάσει να εφαρμόσουν τη συγκεκριμένη λύση με άλλα κριτήρια προτεραιότητας πέρα από τα χρονικά παράθυρα, όπως η απόσταση μεταξύ των κόμβων ή η χωρητικότητα των οχημάτων. Το 1999 οι Louis et al. βελτιστοποίησαν τον παραπάνω αλγόριθμο ώστε να έχει καλύτερα αποτελέσματα σε προβλήματα, ομαδοποιώντας τις τοποθεσίες των πελατών. Το πρόβλημα ήταν πως οι πρώτοι Ν πελάτες προς εξυπηρέτηση αντιστοιχιζόντουσαν σε Ν τυχαία οχήματα. Με τον τρόπο αυτό υπάρχει μεγάλη πιθανότητα, περισσότεροι από έναν πελάτες ενός cluster να αντιστοιχηθούν σε περισσότερα από ένα οχήματα, ενώ επι της ουσίας απαιτείται ένα μόνο όχημα για την εξυπηρέτηση κάθε πελάτη.

Οι Berger et al. το 1998, επίσης εισήγαγαν έναν υβριδικό γενετικό αλγόριθμο με κατασκευαστικές ευρετικές μεθόδους, αναπαριστώντας τη λύση με ένα σετ από εφικτές διαδρομές. Προκειμένου να δημιουργήσουν την αρχική πελατειακή ομάδα, χρησιμοποίησαν την ευρετική μέθοδο του πλησιέστερου γείτονα που αναλύσαμε παραπάνω. Ο τελεστής του συνδυασμού των διαδρομών, συνδυάζει επαναληπτικά διάφορες διαδρομές R1 της γονικής λύσης με ένα άλλο υποσύνολο πελατών, που σχηματίζεται από τις διαδρομές του κοντινότερου γείτονα της γονικής λύσης. Έπειτα, μέσω μιας διαδικασίας αφαιρούνται ορισμένοι κόμβοι και μέσω μιας τοπικής ευρετικής προσέγγισης (Solomon, 1987) δημιουργείται μία εφικτή διαδρομή, θεωρώντας την διαδρομή R1 ως αρχική λύση του προβλήματος.

**3.5 Ant Colony Optimization**

Ο αλγόριθμος Αποικίας Μυρμηγκιών (Ant Colony Optimization, Dorigo and Caro 1999) αναπαριστά την συμπεριφορά της πραγματικής αποικίας μυρμηγκιών οι οποίες στην αναζήτηση τροφής αναζητούν τις καλύτερες διαδρομές που μπορούν να ακολουθήσουν από και προς την πηγή τροφής. Οι διαδικασία λήψης αποφάσεων των μυρμηγκιών ενσωματώνεται στον αλγόριθμο επίλυσης προβλημάτων δρομολόγησης και η προσέγγιση αυτή έχει δείξει ότι είναι ανταγωνιστική με άλλες τεχνικές επίλυσης προβλημάτων VRP.

Η ικανότητα των εντόμων να αναπτύσουν περίπλοκα προβλήματα επιβίωσης έχει κεντρίσει το ενδιαφέρον των επιστημόνων, οι οποίοι αναπτύσουν αλγόριθμους τεχνητής νοημοσύνης ώστε να επιλύουν αντίστοιχα προβλήματα. Ο ACO αλγόριθμος βασίζεται στη συμπεριφορά πραγματικών μυρμηγκιών και κατέχει ενισχυμένη μνήμη ώστε να θυμάται προηγούμενες ενέργειες και να έχει γνώση για τις αποστάσεις μεταξύ των κόμβων. Στην πραγματικότητα, ένα μυρμήγκι μόνο του δεν έχει τη δυνατότητα να επιχειρήσει αποδοτική αναζήτηση τροφής, αλλά ως σύνολο καταφέρνουν τέτοιες περίπλοκες διαδικασίες με μεγάλη αποδοτικότητα. Σε επίπεδο αλγορίθμου, κάθε όχημα αναπαριστά ένα μυρμήγκι και αρχικά κάθε μυρμήγκι ξεκινά από την αφετηρία με το σύνολο των κόμβων/πελατών του να είναι κενό. Στη συνέχεια επιλέγει τον πελάτη που θα επισκεφτεί από μία λίστα εφικτών λύσεων και η χωρητικότητα που διαθέτει το όχημα ενημερώνεται πριν επιλέξει τον επόμενο πελάτη. Το μυρμήγκι επιστρέφει στην αφετηρία όταν συναντήσει τον περιορισμό της χωρητικότητας ή όταν επισκεφτούν όλοι οι πελάτες. Η συνολική απόσταση υπολογίζεται από την αντικειμενική συνάρτηση με τιμή εκχώρησης την συνολική διαδρομή του μυρμηγκιού. Ο αλγόριθμος ACO κατασκευάζει μία πλήρη διαδρομή για το πρώτο μυρμήγκι και στη συνέχεια το επόμενο ξεκινά την περιήγηση του. Αυτή η διαδικασία ολοκληρώνεται όταν ένα προκαθορισμένο πλήθος μυρμηγκιών m έχει κατασκευάσει από μία εφικτή διαδρομή.

**3.6 Simulated Annealing**

Η μέθοδος της προσομοιωμένης ανόπτησης (simulated annealing) ανακαλύφθηκε από τον MN Rosenbluth και δημοσιεύθηκε πρώτη φορά από τους Metropolis, Rosenbluth, Rosenbluth, Teller και Teller 1953. Αποτελεί μία πιθανολογική μέθοδος που αναπτύχθηκε αργότερα από τους Kirkpatrick, Gelett και Vecchi (1983). Στόχος είναι να βρεθεί το ολικό ελάχιστο μιας συνάρτησης ανάμεσα σε πολλαπλά τοπικά ελάχιστα. Προβλήματα που ο αριθμός των αντικειμένων είναι αρκετά μεγάλος , χρειάζονται λύσεις που να μπορούν να διαχειριστούν τέτοιου είδους προβλήματα. Μία τέτοια λύση αποτελεί η προσομοιωμένη ανόπτηση, ωστόσο συνήθως δεν βρίσκει τη βέλτιστη λύση, αλλά μία κοντά σε αυτή. Οι μέθοδοι τοπικής αναζήτησης πραγματοποιούν μόνο κινήσεις οι οποίες βελτιώνουν το αποτέλεσμα, σε αντίθεση με την SA που πραγματοποιεί κάποιες πιθανότητες. Πιο συγκεκριμένα, οι πιθανότητες αυτές εξαρτώνται από μία παράμετρο που ονομάζεται temperature (T). Σε κάθε επανάληψη του αλγορίθμου, δημιουργείται τυχαία ένα νέο σημείο και η απόσταση του από το αμέσως προηγούμενο καθορίζεται αναλογικά με την παράμετρο T. Βάσει του Τ, σπάνια επιλέγεται ένας γείτονας με αρνητική επίπτωση στο πρόβλημα, ωστόσο στην αρχή της λύσης ενδέχεται να γίνει κάποια “κακή” επιλογή στη φάση της εξερεύνησης των λύσεων. Στη συνέχεια με την παραπάνω διαδικασία προσπαθεί να βελτιώσει την αρχική λύση έως ότου να μη μπορεί να βρει καλύτερη.

**4 Λύτες προβλημάτων**

Το “κυνήγι” για αποτελεσματικές και αξιόπιστες λύσεις που μπορούν να χειριστούν περίπλοκα προβλήματα έχει μετατραπεί σε βασικό τομέα εστίασης τόσο για την ακαδημαϊκή έρευνα όσο και για τις πρακτικές εφαρμογές στη βιομηχανία. Πολλές βιομηχανίες, συμπεριλαμβανομένων των logistics και της διαχείρισης της εφοδιαστικής αλυσίδας, των χρηματοοικονομικών, της μηχανικής και άλλων τομέων, έχουν προβλήματα βελτιστοποίησης. Ως εκ τούτου, είναι ζωτικής σημασίας να παρέχονται αποτελεσματικά εργαλεία βελτιστοποίησης προκειμένου να επιταχυνθεί η λήψη αποφάσεων, να αυξηθεί η κατανομή πόρων και τελικά να βελτιωθεί η απόδοση.

Σε αυτή τη διατριβή, οι Gurobi, LocalSolver, VRPy και Google OR-Tools τέσσερις γνωστοί λύτες βελτιστοποίησης, εξετάζονται διεξοδικά προκειμένου να ρίξουν φως στην υποκείμενη μεθοδολογία, τις δυνατότητες και τις επιδόσεις τους στο χειρισμό ενός ευρέος φάσματος προβλημάτων βελτιστοποίησης. Ο κύριος στόχος είναι να αξιολογηθούν και να συγκριθούν διάφορες λύσεις, προσφέροντας οξυδερκείς πληροφορίες.

**4.1 Google Or-Tools**

Τα προβλήματα δρομολόγησης οχημάτων (VRP) και τα προβλήματα δρομολόγησης με περιορισμένη χωρητικότητα (CVRP) είναι δύσκολα προβλήματα βελτιστοποίησης που προκύπτουν σε διάφορους κλάδους, συμπεριλαμβανομένης της διανομής, της εφοδιαστικής αλυσίδας και των μεταφορών. Προκειμένου να μεγιστοποιηθεί η χρήση των πόρων, να μειωθεί το κόστος μεταφοράς και να βελτιωθεί η συνολική λειτουργική απόδοση, είναι απαραίτητο να επιλυθούν αποτελεσματικά αυτές οι δυσκολίες. Μια ισχυρή εργαλειοθήκη βελτιστοποίησης που ονομάζεται Google OR-Tools και τη βρίσκουμε στον σύνδεσμο https://developers.google.com/optimization, χρησιμοποιήθηκε στην παρούσα εργασία για την επίλυση τέτοιων προβλημάτων και παρέχει ένα ευρύ φάσμα αλγορίθμων και μεθόδων για την επιτυχή επίλυση των VRP και CVRP προβλημάτων. Η διαδικασία που ακολουθεί το OR-Tools είναι η εξής:

* **Μοντελοποίηση προβλήματος**: Το Google OR-Tools προσφέρει μια φιλική προς το χρήστη διεπαφή προγραμματισμού που επιτρέπει σε ακαδημαϊκούς και επαγγελματίες του κλάδου να προσδιορίζουν με ακρίβεια προβλήματα δρομολόγησης σαν τα VRP και CVRP. Ο αριθμός των διαθέσιμων οχημάτων, η χωρητικότητά τους, οι κόμβοι/πελάτες που πρέπει να επισκεφτούν, οι αποστάσεις μεταξύ τους, χρονικά πλαίσια (Time Windows) για παραδόσεις και οποιοιδήποτε άλλοι περιορισμοί ειδικά για την περίπτωση του προβληματος μπορούν όλα να καθοριστούν χρησιμοποιώντας το OR-Tools. Αυτό το προσαρμόσιμο πλαίσιο μοντελοποίησης καθιστά απλή την προσαρμογή σε διάφορα σενάρια VRP.
* **Επιλογή επίλυσης**: Το OR-Tools παρέχει μια ποικιλία λύσεων που έχουν δημιουργηθεί ειδικά για VRP και CVRP. Για τον αποτελεσματικό εντοπισμό βέλτιστων λύσεων, το Routing Solver στο OR-Tools χρησιμοποιεί μια ποικιλία μεθόδων αναζήτησης, ευρετικών και μεταευρετικών μεθόδων, όπως ενδεικτικά αναλύσαμε παραπάνω. Για την αντιμετώπιση της πολυπλοκότητας των περιπτώσεων VRP και CVRP, ενσωματώνει μεθόδους όπως εποικοδομητικά ευρετικά (constructive heuristics), τοπική αναζήτηση και αλγόριθμους βελτιστοποίησης.
* **Αρχική δημιουργία λύσεων**: Το OR-Tools χρησιμοποιεί χρήσιμα ευρετικά για να ξεκινήσει τη διαδικασία λύσης. Για τη δημιουργία ενός αρχικού σχεδίου δρομολόγησης, αυτά τα ευρετικά μπορεί να χρησιμοποιούν τεχνικές όπως ευρετικές εισαγωγές, τον αλγόριθμο του πλησιέστερου γείτονα ή αλγόριθμους εξοικονόμησης. Ο στόχος είναι να παρέχουμε μια εφαρμόσιμη λύση που να συμμορφώνεται με τις θεμελιώδεις απαιτήσεις, διασφαλίζοντας ότι όλοι οι πελάτες ή κόμβοι επισκέπτονται τουλάχιστον μια φορά και λαμβάνοντας υπόψη τους περιορισμούς χωρητικότητας του οχήματος.
* Αφού αναπτυχθεί η αρχική λύση, το OR-Tools χρησιμοποιεί **τεχνικές βελτιστοποίησης** για να βελτιώσει την αρχική λύση. Μεταξύ των μεθόδων που χρησιμοποιούνται είναι οι γενετικοί αλγόριθμοι, η αναζήτηση tabu, η προσομοιωμένη ανόπτηση και οι αλγόριθμοι τοπικής αναζήτησης. Ο στόχος αυτών των τεχνικών βελτιστοποίησης είναι η μείωση του συνολικού κόστους ή της αντικειμενικής συνάρτησης εξετάζοντας επαναληπτικά τον χώρο αναζήτησης και κάνοντας μικρές προσαρμογές στις διαδρομές.
* **Περιορισμοί χειρισμού**: Το OR-Tools παρέχει πλήρη βοήθεια για τη διαχείριση μιας ποικιλίας περιορισμών που συναντώνται συχνά στο VRP και το CVRP. Αυτά περιλαμβάνουν περιορισμούς στον αριθμό των αποθηκών, τον αριθμό των παραθύρων παραλαβής και παράδοσης και τη χωρητικότητα των οχημάτων. Κατά τη διαδικασία βελτιστοποίησης, ο λύτης διασφαλίζει ότι όλες οι λύσεις που δημιουργούνται τηρούν αυτούς τους περιορισμούς, παράγοντας νόμιμες και πρακτικές διαδρομές.
* **Ανάλυση λύσεων**: Το OR-Tools προσφέρει εργαλεία για την εξέταση και τη συλλογή σχετικών δεδομένων μετά τον εντοπισμό μιας εφαρμόσιμης ή βελτιστοποιημένης λύσης. Οι ερευνητές μπορούν να ανακτήσουν πληροφορίες για πράγματα όπως ισορροπίες φορτίου, χρόνοι ταξιδιού, αποστάσεις και άλλες κρίσιμες παραμέτρους. Αυτό καθιστά δυνατή την ενδελεχή αξιολόγηση της απάντησης και επιταχύνει τη διαδικασία λήψης αποφάσεων.
* **Συντονισμός απόδοσης**: Η ευελιξία του OR-Tools επιτρέπει τον ακριβή έλεγχο της συμπεριφοράς του λύτη. Ανάλογα με τις ιδιαίτερες ανάγκες τους και τους διαθέσιμους πόρους υπολογιστή, οι ερευνητές μπορεί να καθορίσουν παραμέτρους αναζήτησης, χρονικούς περιορισμούς ή στόχους ποιότητας λύσης. Αυτό δίνει τη δυνατότητα στους ερευνητές να βελτιστοποιούν την απόδοση του για διάφορες καταστάσεις ζητημάτων, ενώ παράλληλα εξισορροπούν την ακρίβεια της λύσης και την υπολογιστική οικονομία.

Τα προβλήματα που θα δούμε παρακάτω λύθηκαν με το Python API της OR-Tools, δίνοντας τα δεδομένα με τον συγκεκριμένο τρόπο:

*def create\_data\_model ():*

*data = {}*

*data["distance\_matrix"] = [*

*[0, 2451, 713, 1018, 1631, 1374, 2408, 213, 2571, 875, 1420, 2145, 1972],*

*[2451, 0, 1745, 1524, 831, 1240, 959, 2596, 403, 1589, 1374, 357, 579],*

*[713, 1745, 0, 355, 920, 803, 1737, 851, 1858, 262, 940, 1453, 1260],*

*[1018, 1524, 355, 0, 700, 862, 1395, 1123, 1584, 466, 1056, 1280, 987],*

*[1631, 831, 920, 700, 0, 663, 1021, 1769, 949, 796, 879, 586, 371],*

*[1374, 1240, 803, 862, 663, 0, 1681, 1551, 1765, 547, 225, 887, 999],*

*[2408, 959, 1737, 1395, 1021, 1681, 0, 2493, 678, 1724, 1891, 1114, 701],*

*[213, 2596, 851, 1123, 1769, 1551, 2493, 0, 2699, 1038, 1605, 2300, 2099],*

*[2571, 403, 1858, 1584, 949, 1765, 678, 2699, 0, 1744, 1645, 653, 600],*

*[875, 1589, 262, 466, 796, 547, 1724, 1038, 1744, 0, 679, 1272, 1162],*

*[1420, 1374, 940, 1056, 879, 225, 1891, 1605, 1645, 679, 0, 1017, 1200],*

*[2145, 357, 1453, 1280, 586, 887, 1114, 2300, 653, 1272, 1017, 0, 504],*

*[1972, 579, 1260, 987, 371, 999, 701, 2099, 600, 1162, 1200, 504, 0],*

*]*

*data["num\_vehicles"] = 1*

*data["depot"] = 0*

*return data*

και στη συνέχεια για να τυπώσουμε τη λύση αρκεί η κλήση της συγκεκριμένης συνάρτησης με τις αντίστοιχες παραμέτρους:

*def print solution(manager, routing, solution):*

*"""Prints solution on console."""*

*print(f"Objective: {solution.ObjectiveValue()} miles")*

*index = routing.Start(0)*

*plan\_output = "Route for vehicle 0:\n"*

*route\_distance = 0*

*while not routing.IsEnd(index):*

*plan\_output += f" {manager.IndexToNode(index)} ->"*

*previous\_index = index*

*index = solution.Value(routing.NextVar(index))*

*route\_distance += routing.GetArcCostForVehicle(previous\_index, index, 0)*

*plan\_output += f" {manager.IndexToNode(index)}\n"*

*print(plan\_output)*

*plan\_output += f"Route distance: {route\_distance}miles\n"*

Συμπερασματικά, το Google OR-Tools παρέχει ένα ισχυρό πλαίσιο εύρεσης λύσεων προβλημάτων VRP και CVRP, καθώς με τις ευέλικτες μεθόδους μπορεί να εντοπίσει σε λιγοστό χρόνο λύσεις σε τέτοια προβλήματα. Ωστόσο, είναι χρήσιμο να αναφέρουμε, πως λόγω του ότι χρησιμοποιεί ευρετικές και μεταευρετικές μεθόδους, δεν εγγυάται πως θα βρεί την βέλτιστη λύση στα παραπάνω προβλήματα, αλλά μία εξίσου καλή λύση σε μικρό χρόνο.

**4.2 Gurobi Optimization**

Ένας επίσης αποτελεσματικός λύτης που χρησιμοποιήθηκε στην παρούσα εργασία είναι το Gurobi Optimization το οποίο διαφέρει απο το OR-Tools ως προς την προσέγγιση της λύσης, καθώς δε χρησιμοποιεί ευρετικές μεθόδους, αλλά μαθηματικά μοντέλα.

* **Μοντελοποίηση προβλήματος**: Το Gurobi, όπως και το OR-Tools, προσφέρει ένα φιλικό προς τον χρήστη περιβάλλον μοντελοποίησης που δίνει τη δυνατότητα σε ερευνητές και επαγγελματίες του κλάδου να χαρακτηρίζουν με ακρίβεια τις δυσκολίες VRP και CVRP. Ο αριθμός των φορτηγών, η χωρητικότητά τους, ο κατάλογος των τοποθεσιών που πρέπει να επισκεφθείτε, οι πίνακες χρόνου απόστασης ή ταξιδιού, τα χρονικά πλαίσια για τις παραδόσεις και οποιοιδήποτε άλλοι περιορισμοί, μοντελοποιούνται όπως και στο OR-Tools από τον χρήστη μέσω μιας γλώσσας προγραμματισμού και στη δικια μας περίπτωση χρησιμοποιήθηκε η γλώσσα Python.
* **Επιλογή επίλυσης**:Ο Gurobi χρησιμοποιεί εξελιγμένα μαθηματικά μοντέλα προγραμματισμού για να λύσει το VRP και το CVRP. Τα ζητήματα VRP και CVRP μπορούν να διαμορφωθούν ως μοντέλα γραμμικού προγραμματισμού μικτού ακέραιου (MILP) ή τετραγωνικού προγραμματισμού μικτού ακέραιου (MIQP), αντίστοιχα. Αυτά τα μοντέλα προσφέρουν μια μαθηματική αναπαράσταση του προβλήματος που το Gurobi μπορεί να λύσει αποτελεσματικά ενώ κωδικοποιεί τους στόχους, τους περιορισμούς και τους παράγοντες απόφασης.
* Ο Gurobi, όπως αναφέραμε χρησιμοποιεί μαθηματικά μοντέλα για να βελτιστοποιήσει ένα δεδομένο πρόβλημα, μεταφράζοντας το μαθηματικό μοντέλο σε προγραμματιστική μορφή. Με τον ίδιο τρόπο, μοντελοποιούνται και όλοι οι περιορισμοί που είναι απαραίτητοι ώστε να βρεθεί η καλύτερη δυνατή λύση, όπως ο περιορισμός της χωρητικότητας των οχημάτων, το χρονικό περιορισμό για την επίλυση του προβλήματος. Στη συγκεκριμένη εργασία χρησιμοποιήθηκε το μαθηματικό μοντέλο Miller-Tucker-Zemlin (MTZ), το οποίο θα αναλύσουμε διεξοδικότερα.
* Η συνάρτηση στόχου επιδιώκει να μειώσει το συνολικό κόστος, την απόσταση ή οποιαδήποτε άλλη αντικειμενική παράμετρο που σχετίζεται με το CVRP. Μπορεί να γραφτεί ως το σύνολο του κόστους ή των αποστάσεων επί τις αντίστοιχες μεταβλητές τόξου: Ο στόχος είναι να ελαχιστοποιηθεί το (i, j) c[i, j] \* x[i, j], όπου c[i, j] είναι το κόστος ή η απόσταση μεταξύ των κόμβων i και j. Στη δική μας περίπτωση στόχος είναι η ελαχιστοποίηση της απόστασης.
* **Περιορισμοί χωρητικότητας**: Η συνάρτηση MTZ προσθέτει περιορισμούς που σχετίζονται με τη χωρητικότητα προκειμένου να εφαρμοστούν οι περιορισμοί χωρητικότητας. Το συνολικό φορτίο μέχρι κάθε κόμβο i (εκτός από την αποθήκη) δεν πρέπει να είναι μεγαλύτερο από την ικανότητα μεταφοράς του οχήματος (Q). Αυτοί οι περιορισμοί μπορούν να γραφτούν ως εξής: u[i] Q, όπου i=1 υποδηλώνει την αποθήκη (depot).
* Η διατύπωση MTZ περιέχει περιορισμούς διατήρησης ροής για να εγγυηθεί ότι κάθε κόμβος (εκτός από την αποθήκη) επισκέπτεται ακριβώς μία φορά. Αυτοί οι περιορισμοί μπορούν να γραφτούν ως εξής: (j i) x[i, j] = 1, για όλα τα j1 (εισερχόμενη ροή προς την αποθήκη), (i j) x[i, j] = 1, για όλα τα j1 (εξερχόμενα ροή από την αποθήκη). Για κάθε j 1 (ισοζύγιο ροής σε ενδιάμεσους κόμβους), (i) x[i, j] = (i) x[j, i].
* Κατάργηση των περιορισμών Subtour: Η σύνθεση MTZ προσθέτει επιπλέον περιορισμούς χρησιμοποιώντας μια βοηθητική μεταβλητή, y[i], προκειμένου να αφαιρεθούν οι υποπεριηγήσεις (διαδρομές που δεν ξεκινούν και δεν τελειώνουν στο αμαξοστάσιο). Αυτοί οι περιορισμοί διασφαλίζουν ότι κάθε διαδρομή διατηρεί την ισορροπία της ροής και αρχίζει και τελειώνει στο αμαξοστάσιο.

Από προγραμματιστική άποψη, η εφαρμογή του MTZ μοντέλου στον κώδικα του Gurobi έγινε ως εξής:

*m = gp.Model()*

*x = m.addVars(G.edges,vtype=GRB.BINARY)*

*m.setObjective( gp.quicksum( G.edges[i,j]['length']\* x[i,j] for i,j in G.edges ), GRB.MINIMIZE )*

*# Enter each demand point once*

*m.addConstrs( gp.quicksum( x[i,j] for i in G.predecessors(j) ) == 1 for j in dem\_points )*

*# Leave each demand point once*

*m.addConstrs( gp.quicksum( x[i,j] for j in G.successors(i) ) == 1 for i in dem\_points )*

*# Leave the depot k times*

*m.addConstr( gp.quicksum( x[depot,j] for j in G.successors(depot) ) == k )*

*# Add the MTZ variables and constraints, and solve*

*u = m.addVars( G.nodes )*

*u[depot].LB = 0*

*u[depot].UB = 0*

*for i in dem\_points:*

*u[i].LB = q[i]*

*u[i].UB = Q*

*c = m.addConstrs( u[i] - u[j] + Q \* x[i,j] <= Q - q[j] for i,j in G.edges if j != depot )*

*m.Params.TimeLimit = 120*

Στη δική μας περίπτωση, ως κόστος εννοούμε την απόσταση μεταξύ των κόμβων προς επίσκεψη. Στόχος είναι η εύρεση της βέλτιστης διαδρομής των οχημάτων, ώστε να επισκεφτούν όλοι οι κόμβοι, διανύοντας την συντομότερη δυνατή διαδρομή.

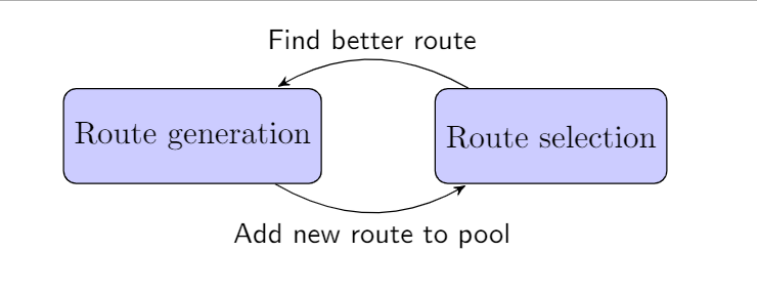
**4.3 VRPy**

Το CVRP απεικονίζεται από το VRPy ως γράφημα G, όπου οι κόμβοι είναι οι πελάτες προς επίσκεψη και οι ακμές είναι οι συνδέσεις μεταξύ τους. Κάθε πελάτης έχει μια ζήτηση (την ποσότητα των αντικειμένων που θέλει να παραδοθούν) και συνδέεται με μια, ή περισσότερες, τοποθεσία. Στόχος, όπως και με τους υπόλοιπους λύτες, είναι η ελαχιστοποίηση της απόστασης που θα διανύσει ο στόλος των οχημάτων. Η λογική επίλυσης προβλημάτων μέσω του VRPy είναι παρόμοια με αυτή του OR-Tools:

* **Αρχικοποίηση:** Το VRPy αρχικά ορίζει τα δεδομένα του προβλήματος, όπως ο αριθμός τον οχημάτων που θα εξυπηρετήσουν τους πελάτες, οι συντεταγμένες των κόμβων που απεικονίζουν τους πελάτες, η ζήτηση κάθε κόμβου, καθώς και η χωρητικότητα των οχημάτων.
* **Δημιουργία λύσης:** Για την εύρεση λύσης, χρησιμοποιούνται κυρίως ευρετικές μέθοδοι όπως ο αλγόριθμος αποταμίευσης Clarke and Wright. Στα οχήματα, αντιστοιχούνται συγκεκριμένοι πελάτες που θα εξυπηρετηθούν απο τα συγκεκριμένα οχήματα και η δημιουργία δοκιμαστικών διαδρομών αποτελεί την αρχική λύση.

* **Τεχνικές βελτίωσης:** Για να βελτιώσει την αρχική λύση, το VRPy χρησιμοποιεί πολλαπλές τεχνικές βελτίωσης. Η αλληλουχία των πελατών που επισκέπτονται σε κάθε διαδρομή βελτιστοποιείται από αυτούς τους αλγόριθμους, οι οποίοι περιλαμβάνουν και τεχνικές τοπικής αναζήτησης όπως 2-opt και 3-opt. Η συνολική απόσταση θα πρέπει να περιορίζεται στο ελάχιστο και θα πρέπει να τηρούνται οι χωρητικότητες των οχημάτων, καθώς και όλοι οι πιθανοί περιορισμοί όπως χρονικά παράθυρα.
* **Metaheuristics:** Για να εξερευνήσει το χώρο των λύσεων και να εντοπίσει καλύτερες λύσεις, αν όχι βέλτιστες, το VRPy χρησιμοποιεί επίσης μεταευρετικές μεθόδους. Η επαναληπτική αναζήτηση καλύτερων λύσεων τροποποιώντας τις τρέχουσες διαδρομές, εναλλαγή καταναλωτών μεταξύ διαδρομών ή αναπτύσσοντας νέες διαδρομές γίνεται αποτελεσματική με τη χρήση μεταευρετικών όπως γενετικοί αλγόριθμοι, προσομοιωμένης ανόπτησης ή αναζήτηση tabu, τα οποία έχουμε αναφέρει σε προηγούμενη ενότητα.

Τα προβλήματα δρομολόγησης επιλύονται από το VRPy χρησιμοποιώντας μια μέθοδο δημιουργίας στήλης. Η δημιουργία στηλών περιγράφει την επαναληπτική διαδικασία δημιουργίας διαδρομών (ή στηλών) με πρόβλημα τιμολόγησης και τροφοδοσίας αυτών των διαδρομών σε ένα κύριο πρόβλημα που επιλέγει τις καλύτερες διαδρομές από ένα σύνολο, έτσι ώστε κάθε κόμβος να εξυπηρετείται με ακριβώς μία φορά. Στη συνέχεια, οι λύσεις στο κύριο πρόβλημα χρησιμοποιούνται για την αναζήτηση πρόσθετων μελλοντικών διαδρομών που θα μπορούσαν να μειώσουν το κόστος της λύσης, μεταξύ άλλων.



Πηγή: <https://joss.theoj.org/papers/10.21105/joss.02408>

Ωστόσο, όπως αναφέρεται αναλυτικά στο documentation, το VRPy δεν βρίσκει υποχρεωτικά βέλτιστη λύση στο εκάστοτε πρόβλημα. Στην πραγματικότητα όταν το πρόβλημα τιμολόγησης δεν μπορεί να επιλυθεί, τότε το αρχικό πρόβλημα επιλύεται ως ένα πρόβλημα μικτού ακέραιου προγραμματισμού

Όπως αναφέρεται και στο documentation της βιβλιοθήκης VRPy στον σύνδεσμο <https://vrpy.readthedocs.io/en/latest/>, οι κόμβοι του κάθε προβλήματος αποθηκεύονται σε ένα γράφημα G της βιβλιοθήκης networkx (<https://networkx.org/>). Στη συγκεκριμένη υλοποίηση, χρησιμοποιήθηκε ο parser της βιβλιοθήκης tsplib95 (<https://pypi.org/project/tsplib95/>), ώστε τα δεδομένα του προβλήματος να αποθηκεύονται σε μία μεταβλητή problem.

Η κλήση της συγκεκριμένης βιβλιοθήκης γίνεται με μοναδικό όρισμα το path του αρχείου με κατάληξη .vrp, το οποίο περιέχει τα δεδομένα του προβλήματος

*problem = tsplib95.load("./B-n63-k10.vrp")*

*G = DiGraph()*

*# Initialize the distance matrix*

*n = len(problem.node\_coords)*

*distances = [[0] \* n for i in range(n)]*

*# Calculate the Euclidean distance between each pair of nodes*

*nx.set\_node\_attributes(G, {0: {"source": True}}) #Depot 0 is Source and Sink as well.*

*nx.set\_node\_attributes(G, {0: {"sink": True}})*

*for i in problem.node\_coords:*

*for j in problem.node\_coords:*

*xi, yi = problem.node\_coords[i]*

*xj, yj = problem.node\_coords[j]*

*distances[i - 1][j - 1] =int(math.sqrt((xi - xj) \*\* 2 + (yi - yj) \*\* 2))*

*for i in range(1,problem.dimension):*

*G.add\_edge("Source", i, cost= distances[0][i])*

*G.add\_edge(i, "Sink", cost=distances[i][1])*

*for i in range(1,len(problem.node\_coords)):*

*for j in range(1, len(problem.node\_coords)):*

*G.add\_edge(i, j, cost = distances[i][j])*

*for i in range(0,problem.dimension-1):*

*G.nodes[i+1]["demand"] = problem.demands[i+1]*

*prob = VehicleRoutingProblem(G, load\_capacity=problem.capacity)*

*prob.solve()*

Στον παραπάνω κώδικα, υπολογίζεται η ευκλείδια απόσταση μεταξή των κόμβων του προβλήματος και αρχικοποιείται το γράφημα G με τους κόμβους, το depot και το sink.

Ένα πρόβλημα που αντιμετωπίσαμε, ήταν πως η συγκεκριμένη βιβλιοθήκη θεωρεί την αφετηρία και την κατάληξη ως διαφορετικά σημεία, ενώ στη δική μας περίπτωση αυτά τα δύο ταυτίζονται. Το πρόβλημα επιλύθηκε με ορισμένες προσαρμογές στον πηγαίο κώδικα και έτσι επιστρέφονται τα ζητούμενα.

**4.4 Local Solver**

Μια ποικιλία θεμάτων συνδυαστικής βελτιστοποίησης, συμπεριλαμβανομένου του Προβλήματος Δρομολόγησης Χωρητικών Οχημάτων (CVRP), μπορούν να επιλυθούν χρησιμοποιώντας το εμπορικό πρόγραμμα βελτιστοποίησης LocalSolver (<https://www.localsolver.com/>) . Για την εγκατάσταση του LocalSolver απαιτείται ειδική άδεια, για την οποία στέλνουμε αίτημα στην επίσημη σελίδα. Ο κώδικας της υλοποίησης της λύσης είναι δωρεάν για όλους, ωστόσο το πρόγραμμα δεν είναι εκτελέσιμο καθώς δε θα είναι διαθέσιμες οι βιβλιοθήκες χωρίς την ειδική άδεια. Για τον εντοπισμό βέλτιστων λύσεων, συνδυάζει τοπική αναζήτηση και μεθόδους μαθηματικού προγραμματισμού. Το LocalSolver παρέχει μια ποικιλία από APIs ώστε τα διάφορα προβλήματα να μπορούν να λυθούν σε διάφορες γλώσσες προγραμματισμού, όπως Python, Java, C++ κλπ. Στην παρούσα εργασία χρησιμοποιείται η γλώσσα προγραμματισμού Python.

Το Local Solver επιλύει προβλήματα CVRP ως εξής:

* **Μοντελοποίηση:** Μιας και το LocalSolver παρέχει έτοιμα APIs προς εκτέλεση, από τη μεριά του χρήστη το μόνο που έχουμε να κάνουμε είναι να δώσουμε στο εργαλείο ένα αρχείο με όλα τα δεδομένα του προβλήματος, δηλαδή τον αριθμό των διαθέσιμων οχημάτων, τη χωρητικότητα τους, την τοποθεσία των κόμβων προς εξυπηρέτηση, τη ζήτηση τους καθώς και άλλους παραπάνω περιορισμούς που ενδέχεται να υπάρχουν.
* **Αρχικοποίηση**: Δημιουργώντας μια πρώτη εφαρμόσιμη λύση, το LocalSolver προετοιμάζει τη λύση του προβλήματος. Λαμβάνοντας υπόψη τους περιορισμούς χωρητικότητας και τους πρόσθετους περιορισμούς που αφορούν το συγκεκριμένο πρόβλημα, διανέμει τους πελάτες στα οχήματα. Ευρετικές μέθοδοι, καθώς και άλλες τεχνικές,

χρησιμοποιούνται για την εύρεση μιας εφικτής λύσης.

* **Τοπική αναζήτηση**: Το LocalSolver χρησιμοποιεί μια μέθοδο τοπικής αναζήτησης για να αναζητήσει επαναληπτικά βελτιώσεις στην περιοχή γύρω από την υπάρχουσα λύση. Προσαρμόζει την παρούσα λύση τοπικά, όπως εναλλαγή ή αναδιάταξη πελατών μεταξύ διαδρομών, και στη συνέχεια αξιολογεί τα αποτελέσματα.
* **Μαθηματικός προγραμματισμός**: Για τη βελτιστοποίηση της υπάρχουσαν λύσης, το LocalSolver χρησιμοποιεί επιπλέον μεθόδους μαθηματικού προγραμματισμού. Μοντελοποιεί το πρόβλημα ως πρόβλημα μαθηματικού προγραμματισμού και στη συνέχεια το επιλύει χρησιμοποιώντας τεχνικές βελτιστοποίησης όπως προγραμματισμός μικτού ακέραιου ή γραμμικού προγραμματισμού.
* **Επαναληπτική Βελτιστοποίηση:** Για να βελτιώσει σταδιακά την λύση, το LocalSolver χρησιμοποιεί μια επαναληπτική διαδικασία τοπικής αναζήτησης και μαθηματικού προγραμματισμού. Βελτιώνοντας επαναληπτικά τη λύση, κάνει μικρές προσαρμογές και αξιολογεί πώς αυτές επηρεάζουν το αποτέλεσμα. Αυτή η διαδικασία συνεχίζεται έως ότου ανακαλυφθεί μια αποδεκτή ή βέλτιστη λύση ή μέχρι να ικανοποιηθεί ένας προκαθορισμένος περιορισμός.

Η κλίση των CVRP προβλημάτων με το API του LocalSolver γίνεται με την κλήση της *main* συνάρτησης με ορίσματα το .vrp αρχείο με τα δεδομένα, το χρονικό περιθώριο στο οποίο θέλουμε να βρεθεί λύση, το αρχείο στο οποίο θα αποθηκευτεί η λύση, καθώς και ο αριθμός των οχημάτων τπυ προβλήματος αφού δεν παρουσιάζεται μέσα στο αρχείο των δεδομένων. Συγκεκριμένα, η κλίση πρέπει να είναι ως εξής:

*main(instance\_file, str\_time\_limit, output\_file, str\_nb\_trucks)*

Για την αποτελεσματική αντιμετώπιση προβλημάτων CVRP, το LocalSolver συνδυάζει αλγορίθμους τοπικής αναζήτησης και μαθηματικές μεθόδους προγραμματισμού. Ο στόχος του LocalSolver είναι να αναπτύξει λύσεις υψηλής ποιότητας για CVRP προβλήματα που πληρούν τους περιορισμούς χωρητικότητας και βελτιστοποιούν το αποτέλεσμα . Αυτό το επιτυγχάνει εξερευνώντας επανειλημμένα το χώρο της λύσης, βελτιώνοντας τη λύση μέσω τοπικών προσαρμογών και χρησιμοποιώντας τις παραπάνω μεθόδους.

**5** **Συλλογές Μετροπροβλημάτων**

Κατα την ανάπτυξη της συγκεκριμένης εργασίας, αναζητήσαμε δεδομένα προβλημάτων στο διαδίκτυο και συγκεκριμένα από συλλογές μετροπροβλημάτων. Οι δύο σημαντικότερες από αυτές είναι η [TSPLIB95](http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) καθώς και η [CVRPLIB](http://vrp.galgos.inf.puc-rio.br/index.php/en/), από την οποία τελικά αντλήσαμε όλα τα προβλήματα. Και στις δύο συλλογές διατίθενται δεδομένα προβλημάτων δρομολόγησης καθώς και για προβλήματα περιοδεύοντος πωλητή (TSP).

**5.1 TSPLIB95**

Η TSPLIB95 δημιουργήθηκε το 1991 από τον Gerhard Reinelt, ερευνητή του Πανεπιστημίου του Άουγκσμπουργκ (Reinelt, G. 1991) με σκοπό να παρέχει δεδομένα προβλημάτων TSP ώστε να μπορούν και άλλοι ερευνητές να συγκρίνουν αλγορίθμους που επιλύουν τέτοια προβλήματα. Πέρα από TSP προβλήματα ωστόσο, η βιβλιοθήκη παρέχει δεδομένα και για άλλες κατηγορίες προβλημάτων όπως CVRP(Capacitated Vehicle Routing Problem), HCP (Hamiltonian cycle problem) , ATSP (Asymmetric traveling salesman problem) και άλλα.

Πέρα από τα δεδομένα προβλημάτων όμως, η ίδια η βιβλιοθήκη παρέχει και έναν parser που διευκολύνει την ανάγνωση τέτοιων προβλημάτων. Ο συγκεκριμένος parser μπορεί να γίνει εγκατάσταση σε κάποιο σύστημα με εγκατεστημένη την Python (όχι απαραίτητα) χρησιμοποιώντας την εντολή “pip install tsplib95” στο terminal του υπολογιστή. Με το συγκεκριμένο εργαλείο μπορούμε να έχουμε πρόσβαση σε δεδομένα προβλημάτων της μορφής “.vrp” και το μόνο που χρειάζεται είναι να την καλέσουμε με όρισμα το όνομα (ή το path) του αρχείου με τον τρόπο data = tsplib95.load("B-n63-k10.vrp"), όπου “Β-n63-k10.vrp” το όνομα του αρχείου. Στο section που αφορά το Gurobi θα δούμε αναλυτικότερα τον τρόπο χρήσης του parser.

Οι παροχές του parser είναι οι εξής:

Ανάγνωση και εγγραφή: Η tsplib95 επιτρέπει την ανάγνωση και την εγγραφή δεδομένων σε μορφή TSPLIB. Η μορφή TSPLIB είναι ένα ευρέως χρησιμοποιούμενο πρότυπο για την αναπαράσταση περιπτώσεων TSP, καθιστώντας εύκολη την εργασία με σύνολα δεδομένων αναφοράς. Ωστόσο, σε πολλές περιπτώσεις όπως και στη δική μας, χρησιμοποιείται σε διάφορα προβλήματα δρομολόγησης, όπως το CVRP.

Δομές δεδομένων: Η βιβλιοθήκη παρέχει δομές δεδομένων για την αναπαράσταση προβλημάτων δρομολόγησης, συμπεριλαμβανομένων των κόμβων (πόλεις), των αποστάσεων μεταξύ των κόμβων και της συνολικής παρουσίας προβλήματος. Μπορείτε να δημιουργήσετε, να χειριστείτε και να αναλύσετε αυτές τις δομές δεδομένων στην Python.

Μετρικές αποστάσεων: Η tsplib95 περιλαμβάνει διάφορες συναρτήσεις απόστασης για τον υπολογισμό της απόστασης μεταξύ δύο κόμβων. Υποστηρίζονται κοινές μετρήσεις απόστασης, όπως η Ευκλείδεια απόσταση και η EUC\_2D.

Μετατροπή σε networkx γράφημα: Όπως είδαμε νωρίτερα, τα δεδομένα των κόμβων με την υλοποίηση της VRPy αναπαριστώνται σε ένα γράφημα networkx. Τη μετατροπή αυτή τη διευκολύνει η tsplib95 χωρίς να χρειάζεται να γράψουμε περίπλοκα κομμάτια κώδικα για την υλοποίηση της.

Ένα παράδειγμα για να γίνει κατανοητή η χρήση της tsplib95, παρουσιάζεται παρακάτω:

import tsplib95

# Load a TSP instance from a file

problem = tsplib95.load('example.tsp')

# Access information about the problem

print(problem.name) # Name of the TSP instance

print(problem.comment) # Comments about the instance

print(problem.dimension) # Number of cities

# Access the coordinates of the cities

for node in problem.node\_coords:

print(f"City {node} coordinates: {problem.node\_coords[node]}")

# Access distance between two cities

distance = problem.get\_weight(1, 2)

print(f"Distance between city 1 and city 2: {distance}")

**5.2 CVRPLIB**

Στην παρούσα εργασία, όλα τα δεδομένα αντλήθηκαν από τη βιβλιοθήκη CVRPLIB, η οποία δημιουργήθηκε το 2014 από τον Ivan Lima και τους (Uchoa et al. 2017) και συντηρείται από τον Daniel Oliveira (2016-2018) και τον Eduardo Queiroga (2019-σήμερα). Το Πρόβλημα Δρομολόγησης Χωρητικών Οχημάτων (CVRP) μελετάται και αναπτύσσονται λύσεις, ενώ η CVRPLIB αναπτύσσεται ως κύριος και ουσιαστικός πόρος. Αυτή η ενότητα διερευνά την αξία και τα χαρακτηριστικά της βιβλιοθήκης CVRPLIB ενώ παρουσιάζει τη δυνατότητα εφαρμογής του στο πεδίο CVRP.

Μια εμπεριστατωμένη βιβλιοθήκη σημείων αναφοράς, η CVRPLIB επιλέχθηκε ειδικά για να λύσει την πολυπλοκότητα και τις λεπτότητες του Προβλήματος Δρομολόγησης Οχημάτων με περιορισμένη χωρητικότητα. Η κύρια αξία της CVRPLIB προέρχεται από τη λειτουργία της ως τυποποιημένης βιβλιοθήκης που επιτρέπει διεξοδικές αξιολογήσεις των προβλημάτων CVRP, ενισχύει μια πλατφόρμα συγκριτικής ανάλυσης και προωθεί την ανάπτυξη έρευνας δρομολόγησης οχημάτων.

Η ευρεία και ποικίλη σειρά προβλημάτων CVRP που παρέχει η CVRPLIB είναι ένα από τα αξιοσημείωτα χαρακτηριστικά της. Το CVRPLIB προσφέρει ένα ευρύ φάσμα πολυπλοκότητας προβλημάτων, από καταστάσεις μικρής κλίμακας έως παραδείγματα προβλημάτων μεγάλης κλίμακας και υπολογιστικά δύσκολα. Τα παραδείγματα εξασφαλίζουν μια λεπτομερή απεικόνιση των δυσκολιών δρομολόγησης του πραγματικού κόσμου, συμπεριλαμβάνοντας μια σειρά από αριθμούς πελατών, χωρητικότητες οχημάτων και γεωγραφικές διαμορφώσεις. Πέρα από την ποικιλία των περιπτώσεων του, η CVRPLIB περιλαμβάνει μια ποικιλία παραλλαγών ζητημάτων που διευρύνουν το πεδίο εφαρμογής του CVRP ώστε να περιλαμβάνει περιορισμούς και σενάρια από τον πραγματικό κόσμο. Αυτές οι παραλλαγές περιλαμβάνουν καταστάσεις με χρονικά παράθυρα, πολλές αποθήκες και διάφορους στόλους οχημάτων. Οι ερευνητές μπορούν να εξετάσουν και να αξιολογήσουν την προσαρμοστικότητα και την αποτελεσματικότητα των αλγορίθμων κάτω από διάφορους περιορισμούς, συμπεριλαμβάνοντας αυτά τα ρεαλιστικά χαρακτηριστικά στο CVRPLIB.

**5.2 Μορφή Αρχείων**

Η κατανόηση της μορφής αρχείου.vrp που προσφέρεται από το CVRPLIB είναι ζωτικής σημασίας για την ανάλυση των προβλημάτων δρομολόγησης CVRP. Αυτά τα αρχεία περιγράφουν τις εμφανίσεις CVRP με τον πιο αποτελεσματικό τρόπο και περιέχουν λεπτομέρειες σχετικά με τον τρόπο δημιουργίας του προβλήματος. Περιγράφουμε την οργάνωση των αρχείων.vrp σε αυτό το μέρος και εξηγούμε τον σκοπό κάθε ενότητας.

Η ενότητα κεφαλίδας, η οποία αποτελεί το πρώτο μέρος του αρχείου.vrp, παρέχει κρίσιμες πληροφορίες σχετικά με την παρουσία του προβλήματος. Περιλαμβάνει τα ακόλουθα σημαντικά πεδία:

**NAME**: Το όνομα της παρουσίας CVRP, που χρησιμοποιείται ως διακριτική αναγνώριση.

**TYPE**: Προσδιορίζει το είδος του προβλήματος, σε αυτό το παράδειγμα, CVRP.

**COMMENT**: Ένα πεδίο που είναι προαιρετικό για τυχόν περαιτέρω παρατηρήσεις ή επεξηγήσεις.

**DIMENSION**: Αυτός ο αριθμός αντιπροσωπεύει όλους τους κόμβους του προβλήματος, συμπεριλαμβανομένων των κόμβων αποθήκης και πελάτη.

**EDGE\_WEIGHT\_TYPE**: Καθορίζει τη μονάδα μέτρησης για το κόστος ή τις αποστάσεις μεταξύ των κόμβων (για παράδειγμα, EUC\_2D για Ευκλείδεια απόσταση 2D).

Στη δική μας περίπτωση όλα τα προβλήματα ήταν τύπου EUC\_2D.

Η χωρητικότητα κάθε οχήματος που χρησιμοποιείται στη διαδικασία δρομολόγησης υποδεικνύεται με τη λέξη **CAPACITY**.

**Node Section**

Το τμήμα κόμβων του αρχείου.vrp ακολουθεί την ενότητα κεφαλίδας και περιέχει πληροφορίες για κάθε κόμβο, όπως την αποθήκη και τους πελάτες.

Ο όρος "Αριθμός Κόμβου" αναφέρεται στη διακριτική ταυτότητα του κόμβου. Συνήθως, ο Κόμβος Αριθμός 1 αντιπροσωπεύει την αποθήκη και ακολουθούν οι κόμβοι πελατών με αύξουσα σειρά. Η θέση του κόμβου σε ένα επίπεδο 2D υποδεικνύεται από τις συντεταγμένες X και Y. Το ποσό που απαιτείται για την παράδοση από κάθε κόμβο πελάτη εμφανίζεται επίσης στο πεδίο Ζήτηση.

**Demand Section**

Η ενότητα ζήτησης καθορίζει την ποσότητα παράδοσης που χρειάζεται κάθε πελάτης και αφορά μόνο τους κόμβους πελατών (όχι την αποθήκη).

**Depot Section**

Το αρχείο.vrp μπορεί προαιρετικά να παρέχει περισσότερες λεπτομέρειες για την αποθήκη, όπως τη θέση της και άλλα μοναδικά χαρακτηριστικά.

**Προαιρετικές πρόσθετες ενότητες**

Σε σπάνιες περιπτώσεις, τα αρχεία .vrp από το CVRPLIB ενδέχεται να έχουν επιπλέον ενότητες που περιλαμβάνουν σημαντικές πληροφορίες για συγκεκριμένες περιπτώσεις ή περιορισμούς ζητημάτων.

Συμπερασματικά, η κατανόηση της μορφής αρχείου.vrp από το CVRPLIB είναι απαραίτητη για την έρευνά μας σε περιπτώσεις CVRP. Αυτά τα αρχεία χρησιμεύουν ως βάση για την αλγοριθμική ανάλυση και τη δημιουργία λύσεων, καθώς περιέχουν κρίσιμες πληροφορίες σχετικά με τη διαμόρφωση του προβλήματος, όπως τις ιδιαιτερότητες του κόμβου, τις τιμές ζήτησης, τον πίνακα συντεταγμένων και τη χωρητικότητα του οχήματος.

**5.3 Παράδειγμα αρχείου**

NAME : A-n32-k5

COMMENT : (Augerat et al, No of trucks: 5, Optimal value: 784)

TYPE : CVRP

DIMENSION : 32

EDGE\_WEIGHT\_TYPE : EUC\_2D

CAPACITY : 100

NODE\_COORD\_SECTION

1 82 76

2 96 44

3 50 5

4 49 8

5 13 7

6 29 89

7 58 30

8 84 39

9 14 24

10 2 39

11 3 82

12 5 10

13 98 52

14 84 25

15 61 59

16 1 65

17 88 51

18 91 2

19 19 32

20 93 3

21 50 93

22 98 14

23 5 42

24 42 9

25 61 62

26 9 97

27 80 55

28 57 69

29 23 15

30 20 70

31 85 60

32 98 5

DEMAND\_SECTION

1 0

2 19

3 21

4 6

5 19

6 7

7 12

8 16

9 6

10 16

11 8

12 14

13 21

14 16

15 3

16 22

17 18

18 19

19 1

20 24

21 8

22 12

23 4

24 8

25 24

26 24

27 2

28 20

29 15

30 2

31 14

32 9

DEPOT\_SECTION

1

-1

EOF

Όπως αναφέραμε και νωρίτερα, η CVRPLIB παρέχει και ένα αντίστοιχο αρχείο .sol, που περιέχει τη βέλτιστη λύση κάθε προβλήματος. Για το παραπάνω πρόβλημα, η λύση που παρέχεται είναι η εξης:

Route #1: 21 31 19 17 13 7 26

Route #2: 12 1 16 30

Route #3: 27 24

Route #4: 29 18 8 9 22 15 10 25 5 20

Route #5: 14 28 11 4 23 3 2 6

Cost 784

**6 Αποτελέσματα**

Στη συγκεκριμένη ενότητα θα αναλύσουμε τα αποτελέσματα συγκεκριμένων προβλημάτων CVRP τα οποία υλοποιήθηκαν με τους λύτες Gurobi Optimization, LocalSolver, Google OR-Tools και VRPy, τα οποία αναλύσαμε παραπάνω. Για τα LocalSolver και Google OR-Tools χρησιμοποιήθηκαν τα έτοιμα APIs σε γλώσσα προγραμματισμού Python, ενώ για τα Gurobi και VRVy οι αντίστοιχοι πηγαίοι κώδικες δημιουργήθηκαν από την αρχή, με μικρές παραλλαγές. Το κριτήριο που συγκρίνουμε τους λύτες είναι η αποδοτικότητα τους στην εύρεση της καλύτερης λύσης, αν όχι της βέλτιστης. Για την άδεια χρήσης τους απαιτείται χρηματική συνδρομή, είτε ειδικό αίτημα δωρεάν χρήσης του για πανεπιστημιακούς φοιτητές. Τα προβλήματα επιλύθηκαν σε σύστημα με επεξεργαστή Intel i5-10400f, 16 GB RAM καθώς, RX 480 8GB κάρτα γραφικών και σκληρό δίσκο SSD 1TB. Όλα τα προβλήματα επιλύθηκαν μέσα σε χρονικό περιθώριο 120 δευτερολέπτων.

**6.1 Gurobi Optimization**

Στην επίλυση των προβλημάτων CVRP με το Gurobi, χρησιμοποιήθηκε ο parser της tsplib95, ώστε τα δεδομένα να περνούν από το “.vrp” αρχείο σε μορφή κώδικα.

problem = tsplib95.load("B-n63-k10.vrp")

Με τη συγκεκριμένη εντολή, φορτώνονται το πρόβλημα σε μια μεταβλητή problem και ύστερα έχουμε πρόσβαση στα εξής δεδομένα:

type - TYPE

dimension - DIMENSION

capacity - CAPACITY

edge\_weight\_type - EDGE\_WEIGHT\_TYPE

edge\_weight\_format - EDGE\_WEIGHT\_FORMAT

edge\_data\_format - EDGE\_DATA\_FORMAT

node\_coord\_type - NODE\_COORD\_TYPE

display\_data\_type - DISPLAY\_DATA\_TYPE

depots - DEPOT\_SECTION

demands - DEMAND\_SECTION

node\_coords - NODE\_COORD\_SECTION

edge\_weights - EDGE\_WEIGHT\_SECTION

display\_data - DISPLAY\_DATA\_SECTION

edge\_data - EDGE\_DATA\_SECTION

fixed\_edges - FIXED\_EDGES\_SECTION

tours - TOUR\_SECTION

Καθώς τα vrp αρχεία δεν παρέχουν την απόσταση των κόμβων μεταξύ τους, αλλά μόνο τις συντεταγμένες της θέσης τους, κρίθηκε απαραίτητο να υπολογιστεί η απόσταση τους μεταξύ τους και επιλέχθηκε η ευκλείδεια απόσταση.

*# for convenience, suppose that distances are Euclidean*

*def eucl\_dist(x1,y1,x2,y2):*

*return math.sqrt( (x1-x2)\*\*2 + (y1-y2)\*\*2 )*

*for i,j in G.edges:*

*(x1,y1) = my\_pos[i]*

*(x2,y2) = my\_pos[j]*

*G.edges[i,j]['length'] = eucl\_dist(x1,y1,x2,y2)*

Στη συνέχεια, εφαρμόζουμε όλους τους περιορισμούς τους περιορισμούς του προβλήματος, καθώς και αυτούς που περιγράφει το μαθηματικό μοντέλο MTZ. Όλα τα προβλήματα επιλύθηκαν με όλους τους λύτες με time limit 120 δευτερολέπτων, το οποίο αιτιολογεί, όπως θα δούμε, τη χαμηλότερη απόδοση του Gurobi με το MTZ. Διαπιστώθηκε, λοιπόν, ότι η το MTZ δεν παρήγαγε τις βέλτιστες λύσεις εντός του καθορισμένου χρονικού πλαισίου των 120 δευτερολέπτων. Αυτός ο περιορισμός είναι αποτέλεσμα πολλών λόγων. Προκειμένου να διασφαλιστεί ότι ικανοποιούνται οι περιορισμοί εξάλειψης της υποπεριήγησης, η διατύπωση MTZ εισάγει πρώτα επιπλέον μεταβλητές και περιορισμούς. Ενώ αυτές οι τροποποιήσεις είναι απαραίτητες για την αποτελεσματική προσομοίωση του CVRP, επεκτείνουν επίσης το μέγεθος και την πολυπλοκότητα του μαθηματικού μοντέλου. Ως αποτέλεσμα, ο λύτης πρέπει να εξετάσει έναν ευρύτερο χώρο λύσης και να λάβει υπόψη περισσότερους πιθανούς συνδυασμούς, γεγονός που αυξάνει τον χρόνο υπολογισμού. Ακόμα όμως, τα CVRP προβλήματα ανήκουν στην κατηγορία των NP-hard προβλημάτων τα οποία είναι υπολογιστικά δύσκολα ώστε να βρουν λύση εύκολα. Ακόμα και το Gurobi, που αποτελεί έναν από τους πιο αποτελεσματικούς λύτες, χρειάζεται χρόνο προκειμένου να βρει βέλτιστη λύση σε τέτοιου είδους προβλήματα.

Παρακάτω, βλέπουμε τον πίνακα των αποτελεσμάτων του Gurobi στα συγκεκριμένα προβλήματα CVRP που επιλέχθηκαν δειγματοληπτικά από την βιβλιοθήκη CVRPLIB:

|  |  |
| --- | --- |
| Τίτλος Προβλήματος | Αποτέλεσμα  (meters) |
| A\_n37\_k5 | 1062.78 |
| B\_n38\_k6 | 986.85 |
| A-n55-k9 | 1717.45 |
| P-n16-k8 | 1171.51 |
| A-n63-k10 | 2018.78 |
| B-n45-k5 | 951.71 |
| E-n51-k5 | 895.34 |
| E-n101-k8 | 1337 |
| B-n56-k7 | 1328.04 |
| B-n63-k10 | 1699.32 |

Πίνακας 1

**6.2 VRPy**

Το VRPy, όπως και το Gurobi, δε διαθέτουν έτοιμα κομμάτια κώδικα που να επιλύουν συγκεκριμένα προβλήματα CVRP, ο πηγαίος κώδικας γράφτηκε εξ’ ολοκλήρου από την αρχή, με τις κατάλληλες προσαρμογές ώστε το πρόβλημα να επιλύεται σωστά. Χρησιμοποιήθηκε και πάλι ο parser της tsplib95 ώστε να μεταφράζονται τα δεδομένα του προβλήματος.

Οι ευκλείδειες αποστάσεις μεταξύ όλων των κόμβων απεικονίζονται σε ένα DiGraph γράφημα (G), καθώς και οι επιπλέον πληροφορίες που αφορούν τον κάθε κόμβο ξεχωριστά.

Επιλέχθηκαν για επίλυση τα ίδια προβλήματα με τον λύτη Gurobi και τα αποτελέσματα ήταν τα παρακάτω:

|  |  |
| --- | --- |
| Τίτλος Προβλήματος | Αποτέλεσμα  (meters) |
| A\_n37\_k5 | 676.53 |
| B\_n38\_k6 | 790.25 |
| A-n55-k9 | 1117 |
| P-n16-k8 | 399 |
| A-n63-k10 | 1343 |
| B-n45-k5 | 771 |
| E-n51-k5 | 547 |
| E-n101-k8 | 833 |
| B-n56-k7 | 794 |
| B-n63-k10 | 1607 |

Πίνακας 2

**6.3 LocalSolver**

Όπως έχουμε αναφέρει, το LocalSolver παρέχει έτοιμα APIs για την επίλυση προβλημάτων CVRP, επομένως δεν χρειάστηκε να συντάξουμε δικό μας κώδικα. Τα αποτελέσματα του LocalSolver ήταν τα ιδανικότερα υπό τον περιορισμό των 120 δευτερολέπτων, όπως θα παρατηρήσουμε στη συνέχεια.

Πιο συγκεκριμένα, τα αποτελέσματα του LocalSolver στα παραπάνω προβλήματα είναι τα εξής:

|  |  |
| --- | --- |
| Τίτλος Προβλήματος | Αποτέλεσμα  (meters) |
| A\_n37\_k5 | 669 |
| B\_n38\_k6 | 805 |
| A-n55-k9 | 1073 |
| P-n16-k8 | 450 |
| A-n63-k10 | 1314 |
| B-n45-k5 | 751 |
| E-n51-k5 | 521 |
| E-n101-k8 | 815 |
| B-n56-k7 | 710 |
| B-n63-k10 | 1537 |

Πίνακας 3

**6.4 Google OR-Tools**

Όπως αναλύσαμε προηγουμένως τον κώδικα του προβλήματος, έτσι ακριβώς εφαρμόστηκε. Η μόνη διαφορά είναι πως το συγκεκριμένο API εφαρμόζει το πρόβλημα για την συλλογή παραγγελιών από των πελάτη και όχι την παράδοση τους, όπως θεωρήσαμε εμείς κατα την ανάπτυξη της εργασίας και των δοκιμών. Ο κώδικας λοιπόν, τροποποιήθηκε αναλόγως, ώστε οι παραγγελίες να φτάνουν στον πελάτη και όχι να συλλέγονται από αυτόν.

|  |  |
| --- | --- |
| Τίτλος Προβλήματος | Αποτέλεσμα  (meters) |
| A\_n37\_k5 | 669 |
| B\_n38\_k6 | 827 |
| A-n55-k9 | 1115 |
| P-n16-k8 | 450 |
| A-n63-k10 | 1366 |
| B-n45-k5 | 790 |
| E-n51-k5 | 560 |
| E-n101-k8 | 824 |
| B-n56-k7 | 734 |
| B-n63-k10 | 1600 |

Πίνακας 4

**6.5 Βέλτιστες λύσεις και σύγκριση**

Τα συγκεκριμένα προβλήματα που επιλύθηκαν με τους παραπάνω λύτες παρέχονται από την CVRPLIB μαζί με τις βέλτιστες λύσεις τους. Όπως θα δούμε παρακάτω, σε κάποιες περιπτώσεις οι λύτες βρήκαν βέλτιστες λύσεις, καλές λύσεις κοντά στις βέλτιστες, αλλά και μη ικανοποιητικές λύσεις δεδομένου του ελάχιστου χρόνου των 120 δευτερολέπτων που δόθηκαν για την εύρεση λύσης.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Τίτλος προβλήματος | Gurobi | VRPy | LocalSolver | OR-Tools | Βέλτιστη |
| A\_n37\_k5 | 1062.78 | 676.53 | 669 | 669 | **669** |
| B\_n38\_k6 | 986.85 | 805 | 805 | 827 | **805** |
| A-n55-k9 | 1717.45 | 1117 | 1073 | 1115 | **1073** |
| P-n16-k8 | 1171.51 | 450 | 450 | 450 | **450** |
| A-n63-k10 | 2018.78 | 1343 | 1314 | 1366 | **1314** |
| B-n45-k5 | 951.71 | 771 | 751 | 790 | **751** |
| E-n51-k5 | 895.34 | 547 | 521 | 560 | **521** |
| E-n101-k8 | 1337 | 833 | 815 | 824 | **817** |
| B-n56-k7 | 1328.04 | 794 | 710 | 734 | **707** |
| B-n63-k10 | 1699.32 | 1607 | 1537 | 1600 | **1496** |

Πίνακας 5

Στις περισσότερες περιπτώσεις, η καλύτερη λύση έχει βρεθεί από το LocalSolver, με εξαίρεση κάποιες περιπτώσεις ισοβαθμίας με το VRPy και το OR-Tools. Όπως εξηγήσαμε παραπάνω, η μη ικανοποιητική απόδοση του Gurobi οφείλεται στον λιγοστό χρόνο που δόθηκε για την εύρεση λύσης, ενώ το Gurobi με το MTZ μαθηματικό μοντέλο χρειάζονται μεγαλύτερο χρονικό περιθώριο ώστε να αποδώσουν κατάλληλα.

**Σφάλμα**

Σύμφωνα με τις δοσμένες λύσεις από τη CVRPLIB, μελετήσαμε τα αποτελέσματα των μεθόδων που χρησιμοποιήθηκαν για την επίλυση των προβλημάτων, ως προς το μέγιστο ποσοστό σφάλματος.

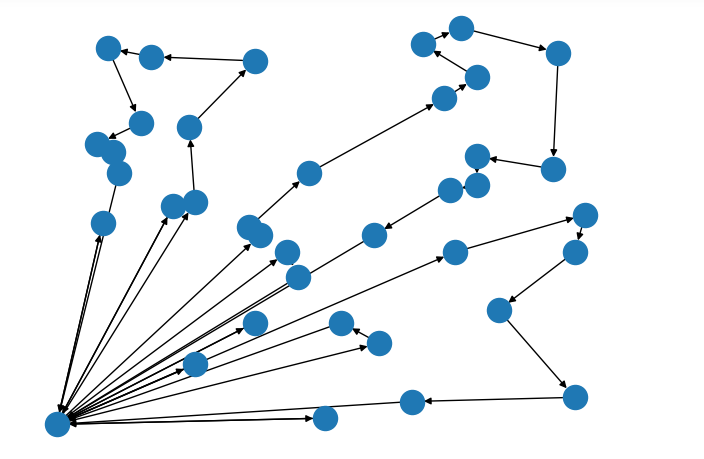
Gurobi: 58.83%

VRPy: 7,41%

OR-Tools: 6,95%

LocalSolver: 0%

Προκειμένου το αποτέλεσμα να έχει περισσότερο νόημα, αποφασίσαμε να οπτικοποιήσουμε τη λύση που βρήκε το Gurobi για το πρόβλημα “Α-n37-k5.vrp”:



Στην κάτω-αριστερή γωνία του παραπάνω σχήματος απεικονίζεται η αφετηρία των οχημάτων και όλοι οι υπόλοιποι κόμβοι αντικατοπτρίζουν του πελάτες που τα οχήματα καλούνται να επισκεφθούν και να εξυπηρετήσουν. Τα βέλη απεικονίζουν τις συγκεκριμένες διαδρομές που πρέπει να ακολουθήσουν τα οχήματα για να διανύσουν τη λιγότερη δυνατή απόσταση. Είναι επομένως η λύση που βρήκε ο Gurobi, με σφάλμα 58,8% από τη βέλτιστη λύση.

**7 Συμπεράσματα και μελλοντική έρευνα**

Η παρούσα πτυχιακή εργασία είχε ως στόχο να παρουσιάσει πως διάφορες τεχνικές με ευρετικές μεθόδους, αλγορίθμους και μαθηματικά μοντέλα βελτιστοποιουν προβλήματα δρομολόγησης οχημάτων.

Συγκεκριμένα, με τους λύτες Gurobi, LocalSolver, VRPy και OR-Tools λύσαμε δέκα διαφορετικά προβλήματα CVRP και συγκρίναμε τις διάφορες λύσεις τους με κριτήριο τη διαφορετική υλοποίησή τους. Για κάθε αποτέλεσμα λάβαμε υπόψη το σφάλμα που προέκυψε συγκριτικά με τις βέλτιστες λύσεις που παρέχονται από τη CVRPLIB.

Τελικώς, ο καλύτερος λύτης ήταν το LocalSolver με μηδενικό σφάλμα, ενώ το μεγαλύτερο σφάλμα εμφανίστηκε από το Gurobi με ποσοστό ίσο με 58,8%, με την λιγότερο ικανοποιητική απόδοση να προέρχεται από τον λιγοστό χρόνο που δόθηκε σε όλους τους λύτες για να λύσουν τα δεδομένα προβλήματα.

Μελλοντικά υπάρχει η βλέψη εφαρμογής παραπάνω λυτών, αλλά και η εφαρμογή τους σε προβλήματα με περισσότερους περιορισμούς, όπως προβλήματα δρομολόγησης με χρονικά παράθυρα παράδοσης (VRPTW). Επίσης θα ήταν ιδιαίτερα ενδιαφέρον να μελετήσουμε πραγματικά σενάρια εφαρμογής τέτοιων αλγορίθμων, σε συνδυασμό με μηχανική μάθηση και τεχνητή νοημοσύνη.

**8 Βιβλιογραφία**

Achuthan, N.R. and Caccetta, L. (1991). Integer linear programming formulation for a vehicle routing problem. *European Journal of Operational Research*, 52(1), pp.86–89. doi:https://doi.org/10.1016/0377-2217(91)90338-v.

Adyatama, A. (2020). *RPubs - Product Distribution with Nearest Neighbour and Genetic Algorithm*. [online] rpubs.com. Available at: https://rpubs.com/Argaadya/cvrp [Accessed 30 Jul. 2023].

Archetti, C. and Speranza, M.G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19(1-2), pp.3–22. doi:https://doi.org/10.1111/j.1475-3995.2011.00811.x.

Barbarosoglu, G. and Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research*, 26(3), pp.255–270. doi:https://doi.org/10.1016/s0305-0548(98)00047-1.

Bell, J.E. and McMullen, P.R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1), pp.41–48. doi:https://doi.org/10.1016/j.aei.2004.07.001.

Borčinová, Z. (2022). Kernel Search for the Capacitated Vehicle Routing Problem. *Applied Sciences*, [online] 12(22), p.11421. doi:https://doi.org/10.3390/app122211421.

Braekers, K., Ramaekers, K. and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, [online] 99, pp.300–313. doi:https://doi.org/10.1016/j.cie.2015.12.007.

Dantzig, G.B. and Ramser, J.H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1), pp.80–91. doi:https://doi.org/10.1287/mnsc.6.1.80.

Eksioglu, B., Vural, A.V. and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, [online] 57(4), pp.1472–1483. doi:https://doi.org/10.1016/j.cie.2009.05.009.

Elshaer, R. and Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, 140, p.106242. doi:https://doi.org/10.1016/j.cie.2019.106242.

Française D'automatique, O. and Recherche Opérationnelle (n.d.). *REVUE The vehicle routing problem*. [online] Available at: http://www.numdam.org/item/RO\_1976\_\_10\_1\_55\_0.pdf.

Gendreau, M. and Tarantilis, C.D. (2010). *Solving Large-Scale Vehicle Routing Problems with Time Windows: The State-of-the-Art*. [online] Available at: https://www.cirrelt.ca/documentstravail/cirrelt-2010-04.pdf.

Konstantakopoulos, G.D., Gayialis, S.P. and Kechagias, E.P. (2020a). Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. *Operational Research*. doi:https://doi.org/10.1007/s12351-020-00600-7.

Laporte, G. and Nobert, Y. (1987). Exact Algorithms for the Vehicle Routing Problem. pp.147–184. doi:https://doi.org/10.1016/s0304-0208(08)73235-3.

Laporte, G., Ropke, S. and Vidal, T. (2014). Chapter 4: Heuristics for the Vehicle Routing Problem. *Vehicle Routing*, pp.87–116. doi:<https://doi.org/10.1137/1.9781611973594.ch4>.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G. and Sormany, J.-S. (n.d.). New Heuristics for the Vehicle Routing Problem. *Logistics Systems: Design and Optimization*, pp.279–297. doi:https://doi.org/10.1007/0-387-24977-x\_9.

Laporte, G. and Semet, F. (2002). 5. Classical Heuristics for the Capacitated VRP. *The Vehicle Routing Problem*, pp.109–128. doi:https://doi.org/10.1137/1.9780898718515.ch5.

Montagné, R., Sanchez, D. and Storbugt, H. (2020). VRPy: A Python package for solving a range of vehicle routing problems with a column generation approach. *Journal of Open Source Software*, 5(55), p.2408. doi:https://doi.org/10.21105/joss.02408.

Mor, A. and Speranza, M.G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*. doi:https://doi.org/10.1007/s10479-021-04488-0.

Oyola, J. and Løkketangen, A. (2014). GRASP-ASP: An algorithm for the CVRP with route balancing. *Journal of Heuristics*, 20(4), pp.361–382. doi:https://doi.org/10.1007/s10732-014-9251-4.

Pessoa, A., Sadykov, R., Uchoa, E. and Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183(1-2), pp.483–523. doi:https://doi.org/10.1007/s10107-020-01523-z.

Ralphs, T.K., Kopman, L., Pulleyblank, W.R. and Trotter, L.E. (2003a). On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2-3), pp.343–359. doi:https://doi.org/10.1007/s10107-002-0323-0.

Toth, P. and Vigo, D. (1998). Exact Solution of the Vehicle Routing Problem. pp.1–31. doi:https://doi.org/10.1007/978-1-4615-5755-5\_1.

Yu, B.X., Yang, Z. and Xie, J.-H. (2011). A parallel improved ant colony optimization for multi-depot vehicle routing problem. 62(1), pp.183–188. doi:https://doi.org/10.1057/jors.2009.161.

Aydınalp, Z., & Özgen, D. (2022). Solving vehicle routing problem with time windows using metaheuristic approaches. International Journal of Intelligent Computing and Cybernetics, 1756-378X. Published on 13 May 2022. Issue published on 7 March 2023.

Ibrahim, A. A., Lo, N., Abdulaziz, R. O., & Ishaya, J. A. (2019). Capacitated Vehicle Routing Problem. Department of Mathematical Science, African Institute for Mathematical Science, Senegal. Department of Energy Engineering, PAUWES, University of Tlemcen, Algeria.

Reinelt, G. (1991). TSPLIB—A Traveling Salesman Problem Library. ORSA Journal on Computing, 3(4), 376–384. doi:10.1287/ijoc.3.4.376

Taunk, K., et al. “A Brief Review of Nearest Neighbor Algorithm for Learning and Classification.” IEEE Xplore, 1 May 2019, ieeexplore.ieee.org/document/9065747.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., & Subramanian, A. (2017). New benchmark instances for the Capacitated Vehicle Routing Problem. European Journal of Operational Research, 257(3), 845–858. doi:10.1016/j.ejor.2016.08.012

Lei, H., Laporte, G. and Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. Computers & Operations Research, 38(12), pp.1775–1783. doi:<https://doi.org/10.1016/j.cor.2011.02.007>.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21–27. doi:10.1109/tit.1967.1053964

Glover, F. (1989). Tabu Search—Part I. ORSA Journal on Computing, 1(3), 190–206. doi:10.1287/ijoc.1.3.190

Holland, J. H. (1984). Genetic Algorithms and Adaptation. Adaptive Control of Ill-Defined Systems, 317–333. doi:10.1007/978-1-4684-8941-5\_21

Van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). Simulated annealing. Simulated Annealing: Theory and Applications, 7–15. doi:10.1007/978-94-015-7744-1\_2

Tunnisaki, Fadlah, and Sutarman. “Clarke and Wright Savings Algorithm as Solutions Vehicle Routing Problem with Simultaneous Pickup Delivery (VRPSPD).” Journal of Physics: Conference Series, vol. 2421, no. 1, 1 Jan. 2023, p. 012045, <https://doi.org/10.1088/1742-6596/2421/1/012045>.

Dorigo, M., & Di Caro, G. (n.d.). Ant colony optimization: a new meta-heuristic. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). doi:10.1109/cec.1999.782657

Berger, J., Salois, M., & Begin, R. (1998). A hybrid genetic algorithm for the vehicle routing problem with time windows. Lecture Notes in Computer Science, 114–127. doi:10.1007/3-540-64575-6\_44