ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΗΝ ΧΡΟΝΟΔΡΟΜΟΛΟΓΗΣΗ ΟΧΗΜΑΤΩΝ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΑΠΟΘΗΚΩΝ

ΒΑΣΙΛΙΚΗ ΓΡΙΒΑΚΗ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΡΙΤΑΣ ΝΙΚΟΛΑΟΣ
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

Λαμία ..... έτος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΗΝ ΧΡΟΝΟΔΡΟΜΟΛΟΓΗΣΗ ΟΧΗΜΑΤΩΝ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΑΠΟΘΗΚΩΝ

ΒΑΣΙΛΙΚΗ ΓΡΙΒΑΚΗ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΡΙΤΑΣ ΝΙΚΟΛΑΟΣ
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

Λαμία ..... έτος 2025

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

DEVELOPMENT OF ALGORITHMS FOR VEHICLE SCHEDULING AND WAREHOUSE MANAGEMENT

VASILIKI GRIVAKI

FINAL THESIS

ADVISOR

TZIRITAS NIKOLAOS
ASSOCIATE PROFESSOR

Lamia ..... year 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 10/09/2025

Η Δηλούσα
Γριβάκη Βασιλική

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου (MDMFVRP), ένα σύνθετο πρόβλημα συνδυαστικής βελτιστοποίησης με σημαντικές εφαρμογές στη διαχείριση εφοδιαστικής αλυσίδας. Αναπτύχθηκε ένας νέος υβριδικός αλγόριθμος ο οποίος συνδυάζει τεχνικές ομαδοποίησης (K-means) για την αρχική ανάθεση πελατών σε αποθήκες, άπληστη στρατηγική για την κατανομή πελατών στα διαθέσιμα οχήματα διαφορετικών τύπων, Βελτιστοποίηση Αποικίας Μυρμηγκιών (ACO) για την κατασκευή και βελτιστοποίηση των δρομολογίων, τοπική αναζήτηση (2-opt) για περαιτέρω βελτίωση των διαδρομών, και μηχανισμό εξισορρόπησης φόρτου εργασίας μεταξύ των αποθηκών. Ο αλγόριθμος υλοποιήθηκε σε γλώσσα Python και αξιολογήθηκε η απόδοσή του σε ένα σύνολο τεχνητά παραγόμενων προβλημάτων αναφοράς διαφορετικών μεγεθών (μικρό, μεσαίο, μεγάλο), εξετάζοντας περιπτώσεις με και χωρίς χρονικά παράθυρα παράδοσης. Επιπλέον, διεξήχθη ανάλυση ευαισθησίας για τις βασικές παραμέτρους του αλγορίθμου ACO. Τα αποτελέσματα καταδεικνύουν ότι ο προτεινόμενος υβριδικός αλγόριθμος είναι ικανός να παράγει εφικτές και ποιοτικές λύσεις για το MDMFVRP, επιτυγχάνοντας καλή ισορροπία δρομολογίων και αξιοποίηση της χωρητικότητας του στόλου, ενώ η ανάλυση ευαισθησίας παρέχει χρήσιμες οδηγίες για τη βέλτιστη ρύθμιση των παραμέτρων του.

ABSTRACT

This bachelor thesis addresses the Multi-Depot Mixed Fleet Vehicle Routing Problem (MDMFVRP), a complex combinatorial optimization problem with significant applications in supply chain management. A novel hybrid algorithm was developed, combining clustering techniques (K-means) for initial customer-to-depot assignment, a greedy strategy for allocating customers to available vehicles of different types, Ant Colony Optimization (ACO) for route construction and optimization, local search (2-opt) for further route improvement, and a workload balancing mechanism between depots. The algorithm was implemented in Python and its performance was evaluated on a set of synthetically generated benchmark problems of varying sizes (small, medium, large), considering cases both with and without delivery time windows. Furthermore, a sensitivity analysis was conducted for the key parameters of the ACO algorithm. The results demonstrate that the proposed hybrid algorithm is capable of producing feasible and high-quality solutions for the MDMFVRP, achieving good route balance and fleet capacity utilization, while the sensitivity analysis provides useful guidelines for optimal parameter tuning.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Υπόβαθρο και Κίνητρο

Η αποτελεσματική διακίνηση αγαθών και υπηρεσιών αποτελεί μία από τις σημαντικότερες προκλήσεις των σύγχρονων logistic και της διαχείρισης της εφοδιαστικής αλυσίδας [1]. Καθώς το παγκόσμιο εμπόριο συνεχίζει να επεκτείνεται και οι προσδοκίες των καταναλωτών για ταχεία παράδοση γίνονται ολοένα και πιο απαιτητικές, οι οργανισμοί αντιμετωπίζουν αυξανόμενη πίεση για τη βελτιστοποίηση των δικτύων διανομής τους με παράλληλη ελαχιστοποίηση του λειτουργικού κόστους [2]. Στις αστικές περιοχές, η εξάπλωση του ηλεκτρονικού εμπορίου έχει μεταμορφώσει ριζικά τις λειτουργίες παράδοσης τελευταίου χιλιομέτρου (last mile), απαιτώντας από τους παρόχους εφοδιαστικής να προσαρμοστούν σε όλο και πιο σύνθετα πρότυπα παράδοσης [3]. Εν τω μεταξύ, στις αγροτικές και αναπτυσσόμενες περιοχές, οι προκλήσεις της παροχής αξιόπιστων υπηρεσιών παράδοσης σε τεράστιες αποστάσεις με περιορισμένες υποδομές δημιουργούν ένα εντελώς διαφορετικό σύνολο απαιτήσεων βελτιστοποίησης [4].

Στο επίκεντρο αυτών των λειτουργιών εφοδιαστικής βρίσκεται το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem - VRP), μια θεμελιώδης πρόκληση βελτιστοποίησης που απασχολεί ερευνητές και επαγγελματίες για πάνω από έξι δεκαετίες [5]. Από την αρχική διατύπωσή του από τους Dantzig και Ramser το 1959 ως «Πρόβλημα δρομολόγησης φορτηγών», το Πρόβλημα Δρομολόγησης Οχημάτων έχει εξελιχθεί σε μια πλούσια οικογένεια προβλημάτων, καθένα από τα οποία αποτυπώνει διαφορετικές πτυχές πραγματικών σεναρίων εφοδιαστικής [6]. Το κλασικό Πρόβλημα Δρομολόγησης Οχημάτων επιδιώκει να καθορίσει το βέλτιστο σύνολο δρομολογίων για ένα στόλο οχημάτων για την εξυπηρέτηση ενός συγκεκριμένου συνόλου πελατών, με την επιφύλαξη διαφόρων λειτουργικών περιορισμών, όπως η χωρητικότητα των οχημάτων, το μήκος της διαδρομής και τα χρονικά παράθυρα παράδοσης.

Ενώ έχουν γίνει σημαντικά βήματα στην επίλυση βασικών παραλλαγών του Προβλήματος Δρομολόγησης Οχημάτων, οι επιχειρήσεις εφοδιαστικής αλυσίδας στον πραγματικό κόσμο σπάνια συμμορφώνονται με αυτά τα απλοποιημένα μοντέλα. Τα σύγχρονα δίκτυα διανομής συχνά λειτουργούν από πολλαπλές αποθήκες, καθεμία από τις οποίες χρησιμεύει ως κόμβος για την αποστολή και την επιστροφή οχημάτων. Επιπλέον, οι πάροχοι εφοδιαστικής βασίζονται ολοένα και περισσότερο σε ετερογενείς στόλους που περιλαμβάνουν οχήματα με διαφορετική χωρητικότητα, λειτουργικό κόστος και χαρακτηριστικά απόδοσης. Τα παραδοσιακά φορτηγά διανομής λειτουργούν πλέον παράλληλα με εξειδικευμένα οχήματα, όπως μοτοσικλέτες για την πλοήγηση σε συμφορημένες αστικές περιοχές και, πιο πρόσφατα, μη επανδρωμένα αεροσκάφη για την ταχεία παράδοση από σημείο σε σημείο ευαίσθητων αντικειμένων.

Η παρούσα πτυχιακή ασχολείται με το Πρόβλημα Δρομολόγησης Οχημάτων μικτού στόλου πολλαπλών αποθηκών (Multi-Depot Mixed Fleet Vehicle Routing Problem - MDMFVRP), μια εξελιγμένη επέκταση του κλασικού VRP που αποτυπώνει αυτές τις πολυπλοκότητες του πραγματικού κόσμου. Το Πρόβλημα Δρομολόγησης Οχημάτων μικτού στόλου πολλαπλών αποθηκών ενσωματώνει δύο κρίσιμες διαστάσεις της πολυπλοκότητας: τη γεωγραφική διασπορά των πολλαπλών αποθηκών και τη λειτουργική ποικιλομορφία των ετερογενών στόλων οχημάτων. Με την ταυτόχρονη εξέταση αυτών των παραγόντων, το Πρόβλημα Δρομολόγησης Οχημάτων μικτού στόλου πολλαπλών αποθηκών παρέχει ένα πλαίσιο μοντελοποίησης που αντικατοπτρίζει με μεγαλύτερη ακρίβεια τις προκλήσεις που αντιμετωπίζουν οι σύγχρονες επιχειρήσεις εφοδιαστικής αλυσίδας, από τα δίκτυα παράδοσης μεγάλης κλίμακας έως την ανθρωπιστική εφοδιαστική σε σενάρια αντιμετώπισης καταστροφών.

Η σημασία της αποτελεσματικής επίλυσης του Προβλήματος Δρομολόγησης Οχημάτων εκτείνεται πέρα από το ακαδημαϊκό ενδιαφέρον. Σύμφωνα με εκτιμήσεις της Παγκόσμιας Τράπεζας, το κόστος των logistics αντιπροσωπεύει συνήθως το 8-10% του ΑΕΠ στις ανεπτυγμένες οικονομίες και έως και το 25% στις αναπτυσσόμενες οικονομίες. Στο πλαίσιο της δομής του κόστους εφοδιαστικής αλυσίδας, η μεταφορά αποτελεί συχνά τη μεγαλύτερη συνιστώσα, συνήθως 40-60% του συνολικού κόστους εφοδιαστικής. Ακόμα και οριακές βελτιώσεις στην αποτελεσματικότητα της

δρομολόγησης μπορούν να μεταφραστούν σε σημαντική εξοικονόμηση απόλυτου κόστους, μειωμένες περιβαλλοντικές επιπτώσεις και βελτιωμένα επίπεδα υπηρεσιών. Σε μια εποχή αυξημένης περιβαλλοντικής συνείδησης και αυστηρών κανονισμών για τις εκπομπές ρύπων, η βελτιστοποίηση των διαδρομών των οχημάτων δεν αποτελεί απλώς οικονομική ανάγκη, αλλά και περιβαλλοντική και κοινωνική ευθύνη.

Παρά την πρακτική του σημασία, το Πρόβλημα Δρομολόγησης Οχημάτων μικτού στόλου πολλαπλών αποθηκών παραμένει δύσκολο να επιλυθεί λόγω της εγγενούς υπολογιστικής πολυπλοκότητας του. Ως ένα NP-δύσκολο πρόβλημα, οι προσεγγίσεις ακριβούς λύσης καθίστανται υπολογιστικά δύσκολες για ρεαλιστικές περιπτώσεις προβλημάτων που περιλαμβάνουν δεκάδες αποθήκες και εκατοντάδες ή χιλιάδες πελάτες. Κατά συνέπεια, οι ευρετικές και μεταευρετικές προσεγγίσεις αποτελούν τους πιο υποσχόμενους δρόμους για την εύρεση λύσεων υψηλής ποιότητας σε λογικά υπολογιστικά χρονικά πλαίσια. Ωστόσο, οι υπάρχουσες προσεγγίσεις συχνά αντιμετωπίζουν μόνο συγκεκριμένες πτυχές του προβλήματος ή κάνουν απλουστευτικές παραδοχές που περιορίζουν τη δυνατότητά τους να εφαρμοστούν σε σενάρια του πραγματικού κόσμου.

Η παρούσα έρευνα παρακινείται από την ανάγκη για πιο ολοκληρωμένες, αποδοτικές και προσαρμοστικές μεθοδολογίες επίλυσης του Προβλήματος Δρομολόγησης Οχημάτων. Με την ανάπτυξη ενός υβριδικού αλγορίθμου που συνδυάζει τα πλεονεκτήματα διαφορετικών τεχνικών βελτιστοποίησης, η παρούσα διατριβή στοχεύει να προωθήσει την κατάσταση της τεχνολογίας στη βελτιστοποίηση δρομολόγησης οχημάτων και να παρέχει πρακτικά εργαλεία για τους σχεδιαστές logistics που αντιμετωπίζουν όλο και πιο σύνθετες προκλήσεις διανομής.

Στόχοι και Συνεισφορές της Έρευνας

Η παρούσα πτυχιακή αποσκοπεί στην ανάπτυξη, ανάλυση και επικύρωση ενός νέου υβριδικού αλγορίθμου για την επίλυση του προβλήματος δρομολόγησης οχημάτων με μεικτό στόλο πολλαπλών αποθηκών με χρονικά παράθυρα. Η έρευνα καθοδηγείται από τους ακόλουθους πρωταρχικούς στόχους:

1. Διατύπωση ενός ολοκληρωμένου μαθηματικού μοντέλου του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, το οποίο αποτυπώνει τους βασικούς περιορισμούς και στόχους των επιχειρήσεων logistics του πραγματικού κόσμου, συμπεριλαμβανομένων των πολλαπλών αποθηκών, των ετερογενών στόλων οχημάτων, των περιορισμών χωρητικότητας και των χρονικών παραθύρων για την εξυπηρέτηση των πελατών.
2. Ανάπτυξη ενός υβριδικού αλγορίθμου που αντιμετωπίζει αποτελεσματικά την πολύπλευρη φύση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου ενσωματώνοντας τεχνικές ομαδοποίησης, άπληστες ευρετικές τεχνικές, μεταευρετική βελτιστοποίηση και διαδικασίες τοπικής αναζήτησης.
3. Να αναλύσει τις θεωρητικές ιδιότητες του προτεινόμενου αλγορίθμου, συμπεριλαμβανομένης της υπολογιστικής πολυπλοκότητας, της ευαισθησίας στις παραμέτρους και της συμπεριφοράς σύγκλισης, παρέχοντας πληροφορίες σχετικά με την επεκτασιμότητα και την αξιοπιστία του.
4. Να αξιολογήσει την απόδοση του αλγορίθμου σε σχέση με καθιερωμένα προβλήματα αναφοράς και εναλλακτικές προσεγγίσεις επίλυσης, αποδεικνύοντας την αποτελεσματικότητά του σε διαφορετικά χαρακτηριστικά και μεγέθη προβλημάτων.
5. Εφαρμογή του αλγορίθμου σε ρεαλιστικές περιπτώσεις, καταδεικνύοντας την πρακτική του χρησιμότητα και την προσαρμοστικότητά του σε διαφορετικά πλαίσια εφοδιαστικής.

Η έρευνα αυτή συμβάλλει σημαντικά στον τομέα της βελτιστοποίησης της δρομολόγησης οχημάτων:

Η πτυχιακή παρουσιάζει έναν νέο υβριδικό αλγόριθμο που ενσωματώνει την ομαδοποίηση K-μέσων για την ανάθεση αποθηκών, την κατανομή οχημάτων με βάση τη χωρητικότητα, τη βελτιστοποίηση αποικίας μυρμηγκιών για την κατασκευή διαδρομής και την τοπική αναζήτηση 2-opt για τη βελτίωση της διαδρομής. Αυτή η ολοκληρωμένη προσέγγιση αντιμετωπίζει την

ιεραρχική δομή λήψης αποφάσεων που υπάρχει στο Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, από τη στρατηγική ανάθεση αποθηκών έως την τακτική κατανομή οχημάτων και τον επιχειρησιακό σχεδιασμό διαδρομών.

Δεύτερον, η έρευνα μας εισάγει μια βελτιωμένη εφαρμογή Βελτιστοποίησης Αποικίας Μυρμηγκιών ειδικά προσαρμοσμένη για το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, ενσωματώνοντας περιορισμούς χρονικών παραθύρων και ειδικά χαρακτηριστικά οχημάτων στους μηχανισμούς ενημέρωσης φερομόνης και κατασκευής λύσεων. Αυτή η προσαρμογή επιτρέπει τον αποτελεσματικότερο χειρισμό των χρονικών περιορισμών, οι οποίοι είναι ζωτικής σημασίας στις σύγχρονες επιχειρήσεις logistics με ακριβή χρονοδιαγράμματα παράδοσης.

Τρίτον, η πτυχιακή παρέχει μια ολοκληρωμένη ανάλυση των παραμέτρων του αλγορίθμου και των αλληλεπιδράσεων τους, προσφέροντας πολύτιμες γνώσεις για τον συντονισμό του αλγορίθμου και την προσαρμογή σε διαφορετικά χαρακτηριστικά του προβλήματος. Η ανάλυση αυτή γεφυρώνει το χάσμα μεταξύ του θεωρητικού σχεδιασμού αλγορίθμων και των πρακτικών εκτιμήσεων εφαρμογής.

Τέταρτον, η συγκριτική αξιολόγηση έναντι βασικών αλγορίθμων και προβλημάτων αναφοράς καθορίζει τα σχετικά πλεονεκτήματα και τους περιορισμούς της προτεινόμενης προσέγγισης, συμβάλλοντας στην ευρύτερη κατανόηση της απόδοσης των αλγορίθμων στο πλαίσιο πολύπλοκων προβλημάτων δρομολόγησης οχημάτων.

Τέλος, οι μελέτες διάφορων περιπτώσεων καταδεικνύουν την πρακτική εφαρμογή του αλγορίθμου σε πραγματικά σενάρια εφοδιαστικής, καταδεικνύοντας πώς οι θεωρητικές εξελίξεις στους αλγορίθμους βελτιστοποίησης μπορούν να μεταφραστούν σε απτές βελτιώσεις στην επιχειρησιακή αποδοτικότητα.

Πλαίσιο και Σημασία του Προβλήματος

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου αποτελεί μια κρίσιμη πρόκληση βελτιστοποίησης στο σημείο τομής της επιχειρησιακής έρευνας, της διαχείρισης εφοδιαστικής και της υπολογιστικής βελτιστοποίησης. Για να εκτιμήσουμε πλήρως τη σημασία του, πρέπει να το εντάξουμε στο ευρύτερο πλαίσιο της βελτιστοποίησης της εφοδιαστικής και να κατανοήσουμε πώς η αποτελεσματική του λύση επηρεάζει διάφορους ενδιαφερόμενους.

Εξέλιξη των Προβλημάτων Δρομολόγησης Οχημάτων

Η μελέτη της δρομολόγησης οχημάτων έχει εξελιχθεί σημαντικά από την αρχική διατύπωση του προβλήματος της δρομολόγησης φορτηγών. Η αρχική έρευνα επικεντρώθηκε κυρίως στο Πρόβλημα Δρομολόγησης Οχημάτων με Χωρητικότητα (Capacitated Vehicle Routing Problem - CVRP), το οποίο εξετάζει μια ενιαία αποθήκη και έναν ομοιογενή στόλο με περιορισμούς χωρητικότητας. Αυτό επεκτάθηκε αργότερα για να ενσωματώσει χρονικά παράθυρα (VRPTW), πολλαπλές αποθήκες (MDVRP), ετερογενείς στόλους (HFVRP) και διάφορους συνδυασμούς αυτών.

Η εξέλιξη από τα απλά προβλήματα δρομολόγησης προς περισσότερο πολύπλοκες παραλλαγές αντανακλά την αυξανόμενη πολυπλοκότητα των επιχειρήσεων εφοδιαστικής. Καθώς οι επιχειρήσεις επεκτάθηκαν γεωγραφικά, οι λειτουργίες πολλαπλών αποθηκών έγιναν ο κανόνας [7]. Καθώς διαφοροποιήθηκαν οι απαιτήσεις παράδοσης - από μαζικές αποστολές έως ευαίσθητα στον χρόνο μικρά δέματα - προέκυψε η ανάγκη για εξειδικευμένα οχήματα [8]. Η άνοδος του ηλεκτρονικού εμπορίου, με την έμφαση που δίνει στα γρήγορα και ακριβή παράθυρα παράδοσης, αύξησε περαιτέρω τη σημασία των χρονικών περιορισμών στις αποφάσεις δρομολόγησης [9].

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου με χρονικά παράθυρα αντιπροσωπεύει τη συμβολή αυτών των εξελικτικών μονοπατιών, αποτυπώνοντας την πολύπλευρη φύση των σύγχρονων επιχειρήσεων εφοδιαστικής. Αντιμετωπίζοντας ταυτόχρονα την ανάθεση αποθηκών, την επιλογή οχημάτων και την κατασκευή διαδρομών, το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου παρέχει ένα πιο ολιστικό

πλαίσιο βελτιστοποίησης από τις απλούστερες παραλλαγές του Προβλήματος Δρομολόγησης Οχημάτων.

Πρακτικές Εφαρμογές και Αντίκτυπος

Οι εφαρμογές του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου εκτείνονται σε πολλούς τομείς όπου τα logistics διανομής παίζουν κρίσιμο ρόλο:

Στο λιανικό εμπόριο και το ηλεκτρονικό εμπόριο, οι εταιρείες λειτουργούν ολοένα και περισσότερο πολλαπλά κέντρα εκπλήρωσης που εξυπηρετούν επικαλυπτόμενες γεωγραφικές περιοχές. Αυτές οι επιχειρήσεις χρησιμοποιούν συνήθως διάφορους τύπους οχημάτων, από μεγάλα φορτηγά για μαζικές παραδόσεις έως μικρότερα οχήματα για αστικές περιοχές, ακόμη και μη επανδρωμένα αεροσκάφη για υπηρεσίες γρήγορης παράδοσης υψηλής ποιότητας [10]. Η βελτιστοποίηση αυτών των λειτουργιών επηρεάζει άμεσα το κόστος παράδοσης, τα επίπεδα υπηρεσιών και την ικανοποίηση των πελατών.

Στα ανθρωπιστικά logistics, η ανταπόκριση σε φυσικές καταστροφές ή κρίσεις υγείας περιλαμβάνει συχνά το συντονισμό προμηθειών βοήθειας από πολλαπλά σημεία διανομής με τη χρήση οποιουδήποτε οχήματος είναι διαθέσιμο - από φορτηγά και βανάκια μέχρι μοτοσικλέτες και ακόμη και ποδήλατα σε δύσβατα εδάφη. Η αποτελεσματική δρομολόγηση σε αυτά τα πλαίσια μπορεί να επηρεάσει δραματικά την έγκαιρη παράδοση της βοήθειας και, κατά συνέπεια, τις ζωές που σώζονται.

Στις υπηρεσίες κοινής ωφέλειας και στις επιχειρήσεις συντήρησης, οι τεχνικοί πρέπει να δρομολογούνται από πολλαπλές αποθήκες σε τοποθεσίες εξυπηρέτησης με συγκεκριμένα χρονικά παράθυρα για υπηρεσίες που έχουν προκαθοριστεί με ραντεβού. Οι εταιρείες σε αυτόν τον τομέα συχνά διατηρούν ετερογενείς στόλους για να ανταποκρίνονται στις διαφορετικές απαιτήσεις των εργασιών, από οχήματα εξειδικευμένου εξοπλισμού έως μικρότερα επιβατικά οχήματα για επιθεωρήσεις [11].

Στις δημόσιες συγκοινωνίες και στις υπηρεσίες κοινής κινητικότητας, οι φορείς εκμετάλλευσης πρέπει να βελτιστοποιούν τις διαδρομές από πολλαπλές βάσεις οχημάτων χρησιμοποιώντας διαφορετικούς τύπους οχημάτων βάσει των αναγκών χωρητικότητας. Η ενσωμάτωση των υπηρεσιών κατά παραγγελία (on-demand services) και των υπηρεσιών σταθερής διαδρομής (fixed-route services) περιπλέκει περαιτέρω αυτές τις αποφάσεις δρομολόγησης [12].

Ο οικονομικός αντίκτυπος της βελτίωσης των λύσεων Προβλημάτων Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου είναι σημαντικός. Σύμφωνα με την βιβλιογραφία, οι μεταφορές αντιπροσωπεύουν συνήθως το 40-50% του συνολικού κόστους εφοδιαστικής και τα καύσιμα από μόνα τους αντιπροσωπεύουν το 30-40% του κόστους λειτουργίας των οχημάτων [13]. Η βελτιστοποίηση της διαδρομής μπορεί να μειώσει την κατανάλωση καυσίμων κατά 5-15%, μειώνοντας έτσι τόσο το λειτουργικό κόστος όσο και τις περιβαλλοντικές επιπτώσεις [14]. Σε επιχειρήσεις μεγάλης κλίμακας, ακόμη και μια μείωση του κόστους διανομής κατά 1% μπορεί να μεταφραστεί σε ετήσια εξοικονόμηση εκατομμυρίων δολαρίων.

Πέρα από τα άμεσα οικονομικά οφέλη, η αποτελεσματική δρομολόγηση συμβάλλει στην περιβαλλοντική βιωσιμότητα, μειώνοντας την κατανάλωση καυσίμων και τις εκπομπές ρύπων. Επίσης, βελτιώνει την ποιότητα των υπηρεσιών μέσω πιο αξιόπιστων χρόνων παράδοσης και μεγαλύτερης ανταπόκρισης στις ανάγκες των πελατών. Αν και η παρούσα πτυχιακή δεν εστιάζει ειδικά σε τομείς με κρίσιμο χρόνο, όπως οι υπηρεσίες έκτακτης ανάγκης και η παράδοση ευαίσθητων αγαθών, οι προτεινόμενες μέθοδοι μπορούν να προσαρμοστούν για τέτοιες εφαρμογές όπου η αποτελεσματικότητα της δρομολόγησης είναι κρίσιμη.

Υπολογιστικές Προκλήσεις και Προσεγγίσεις

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου ανήκει στην κατηγορία των NP-δύσκολων προβλημάτων συνδυαστικής βελτιστοποίησης [15], που σημαίνει ότι οι ακριβείς μέθοδοι επίλυσης γίνονται υπολογιστικά δύσκολες όσο αυξάνεται το μέγεθος του προβλήματος. Η υπολογιστική πολυπλοκότητα προκύπτει από την αλληλεπίδραση πολλαπλών επιπέδων απόφασης:

- Ανάθεση πελατών στις κατάλληλες αποθήκες
- Κατανομή οχημάτων από τον ετερογενή στόλο κάθε αποθήκης
- Κατασκευή εφικτών διαδρομών για κάθε όχημα
- Διασφάλιση της τήρησης των περιορισμών χωρητικότητας και χρονικών παραθύρων

Για ένα πρόβλημα με m αποθήκες, n πελάτες και κατά μέσο όρο k οχήματα ανά αποθήκη, ο χώρος λύσεων αυξάνεται εκθετικά. Μόνο ο αριθμός των πιθανών αναθέσεων μεταξύ πελατών και αποθηκών είναι της τάξης του m^n, και για κάθε ανάθεση υπάρχουν πολλές πιθανές κατανομές οχημάτων και διαμορφώσεις διαδρομών.

Δεδομένης αυτής της υπολογιστικής πολυπλοκότητας, η έρευνα έχει επικεντρωθεί στην ανάπτυξη ευρετικών και μεταευρετικών προσεγγίσεων που μπορούν να βρουν λύσεις υψηλής ποιότητας σε λογικά υπολογιστικά χρονικά πλαίσια. Οι πρώτες προσεγγίσεις, όπως αυτές των Gillett και Miller (1974) και των Wren και Holliday (1972), συχνά αποσυνέθεταν το πρόβλημα σε διαδοχικά υποπροβλήματα: πρώτα ανάθεση αποθήκης, στη συνέχεια κατανομή οχημάτων, και τέλος δρομολόγηση [16, 17]. Αυτή η διαδοχική προσέγγιση οδηγεί συχνά σε μη βέλτιστες συνολικές λύσεις.

Πιο πρόσφατες έρευνες έχουν διερευνήσει ολοκληρωμένες προσεγγίσεις με μεταευρετικές μεθόδους. Οι Cordeau et al. (1997) πρότειναν έναν αλγόριθμο tabu search [18], ενώ οι Salhi και Sari (1997) ανέπτυξαν μια μέθοδο multi-level composite heuristic [19]. Οι Renaud et al. (1996) χρησιμοποίησαν γενετικούς αλγορίθμους [20], και οι Crevier et al. (2007) εφάρμοσαν adaptive large neighborhood search [21]. Αυτές οι προσεγγίσεις προσπαθούν να περιηγηθούν στον πολύπλοκο χώρο λύσεων πιο ολιστικά, λαμβάνοντας υπόψη τις αλληλεξαρτήσεις μεταξύ των διαφόρων επιπέδων λήψης αποφάσεων.

Ο υβριδικός αλγόριθμος που προτείνεται στην παρούσα διατριβή στηρίζεται σε αυτή την εξελικτική πορεία, συνδυάζοντας τα οργανωτικά πλεονεκτήματα των τεχνικών ομαδοποίησης με τις δυνατότητες προσαρμοστικής εξερεύνησης της βελτιστοποίησης αποικίας μυρμηγκιών και τις δυνατότητες βελτίωσης των διαδικασιών τοπικής αναζήτησης. Αυτή η ενσωμάτωση αποσκοπεί στην αντιμετώπιση της πολυεπίπεδης δομής αποφάσεων του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, διατηρώντας παράλληλα την υπολογιστική αποδοτικότητα.

Μεθοδολογία Έρευνας

Η παρούσα έρευνα υιοθετεί μια ολοκληρωμένη μεθοδολογία που συνδυάζει τη θεωρητική ανάλυση, την ανάπτυξη αλγορίθμων και την εμπειρική αξιολόγηση. Η μεθοδολογική προσέγγιση περιλαμβάνει διάφορες αλληλένδετες φάσεις.

Διατύπωση του Προβλήματος και Μαθηματική Μοντελοποίηση

Η έρευνα αρχίζει με την επίσημη μαθηματική διατύπωση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου με χρονικά παράθυρα. Η διατύπωση αυτή ορίζει τις μεταβλητές απόφασης του προβλήματος, τους περιορισμούς και την αντικειμενική συνάρτηση, παρέχοντας μια ακριβή μαθηματική αναπαράσταση που χρησιμεύει ως βάση για την ανάπτυξη αλγορίθμων.

Το μαθηματικό μοντέλο ενσωματώνει βασικούς επιχειρησιακούς περιορισμούς, όπως:

- Περιορισμοί χωρητικότητας οχημάτων
- Περιορισμοί χρονικών παραθύρων για την εξυπηρέτηση πελατών
- Διαθεσιμότητα οχημάτων για συγκεκριμένη αποθήκη
- Απαιτήσεις συνέχειας της διαδρομής και επιστροφής στην αποθήκη

Η αντικειμενική συνάρτηση έχει σχεδιαστεί για την ελαχιστοποίηση του συνολικού λειτουργικού κόστους, ενσωματώνοντας το κόστος βάσει απόστασης (το οποίο διαφέρει ανάλογα με τον τύπο του

οχήματος), το κόστος βάσει χρόνου (συμπεριλαμβανομένων των ποινών για το χρόνο αναμονής) και το σταθερό κόστος χρήσης του οχήματος.

Αυτό το επίσημο μοντέλο εξυπηρετεί πολλαπλούς σκοπούς: παρέχει σαφήνεια και ακρίβεια στον ορισμό του προβλήματος, επιτρέπει τη σύγκριση με άλλες παραλλαγές Προβλημάτων Δρομολόγησης Οχημάτων και δημιουργεί ένα θεωρητικό πλαίσιο για την αξιολόγηση της ποιότητας των λύσεων.

Σχεδιασμός και Υλοποίηση Αλγορίθμου

Με βάση τη μαθηματική διατύπωση, σχεδιάζεται ένας υβριδικός αλγόριθμος για την αντιμετώπιση των ιδιαίτερων χαρακτηριστικών και προκλήσεων του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου. Ο αλγόριθμος ενσωματώνει διάφορα στοιχεία:

- Ομαδοποίηση K-μέσων για την αρχική ανάθεση πελάτη σε αποθήκη
- Άπληστη ανάθεση με προτεραιότητα στη χωρητικότητα για την κατανομή των πελατών στα οχήματα
- Βελτιστοποίηση Αποικίας Μυρμηγκιών για την κατασκευή και βελτιστοποίηση διαδρομής
- 2-opt τοπική αναζήτηση για τη βελτίωση της διαδρομής
- Εξισορρόπηση φόρτου εργασίας για εξισορρόπηση φορτίου αποθήκης

Κάθε μέρος είναι προσεκτικά σχεδιασμένο ώστε να αντιμετωπίζει συγκεκριμένες πτυχές του προβλήματος, με ιδιαίτερη προσοχή στις αλληλεπιδράσεις μεταξύ τους. Η στρατηγική ολοκλήρωσης διασφαλίζει ότι οι αποφάσεις που λαμβάνονται σε κάθε στάδιο υποστηρίζουν και ενισχύουν τα επόμενα βήματα βελτιστοποίησης.

Ο αλγόριθμος υλοποιείται σε Python, αξιοποιώντας βιβλιοθήκες επιστημονικών υπολογιστών όπως η NumPy για αριθμητικές πράξεις και η Scikit-learn για το μέρος της ομαδοποίησης K-μέσων. Αυτή η υλοποίηση παρέχει μια ευέλικτη πλατφόρμα για δοκιμές αλγορίθμων, ρύθμιση παραμέτρων και αξιολόγηση επιδόσεων.

Θεωρητική Ανάλυση

Οι θεωρητικές ιδιότητες του αλγορίθμου αναλύονται από πολλαπλές οπτικές γωνίες.

Η ανάλυση υπολογιστικής πολυπλοκότητας εξετάζει την επεκτασιμότητα του αλγορίθμου σε σχέση με το μέγεθος του προβλήματος, εντοπίζοντας πιθανά υπολογιστικά σημεία συμφόρησης και ευκαιρίες βελτιστοποίησης.

Η ανάλυση ευαισθησίας παραμέτρων διερευνά τον τρόπο με τον οποίο η απόδοση του αλγορίθμου επηρεάζεται από βασικές παραμέτρους, ιδίως εκείνες που διέπουν το στοιχείο της Βελτιστοποίησης Αποικίας Μυρμηγκιών. Η ανάλυση αυτή βοηθά στον καθορισμό καλών ρυθμίσεων παραμέτρων και στην κατανόηση των συμβιβασμών που συνεπάγεται ο συντονισμός των παραμέτρων.

Η ανάλυση σύγκλισης διερευνά τον τρόπο με τον οποίο ο αλγόριθμος εξελίσσεται προς λύσεις υψηλής ποιότητας κατά τη διάρκεια των επαναλήψεων, εντοπίζοντας πρότυπα σύγκλισης και ενημερώνοντας για τα κριτήρια διακοπής για πρακτικές υλοποιήσεις.

Αυτή η θεωρητική ανάλυση συμπληρώνει την εμπειρική αξιολόγηση παρέχοντας πληροφορίες για τη συμπεριφορά του αλγορίθμου και καθορίζοντας τις προσδοκίες για την απόδοσή του σε διαφορετικά πλαίσια προβλημάτων.

Εμπειρική Αξιολόγηση

Η απόδοση του αλγορίθμου αξιολογείται μέσω εκτεταμένων υπολογιστικών πειραμάτων με τη χρήση τόσο προβλημάτων αναφοράς όσο και ρεαλιστικών μελετών περιπτώσεων.

Οι δοκιμές αναφοράς χρησιμοποιούν καθιερωμένες περιπτώσεις Προβλημάτων Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου και Προβλημάτων Δρομολόγησης Οχημάτων Μικτού Στόλου από τη βιβλιογραφία, προσαρμοσμένες ώστε να ενσωματώνουν μικτούς στόλους και

χρονικά παράθυρα, όπως απαιτείται. Αυτό επιτρέπει τη σύγκριση με υπάρχουσες προσεγγίσεις λύσεων και επικυρώνει την αποτελεσματικότητα του αλγορίθμου σε διαφορετικά χαρακτηριστικά προβλημάτων.

Ο έλεγχος ευαισθησίας μεταβάλλει συστηματικά τις παραμέτρους του προβλήματος, όπως την κατανομή των πελατών, το εύρος του χρονικού παραθύρου και τη σύνθεση του στόλου, για να κατανοηθεί πώς αυτοί οι παράγοντες επηρεάζουν την απόδοση του αλγορίθμου.

Η συγκριτική αξιολόγηση αξιολογεί τον υβριδικό αλγόριθμο έναντι βασικών προσεγγίσεων, συμπεριλαμβανομένης μιας καθαρά άπληστης προσέγγισης και ενός αλγορίθμου K-μέσων συν τον αλγόριθμο εξοικονόμησης Clarke-Wright. Η σύγκριση αυτή αναδεικνύει τα σχετικά πλεονεκτήματα της υβριδικής προσέγγισης.

Μελέτες περιπτώσεων εφαρμόζουν τον αλγόριθμο σε ρεαλιστικά σενάρια εφοδιαστικής που προέρχονται από δεδομένα του πραγματικού κόσμου, αποδεικνύοντας την πρακτική χρησιμότητα και την προσαρμοστικότητά του σε διαφορετικά επιχειρησιακά πλαίσια.

Η εμπειρική αξιολόγηση χρησιμοποιεί πολλαπλές μετρήσεις επιδόσεων, όπως το συνολικό λειτουργικό κόστος, το ισοζύγιο διαδρομών, η χρήση της χωρητικότητας και ο υπολογιστικός χρόνος, παρέχοντας μια πολύπλευρη αξιολόγηση της αποτελεσματικότητας του αλγορίθμου.

Προβληματισμοί Υλοποίησης και Πρακτικές Οδηγίες

Πέρα από τη θεωρητική ανάπτυξη και αξιολόγηση, η έρευνα μας ασχολείται με πρακτικά ζητήματα για την εφαρμογή του αλγορίθμου σε συστήματα σχεδιασμού logistics στον πραγματικό κόσμο.

Οι απαιτήσεις προεπεξεργασίας των δεδομένων και οι τεχνικές για το χειρισμό γεωγραφικών και επιχειρησιακών δεδομένων του πραγματικού κόσμου είναι προς συζήτηση.

Παρέχονται, επίσης, στρατηγικές ρύθμισης παραμέτρων, προσφέροντας κατευθυντήριες γραμμές για την προσαρμογή του αλγορίθμου σε συγκεκριμένα επιχειρησιακά πλαίσια και χαρακτηριστικά του προβλήματος.

Αναλύονται οι απαιτήσεις σε υπολογιστικούς πόρους, με συστάσεις για την εφαρμογή σε διαφορετικά υπολογιστικά περιβάλλοντα, από εφαρμογές σχεδιασμού γραφείου έως υπηρεσίες βελτιστοποίησης που βασίζονται στο νέφος.

Αυτή η πρακτική καθοδήγηση γεφυρώνει το χάσμα μεταξύ της θεωρητικής ανάπτυξης αλγορίθμων και της υλοποίησης στον πραγματικό κόσμο, ενισχύοντας τον αντίκτυπο και τη δυνατότητα εφαρμογής της έρευνας.

Δομή Πτυχιακής

Η παρούσα πτυχιακή είναι οργανωμένη σε πέντε κεφάλαια που αναπτύσσουν, αναλύουν και αξιολογούν συστηματικά την προτεινόμενη προσέγγιση για την επίλυση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου.

Κεφάλαιο 1: Εισαγωγή παρέχει το ιστορικό, τα κίνητρα, τους στόχους και τη μεθοδολογία της έρευνας. Καθορίζει τη σημασία του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου στις σύγχρονες επιχειρήσεις logistics και περιγράφει τις συνεισφορές της έρευνας.

Κεφάλαιο 2: Βιβλιογραφική ανασκόπηση εξετάζει την υπάρχουσα βιβλιογραφία σχετικά με τα προβλήματα δρομολόγησης οχημάτων, με ιδιαίτερη έμφαση στις παραλλαγές πολλαπλών αποθηκών και ετερογενών στόλων. Αναλύονται κριτικά οι προηγούμενες προσεγγίσεις λύσεων, εντοπίζοντας τα δυνατά σημεία, τους περιορισμούς και τις ευκαιρίες εξέλιξής τους. Η ανασκόπηση περιλαμβάνει ακριβείς μεθόδους, κλασικές ευρετικές και μεταευρετικές προσεγγίσεις, δημιουργώντας τη θεωρητική και μεθοδολογική βάση για την παρούσα έρευνα.

Κεφάλαιο 3: Διατύπωση του προβλήματος και σχεδιασμός του αλγορίθμου παρουσιάζει μια ολοκληρωμένη μαθηματική διατύπωση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου και περιγράφει λεπτομερώς την ανάπτυξη του υβριδικού αλγορίθμου. Εξηγείται η λογική πίσω από κάθε μέρος του αλγορίθμου και ο τρόπος με τον οποίο

ενσωματώνονται για την αντιμετώπιση της πολύπλευρης φύσης του προβλήματος. Το κεφάλαιο αυτό περιλαμβάνει επίσης μια διεξοδική θεωρητική ανάλυση της υπολογιστικής πολυπλοκότητας του αλγορίθμου, της ευαισθησίας στις παραμέτρους και των ιδιοτήτων σύγκλισης.

Κεφάλαιο 4: Υπολογιστικά αποτελέσματα και ανάλυση παρουσιάζονται τα ευρήματα από εκτεταμένα υπολογιστικά πειράματα με χρήση προβλημάτων αναφοράς και μελετών διάφορων περιπτώσεων. Αξιολογείται η απόδοση του αλγορίθμου σε διαφορετικά χαρακτηριστικά και μεγέθη προβλημάτων, συγκρίνοντας τον με βασικές προσεγγίσεις. Το κεφάλαιο παρέχει πληροφορίες σχετικά με τη συμπεριφορά και την αποτελεσματικότητα του αλγορίθμου σε διάφορα σενάρια, τεκμηριώνοντας την πρακτική του χρησιμότητα.

Κεφάλαιο 5: Συμπεράσματα και μελλοντική έρευνα συνοψίζει τα βασικά ευρήματα και τις συνεισφορές της έρευνας, συζητούνται οι περιορισμοί της τρέχουσας προσέγγισης και προσδιορίζονται κατευθύνσεις για μελλοντική έρευνα. Τοποθετείται η έρευνα στο ευρύτερο πλαίσιο της βελτιστοποίησης δρομολόγησης οχημάτων και της διαχείρισης εφοδιαστικής, αναδεικνύοντας τις επιπτώσεις της τόσο στη θεωρία όσο και στην πράξη.

Τα παραρτήματα παρέχουν συμπληρωματικό υλικό, συμπεριλαμβανομένων λεπτομερών μαθηματικών αποδείξεων, ψευδοκώδικα αλγορίθμων, οδηγιών συντονισμού παραμέτρων και αναλυτικών πινάκων αποτελεσμάτων. Αυτές οι πρόσθετες πληροφορίες υποστηρίζουν το κύριο κείμενο, διατηρώντας παράλληλα τη ροή και την εστίαση της κύριας αφήγησης.

Σε όλη τη διατριβή, δίνεται έμφαση τόσο στη θεωρητική αυστηρότητα όσο και στην πρακτική συνάφεια, αντανακλώντας τη διπλή ακαδημαϊκή και εφαρμοσμένη φύση της έρευνας. Η δομή εξελίσσεται λογικά από τον ορισμό του προβλήματος και τη θεωρητική ανάπτυξη έως την εμπειρική επικύρωση και τις πρακτικές επιπτώσεις, παρέχοντας μια ολοκληρωμένη αντιμετώπιση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου και της προτεινόμενης προσέγγισης λύσης.

Συμπεράσματα

Η παρούσα εισαγωγή καθόρισε το πλαίσιο, τη σημασία, τους στόχους και τη μεθοδολογική προσέγγιση της έρευνας που παρουσιάζεται στην παρούσα διατριβή. Το Πρόβλημα Δρομολόγησης Οχημάτων Μικτού Στόλου Πολλαπλών Αποθηκών αποτελεί μια κρίσιμη πρόκληση βελτιστοποίησης στη σύγχρονη εφοδιαστική, συνδυάζοντας τη γεωγραφική πολυπλοκότητα των πολλαπλών αποθηκών με τη λειτουργική ποικιλομορφία των ετερογενών στόλων οχημάτων. Ο προτεινόμενος υβριδικός αλγόριθμος αντιμετωπίζει αυτή την πρόκληση μέσω μιας ολοκληρωμένης προσέγγισης που συνδυάζει τεχνικές ομαδοποίησης, άπληστες ευρετικές τεχνικές, βελτιστοποίηση αποικίας μυρμηγκιών και διαδικασίες τοπικής αναζήτησης.

Η έρευνα συμβάλλει σημαντικά τόσο στη θεωρητική κατανόηση των πολύπλοκων προβλημάτων δρομολόγησης οχημάτων όσο και στα πρακτικά εργαλεία που είναι διαθέσιμα για τη βελτιστοποίηση της εφοδιαστικής. Μέσω μιας ολοκληρωμένης θεωρητικής ανάλυσης και εμπειρικής αξιολόγησης, η διατριβή αποδεικνύει την αποτελεσματικότητα και την προσαρμοστικότητα της προτεινόμενης προσέγγισης σε διαφορετικά πλαίσια προβλημάτων.

Τα επόμενα κεφάλαια θα αναπτύξουν αυτά τα θέματα λεπτομερώς, παρέχοντας μια διεξοδική διερεύνηση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου και του υβριδικού αλγορίθμου που αναπτύχθηκε για την επίλυσή του. Η έρευνα αυτή αποσκοπεί τελικά να προωθήσει το πεδίο της βελτιστοποίησης της δρομολόγησης οχημάτων, παρέχοντας παράλληλα πρακτικές λύσεις στις προκλήσεις της εφοδιαστικής που αντιμετωπίζουν οι οργανισμοί σε διάφορους τομείς της οικονομίας.

ΚΕΦΑΛΑΙΟ 2 Βιβλιογραφική Επισκόπηση

Εισαγωγή

Το πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem - VRP) και οι παραλλαγές του έχουν μελετηθεί εκτενώς στην επιχειρησιακή έρευνα, την επιστήμη των υπολογιστών και τα logistics λόγω των πρακτικών τους εφαρμογών στις μεταφορές [22], τη διαχείριση της εφοδιαστικής αλυσίδας και τα αυτόνομα συστήματα. Το παρόν κεφάλαιο παρέχει μία ολοκληρωμένη ανασκόπηση της βιβλιογραφίας για το VRP και τις επεκτάσεις του, εστιάζοντας σε δύο βασικούς τομείς: το **Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών** (Multi-Depot Vehicle Routing Problem - MDVRP) και το **Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου** (Mixed Fleet Capacitated Multiple Traveling Salesman Problem (mfcmTSP). Αυτά τα προβλήματα σχετίζονται ιδιαίτερα με τα σύγχρονα logistics, όπου η ένωση ετερογενών στόλων (π.χ., φορτηγών, drone, μοτοσικλετών) και των επιχειρήσεων πολλαπλών αποθηκών γίνεται ολοένα και πιο σημαντική. Το παρόν κεφάλαιο είναι δομημένο για να συζητηθούν πρώτα οι θεμελιώδεις έννοιες του VRP, έπειτα μία λεπτομερής ανάλυση του MDVRP και του mfcmTSP, και ολοκληρώνεται με μία σύνθεση των προσεγγίσεων τελευταίας τεχνολογίας και των συνεπειών τους για την προτεινόμενη εργασία.

Βασικές Αρχές του Προβλήματος Δρομολόγησης Οχημάτων (VRP)

Κλασικό VRP

Το Πρόβλημα Δρομολόγησης Οχημάτων (VRP), που εισήχθη για πρώτη φορά από τους Dantzig και Ramser το 1959, είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης που επιδιώκει να προσδιορίσει το βέλτιστο σύνολο διαδρομών για έναν στόλο οχημάτων για την εξυπηρέτηση ενός συνόλου πελατών [5]. Το κλασικό VRP περιλαμβάνει μια ενιαία αποθήκη, έναν ομοιογενή στόλο οχημάτων και περιορισμούς όπως η χωρητικότητα του οχήματος και το μήκος διαδρομής. Ο στόχος είναι να ελαχιστοποιηθεί το συνολικό κόστος ταξιδιού, διασφαλίζοντας παράλληλα ότι ικανοποιούνται όλες οι απαιτήσεις των πελατών. Το VRP είναι NP-hard, πράγμα που σημαίνει ότι η εύρεση μιας ακριβούς λύσης για μεγάλες περιπτώσεις είναι υπολογιστικά ανέφικτη, καθιστώντας απαραίτητη τη χρήση ευρετικών και μεταευρετικών προσεγγίσεων.

Παραλλαγές του VRP

Με τα χρόνια, πολλές παραλλαγές του VRP έχουν προταθεί για την αντιμετώπιση συγκεκριμένων περιορισμών και σεναρίων του πραγματικού κόσμου. Μερικές από τις πιο αξιοσημείωτες παραλλαγές περιλαμβάνουν:

- Capacitated VRP (CVRP): Ενσωματώνει περιορισμούς χωρητικότητας του οχήματος [23].
- Time-WIndow VRP (VRPTW): Προσθέτει χρονικά παράθυρα για παραδόσεις πελατών [24].
- Dynamic VRP (DVRP): Λαμβάνει υπόψη τις δυναμικές αλλαγές στις απαιτήσεις των πελατών ή στις συνθήκες κυκλοφορίας.
- Multi-Depot VRP (MDVRP): Επεκτείνει το VRP σε πολλαπλές αποθήκες, όπου τα οχήματα ξεκινούν και τελειώνουν τις διαδρομές τους σε διαφορετικές τοποθεσίες [25].
- Mixed Fleet VRP (MFVRP): Περιλαμβάνει έναν ετερογενή στόλο οχημάτων με ποικίλες χωρητικότητες, ταχύτητες και κόστος [26].

Αυτές οι παραλλαγές έχουν εφαρμοστεί σε διάφορους τομείς, συμπεριλαμβανομένων των logistics, των δημόσιων συγκοινωνιών, και της παράδοσης με τη βοήθεια drone, υπογραμμίζοντας την ευελιξία και τη σημασία του VRP στην επίλυση προβλημάτων του πραγματικού κόσμου.

Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών

Ορισμός του Προβλήματος

Το πρόβλημα δρομολόγησης οχημάτων πολλαπλών αποθηκών (Multi-Depot Vehicle Routing Problem - MDVRP) είναι μια επέκταση του κλασικού VRP όπου τα οχήματα αποστέλλονται από πολλαπλές αποθήκες για να εξυπηρετήσουν ένα σύνολο πελατών [27]. Ο στόχος είναι η ελαχιστοποίηση της συνολικής απόστασης του ταξιδιού ή του κόστους, διασφαλίζοντας παράλληλα ότι κάθε πελάτης εξυπηρετείται ακριβώς μία φορά και ότι τηρούνται οι περιορισμοί χωρητικότητας του οχήματος. Το MDVRP είναι ιδιαίτερα σημαντικό σε σενάρια όπου οι επιχειρήσεις logistics είναι αποκεντρωμένες, όπως σε στρατιωτικά logistics, δίκτυα παράδοσης ηλεκτρονικού εμπορίου και συστήματα αστικών μεταφορών.

Προσεγγίσεις Επίλυσης

Ακριβείς Μέθοδοι: Οι Balakrishnan et al. (1987) διατύπωσαν το MDVRP ως πρόβλημα ακέραιου προγραμματισμού [28]. Οι Baldacci και Mingozzi (2009) ανέπτυξαν έναν exact algorithm βασισμένο σε σύνολα [29].

Μεταευρετικές Προσεγγίσεις: Οι πρόσφατες εξελίξεις στο MDVRP έχουν επικεντρωθεί στην ανάπτυξη αποτελεσματικών μεταευρετικών αλγορίθμων για την αντιμετώπιση της πολυπλοκότητας του προβλήματος. Μια αξιοσημείωτη προσέγγιση είναι η *Adaptive Ant Colony Optimization with Node Clustering (AACO-NC)* που προτάθηκε από τους Stodola και Nohel (2023) [2]. Αυτός ο αλγόριθμος εισάγει αρκετούς νέους μηχανισμούς, όπως:

- Ομαδοποίηση Κόμβων: Οργανώνει τους πελάτες σε συμπλέγματα με βάση την εγγύτητα και την τοπολογία γραφήματος, μειώνοντας τον χώρο αναζήτησης διατηρώντας παράλληλα την ποικιλία λύσεων.
- Προσαρμοστική Εξάτμιση Φερομόνης: Προσαρμόζει δυναμικά τον ρυθμό εξάτμισης φερομόνης με βάση την ποικιλομορφία του πληθυσμού, εξισορροπώντας την εξερεύνηση και την εκμετάλλευση.
- Τερματισμός με βάση την Εντροπία: Χρησιμοποιεί την εντροπία Shannon για τη μέτρηση της ποικιλομορφίας του πληθυσμού και τον τερματισμό του αλγορίθμου όταν δεν είναι πιθανές περαιτέρω βελτιώσεις.

Ο αλγόριθμος AACO-NC έχει επιδείξει ανώτερη απόδοση σε στιγμιότυπα αναφοράς MDVRP, ξεπερνώντας τις υπερσύγχρονες μεθόδους όσον αφορά την ποιότητα της λύσης και την υπολογιστική απόδοση. Η ικανότητά του να χειρίζεται περιπτώσεις μεγάλης κλίμακας (μέχρι 360 κόμβους) το καθιστά ιδιαίτερα κατάλληλο για εφαρμογές πραγματικού κόσμου.
Επίσης,οι Polacek et al. (2004) ανέπτυξαν την Variable NEighborhood Search για το MDVRP [20] και οι Thangiah και Salhi (2001) χρησιμοποίησαν γενετικούς αλγορίθμους [31].

Εφαρμογές του MDVRP

Το MDVRP έχει εφαρμοστεί σε διάφορους τομείς, όπως:

- Στρατιωτική Επιμελητεία: Βελτιστοποίηση της ανάπτυξης πόρων σε σενάρια πολλαπλών αποθηκών, όπως η παράδοση προμηθειών σε μπροστινές επιχειρησιακές βάσεις. [35]
- Παράδοση Ηλεκτρονικού Εμπορίου: Διαχείριση αποκεντρωμένων αποθηκών και στόλων παράδοσης για την ελαχιστοποίηση του χρόνου και του κόστους παράδοσης. [36]
- Δημόσιες Συγκοινωνίες: Σχεδιασμός αποτελεσματικών διαδρομών για λεωφορεία ή τραμ που λειτουργούν από πολλαπλά αμαξοστάσια. [37]
- Διανομή Φαρμάκων: Οργάνωση δρομολογίων από πολλαπλές αποθήκες φαρμάκων προς νοσοκομεία και φαρμακεία. [38]

Αυτές οι εφαρμογές υπογραμμίζουν τη σημασία του MDVRP για την αντιμετώπιση πολύπλοκων

προκλήσεων στα logistics.

Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου

Ορισμός προβλήματος

Το Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου (Mixed Fleet Capacitated Multiple TSP - mfcmTSP) είναι μια παραλλαγή του κλασικού TSP που περιλαμβάνει έναν ετερογενή στόλο οχημάτων (π.χ. φορτηγά, μοτοσυκλέτες και drones) με ποικίλες χωρητικότητες και ταχύτητες [32]. Ο στόχος είναι να ελαχιστοποιηθεί η μέγιστη διάρκεια ολοκλήρωσης (δηλαδή ο χρόνος που χρειάζεται το πιο αργό όχημα) διασφαλίζοντας παράλληλα ότι όλοι οι πελάτες εξυπηρετούνται. Αυτό το πρόβλημα είναι ιδιαίτερα σημαντικό σε σύγχρονα συστήματα logistic όπου διαφορετικοί τύποι οχημάτων (φορτηγά, μοτοσυκλέτες, drones) συνεργάζονται.

Πρόσφατες Εξελίξεις

Μια πρόσφατη μελέτη των Oikonomou et al. (2021) [3] πρότεινε έναν γρήγορο ευρετικό αλγόριθμο για το mfcmTSP που αξιοποιεί μια προσέγγιση δύο φάσεων:

- Φάση 1: Επιλύει ένα μόνο πρόβλημα Πλανόδιου Πωλητή για το κύριο όχημα (συνήθως φορτηγό) χρησιμοποιώντας την ευρετική προσέγγιση του πλησιέστερου γείτονα (Nearest Neighbor - NN).
- Φάση 2: Επαναληπτική μεταφορά πελατών σε γρηγορότερα οχήματα, δηλαδή από τη διαδρομή του φορτηγού προς τη μοτοσικλέτα ή το drone, δίνοντας προτεραιότητα στο όχημα με το μεγαλύτερο κενό ολοκλήρωσης σε σχέση με το φορτηγό.

Αυτός ο ευρετικός αλγόριθμος έχει επιδείξει σημαντικές βελτιώσεις στην ολοκλήρωση (έως και 60%) σε σύγκριση με τις λύσεις ενός μοναδικού οχήματος, καθιστώντας το μια πολλά υποσχόμενη προσέγγιση για δρομολόγηση σε πραγματικό χρόνο σε σενάρια αστικών ταχυμεταφορών.

Εφαρμογές του mfcmTSP

Το Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου έχει εφαρμοστεί σε:

- Παράδοση με τη βοήθεια Drone: Βελτιστοποίηση του συντονισμού drones και φορτηγών για last-mile παράδοση σε αστικές περιοχές.
- Παιχνίδια βάσει τοποθεσίας: Σχεδιασμός βέλτιστων διαδρομών για παίκτες σε παιχνίδια όπως το Pokémon GO, όπου οι παίκτες πρέπει να επισκεφτούν πολλές τοποθεσίες.
- Αντιμετώπιση Έκτακτων Αναγκών: Συντονισμός ετερογενών στόλων για ταχεία ανάπτυξη πόρων σε σενάρια καταστροφών.

Αυτές οι εφαρμογές υπογραμμίζουν την ευελιξία του προβλήματος Πολλαπλών Πλανόδιων

Πωλητών με χωρητικότητα μικτού στόλου στην αντιμετώπιση των σύγχρονων προκλήσεων των logistic.

Σύνθεση και Ερευνητικά Κενά

Συγκριτική Ανάλυση

Ενώ τόσο το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκευτικών Χώρων όσο και το Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου αντιμετωπίζουν κρίσιμες πτυχές των σύγχρονων logistics, διαφέρουν ως προς τις διατυπώσεις προβλημάτων και τις προσεγγίσεις λύσεων. Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκευτικών Χώρων εστιάζει σε λειτουργίες πολλαπλών αποθηκών με ομοιογενείς στόλους, ενώ το Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου ασχολείται με λειτουργίες μίας αποθήκης που περιλαμβάνουν ετερογενείς στόλους. Παρά αυτές τις διαφορές, και τα δύο προβλήματα μοιράζονται κοινές προκλήσεις, όπως η εξισορρόπηση της εξερεύνησης και της εκμετάλλευσης σε χώρους αναζήτησης λύσεων και ο αποτελεσματικός χειρισμός περιπτώσεων μεγάλης κλίμακας.

Συνδυασμός MDVRP και mfcmTSP

Η ενσωμάτωση σεναρίων πολλαπλών αποθηκών και μικτού στόλου παραμένει σχετικά ανεξερεύνητη στη βιβλιογραφία. Οι λίγες υπάρχουσες εργασίες περιλαμβάνουν

- Οι Bettinelli et al. (2011) μελέτησαν το MDVRP με ετερογενή στόλο χρησιμοποιώντας branch-and-cut-and-price [33].
- Οι Belfiore και Yoshizaki (2009) εξέτασαν scatter search για παρόμοια προβλήματα [34].

Ερευνητικά Κενά

Η βιβλιογραφική ανασκόπηση αποκαλύπτει αρκετά ερευνητικά κενά:

- Περιορισμένη Ενσωμάτωση: Οι περισσότερες υπάρχουσες προσεγγίσεις εστιάζουν είτε σε προβλήματα πολλαπλών αποθηκών είτε σε προβλήματα μεικτού στόλου, αλλά όχι και στα δύο.
- Χρονικά Παράθυρα: Λίγες εργασίες αντιμετωπίζουν συνδυαστικά MDMFVRP με χρονικά παράθυρα.
- Υβριδικές Προσεγγίσεις: Έλλειψη αλγορίθμων που συνδυάζουν πολλαπλές τεχνικές βελτιστοποίησης.

Επιπτώσεις για την Παρούσα Πτυχιακή

Η παρούσα πτυχιακή εργασία στοχεύει να αντιμετωπίσει αυτά τα κενά αναπτύσσοντας έναν υβριδικό αλγόριθμο που συνδυάζει τα πλεονεκτήματα των ευρετικών AACO-NC και mfcmTSP. Συγκεκριμένα, ο αλγόριθμος θα:

- Ενσωματώνει σενάρια πολλαπλών αποθηκών και μικτού στόλου για την διαχείριση πολύπλοκων λειτουργιών των logistics.

- Ενσωματώνει προσαρμοστικούς μηχανισμούς (π.χ. τερματισμό με βάση την εντροπία) για να εξισορροπείται η εξερεύνηση και η εκμετάλλευση.
- Αξιοποιεί τις αρχές εξισορρόπησης φόρτου εργασίας για να βελτιστοποιηθούν οι αναθέσεις οχημάτων σε πραγματικό χρόνο.

Συμπέρασμα

Αυτό το κεφάλαιο παρέχει μια περιεκτική ανασκόπηση της βιβλιογραφίας σχετικά με το Πρόβλημα Δρομολόγησης Οχημάτων, το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκευτικών Χώρων και το Πρόβλημα Πολλαπλών Πλανόδιων Πωλητών με χωρητικότητα μικτού στόλου, επισημαίνοντας τις θεωρητικές βάσεις, τις προσεγγίσεις τελευταίας τεχνολογίας και τις πρακτικές εφαρμογές τους. Η σύνθεση αυτών των εργασιών εντόπισε βασικά ερευνητικά κενά και παρακίνησε την ανάπτυξη ενός νέου υβριδικού αλγορίθμου για πολύπλοκα σενάρια εφοδιαστικής. Στο επόμενο κεφάλαιο θα παρουσιαστεί η προτεινόμενη μεθοδολογία, περιγράφοντας λεπτομερώς τον σχεδιασμό, την υλοποίηση και το πλαίσιο αξιολόγησης του αλγορίθμου.

ΚΕΦΑΛΑΙΟ 3 Υλοποίηση Υβριδικού Αλγορίθμου

Διατύπωση Προβλήματος

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου (Multi-Depot Mixed Fleet Vehicle Routing Problem - MDMFVRP) αποτελεί μια σημαντική επέκταση του κλασικού Προβλήματος Δρομολόγησης Οχημάτων (Vehicle Routing Problem - VRP), εισάγοντας δύο βασικές πολυπλοκότητες που αντικατοπτρίζουν καλύτερα τα σενάρια εφοδιαστικής του πραγματικού κόσμου: πολλαπλές αποθήκες και ετερογενείς στόλους οχημάτων. Αυτός ο συνδυασμός καθιστά το πρόβλημα ιδιαίτερα δύσκολο αλλά και εξαιρετικά σημαντικό για τις σύγχρονες επιχειρήσεις εφοδιαστικής.

Ορισμός Προβλήματος

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου μπορεί να οριστεί επίσημα ως εξής:

Δεδομένου ενός συνόλου γεωγραφικά διασκορπισμένων αποθηκών, καθεμία από τις οποίες στεγάζει έναν ετερογενή στόλο οχημάτων με διαφορετική χωρητικότητα και χαρακτηριστικά απόδοσης, και ενός συνόλου πελατών με γνωστές απαιτήσεις, ο στόχος είναι να προσδιοριστεί η βέλτιστη ανάθεση των πελατών στις αποθήκες και οι βέλτιστες διαδρομές για κάθε όχημα, έτσι ώστε:

1. Κάθε πελάτης εξυπηρετείται ακριβώς μία φορά
2. Να τηρούνται όλοι οι περιορισμοί χωρητικότητας των οχημάτων
3. Κάθε διαδρομή ξεκινά και καταλήγει στην ίδια αποθήκη
4. Το συνολικό λειτουργικό κόστος ελαχιστοποιείται

Το πρόβλημα αποτυπώνει την ουσία σύνθετων σεναρίων εφοδιαστικής όπου οι εταιρείες λειτουργούν πολλαπλά κέντρα διανομής με διαφορετικούς τύπους οχημάτων παράδοσης, από φορτηγά μεγάλης χωρητικότητας έως ευέλικτες μοτοσικλέτες και ακόμη και μη επανδρωμένα αεροσκάφη (drone) για εξειδικευμένες παραδόσεις.

Μαθηματικοί Συμβολισμοί

Για να τυποποιήσουμε το πρόβλημα, καθορίζουμε τον παρακάτω συμβολισμό:

- $D = \{1, 2, \ldots, m\}$: Σύνολο αποθηκών
- $V_d = \{1, 2, \ldots, n_d\}$: Σύνολο διαθέσιμων οχημάτων στην αποθήκη $d \in D$
- $C = \{1, 2, \ldots, n\}$: Σύνολο πελατών
- $l_d = (x_d, y_d)$: Συντεταγμένες θέσης της αποθήκης $d \in D$
- $l_c = (x_c, y_c)$: Συντεταγμένες θέσης του πελάτη $c \in C$
- q_c: Ζήτηση του πελάτη $c \in C$
- Q_{dv}: Χωρητικότητα του οχήματος $v \in V_d$ στην αποθήκη $d \in D$
- S_{dv}: Ταχύτητα του οχήματος $v \in V_d$ στην αποθήκη $d \in D$
- $[a_c, b_c]$: Χρονικό παράθυρο για τον πελάτη $c \in C$, όπου a_c είναι ο νωρίτερος χρόνος εξυπηρέτησης και b_c είναι ο αργότερος χρόνος εξυπηρέτησης.
- t_{ij}: Χρόνος ταξιδιού μεταξύ των τοποθεσιών i και j
- δ_{ij}: Απόσταση μεταξύ των θέσεων i και j

Επιπλέον, ορίζουμε τις μεταβλητές απόφασης:

- x_{ijdv}: Δυαδική μεταβλητή απόφασης ίση με 1 εάν το όχημα v από την αποθήκη d ταξιδεύει απευθείας από τον κόμβο i στον κόμβο j, και 0 διαφορετικά.
- y_{cdv}: Δυαδική μεταβλητή απόφασης ίση με 1 εάν ο πελάτης c εξυπηρετείται από το όχημα v από την αποθήκη d, και 0 διαφορετικά.
- T_c: Ώρα έναρξης εξυπηρέτησης στον πελάτη c

Περιορισμοί και Αντικειμενική Συνάρτηση

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου υπόκειται στους ακόλουθους περιορισμούς:

1. **Περιορισμός ανάθεσης πελατών:** Κάθε πελάτης πρέπει να αντιστοιχίζεται σε ακριβώς ένα όχημα από μία αποθήκη: $\Sigma_{d\in D}\Sigma_{v\in V_d} y_{cvd} = 1,\ \forall c \in C$
2. **Περιορισμός χωρητικότητας οχημάτων:** Η συνολική ζήτηση των πελατών που αντιστοιχούν σε ένα όχημα δεν πρέπει να υπερβαίνει τη χωρητικότητά του: $\Sigma_{c\in C} q_c \cdot y_{cvd} \le Q_{dv}, \forall d \in D,\ \forall v \in V_d$
3. **Περιορισμός διατήρησης της ροής:** Κάθε όχημα που εισέρχεται σε έναν κόμβο πελάτη πρέπει επίσης να εξέρχεται από αυτόν τον κόμβο: $\Sigma_{i\in C\cup\{d\}} x_{icdv} = \Sigma_{j\in C\cup\{d\}} x_{cjdv} = y_{cdv}, \forall c \in C, \forall d \in D, \forall v \in V_d$
4. **Περιορισμός αρχής και τέλους αποθήκης:** Κάθε διαδρομή οχήματος πρέπει να αρχίζει και να τελειώνει στην αποθήκη που της έχει ανατεθεί: $\Sigma_{j\in C} x_{djdv} = \Sigma_{i\in C} x_{iddv} \le 1, \forall d \in D, \forall v \in V_d$
5. **Περιορισμοί χρονικού παραθύρου:** Η εξυπηρέτηση σε κάθε πελάτη πρέπει να αρχίσει εντός του καθορισμένου χρονικού παραθύρου: $a_c \le T_c \le b_c,\ \forall c \in C_{a_c}$
6. **Περιορισμοί χρονικής συνέπειας:** Εάν ένα όχημα ταξιδεύει από τον πελάτη i στον πελάτη j , ο χρόνος έναρξης εξυπηρέτησης στο j πρέπει να είναι τουλάχιστον ίσος με τον χρόνο έναρξης εξυπηρέτησης στο i συν τον χρόνο εξυπηρέτησης στον i συν τον χρόνο ταξιδιού από τον i στο j: $T_i + t_{ij} \le T_j,$ αν $x_{ijdv} = 1, \forall i,j \in C, \forall d \in D, \forall v \in V_d$

Η αντικειμενική συνάρτηση του MDMFVRP είναι:

$$Z = \Sigma_{d\in D}\Sigma_{v\in V_d}\Sigma_{i\in C\cup\{d\}}\Sigma_{j\in C\cup\{d\}} c_{ijdv} \cdot x_{ijdv} + \Sigma_{d\in D}\Sigma_{v\in V_d} f_{dv} \cdot u_{dv} + P$$

όπου:

- c_{ijdv}: Κόστος ταξιδιού από i σε j με όχημα v της αποθήκης d
- f_{dv}: Σταθερό κόστος χρήσης οχήματος v της αποθήκης d
- u_{dv}: Δυαδική μεταβλητή, 1 εάν το όχημα v της αποθήκης d χρησιμοποιείται
- P: Ποινές για παραβίαση χρονικών παραθύρων

Το κόστος ταξιδιού υπολογίζεται ως:

$$c_{ijdv} = (d_{ij}/S_{dv}) + w_c$$

όπου d_{ij} είναι η απόσταση μεταξύ i και j, και w_c είναι ο χρόνος αναμονής.

Υπολογιστική Πολυπλοκότητα

Το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου είναι ένα NP-δύσκολο πρόβλημα, καθώς γενικεύει το κλασικό Πρόβλημα Δρομολόγησης Οχημάτων, το οποίο είναι επίσης NP-δύσκολο. Η προσθήκη πολλαπλών αποθηκών και ετερογενών στόλων αυξάνει

σημαντικά την πολυπλοκότητα του προβλήματος. Για ένα πρόβλημα με m αποθήκες, κατά μέσο όρο k οχήματα ανά αποθήκη και n πελάτες, ο αριθμός των πιθανών αναθέσεων πελατών σε αποθήκες είναι της τάξης των m^n. Επιπλέον, για κάθε ανάθεση, η εύρεση βέλτιστων δρομολογίων περιλαμβάνει την επίλυση πολλαπλών περιπτώσεων Προβλημάτων Πλανόδιου Πωλητή, κάθε μία από τις οποίες είναι NP-δύσκολη.

Δεδομένης αυτής της υπολογιστικής πολυπλοκότητας, οι ακριβείς μέθοδοι επίλυσης καθίστανται ανέφικτες για περιπτώσεις του πραγματικού κόσμου, οι οποίες συνήθως περιλαμβάνουν δεκάδες αποθήκες, εκατοντάδες οχήματα και χιλιάδες πελάτες. Επομένως, είναι απαραίτητες ευρετικές και μετά-ευρετικές προσεγγίσεις για την εύρεση λύσεων υψηλής ποιότητας σε εύλογο υπολογιστικό χρόνο.

Σχεδιασμός Αλγορίθμου

Για να αντιμετωπίσουμε τη δύσκολη φύση του Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, προτείνουμε έναν νέο υβριδικό αλγόριθμο που συνδυάζει τεχνικές ομαδοποίησης, άπληστες ευρετικές τεχνικές και μετα-ευρετική βελτιστοποίηση. Ο αλγόριθμος αναλύει το πολύπλοκο πρόβλημα σε διαχειρίσιμα υποπροβλήματα και εφαρμόζει εξειδικευμένες τεχνικές σε κάθε στάδιο.

Γενική Δομή

Ο υβριδικός μας αλγόριθμος ακολουθεί μια ιεραρχική προσέγγιση με έξι κύρια στοιχεία:

1. **Ανάθεση από πελάτη σε αποθήκη** με χρήση ομαδοποίησης K-μέσων
2. **Ανάθεση οχημάτων σε κάθε αποθήκη** με άπληστη προσέγγιση
3. **Βελτιστοποίηση διαδρομής** με χρήση βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization - ACO)
4. **Βελτίωση τοπικής αναζήτησης** με χρήση ανταλλαγών 2-opt
5. **Εξισορρόπηση του φόρτου εργασίας** μεταξύ των αποθηκών
6. **Επαλήθευση εφικτότητας χρονικού παραθύρου**

Αυτό το πλαίσιο αντιμετωπίζει τη λήψη αποφάσεων πολλαπλών επιπέδων που απαιτούνται στο Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου: πρώτα προσδιορίζεται ποια αποθήκη εξυπηρετεί κάθε πελάτη, στη συνέχεια ποιος τύπος οχήματος είναι ο καταλληλότερος και, τέλος, βελτιστοποιείται η λεπτομερής ακολουθία δρομολόγησης.

Ανάθεση από Πελάτη σε Αποθήκη

Η πρώτη πρόκληση για την επίλυση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου είναι ο καθορισμός της αποθήκης που θα πρέπει να εξυπηρετεί κάθε πελάτη. Χρησιμοποιούμε την ομαδοποίηση K-μέσων, έναν καθιερωμένο αλγόριθμο μη επιβλεπόμενης μάθησης, για να ομαδοποιήσουμε τους πελάτες με βάση τη γεωγραφική τους εγγύτητα στις αποθήκες.

Ο αλγόριθμος K-μέσων λειτουργεί ως εξής:

1. Αρχικοποίηση των κέντρων συστάδων στις τοποθεσίες των αποθηκών
2. Ανάθεση κάθε πελάτη στην πλησιέστερη αποθήκη (κέντρο συστάδας)
3. Υπολογισμός του γεωμετρικού κεντροειδούς κάθε συστάδας
4. Επανάληψη των βημάτων 2 και 3 μέχρι σύγκλισης

Σε αντίθεση με τις παραδοσιακές υλοποιήσεις του αλγορίθμου K-μέσων, καθορίζουμε τα αρχικά κέντρα συστάδων σε θέσεις αποθηκών αντί για τυχαία αρχικοποίηση. Αυτός ο περιορισμός διασφαλίζει ότι οι συστάδες που προκύπτουν είναι κεντραρισμένες γύρω από τις πραγματικές θέσεις αποθηκών. Ο αλγόριθμος αναθέτει τους πελάτες στις αποθήκες με βάση την ευκλείδεια απόσταση, η οποία χρησιμεύει ως ένα λογικό υποκατάστατο για το κόστος ταξιδιού στις περισσότερες γεωγραφικές ρυθμίσεις.

Η μαθηματική διατύπωση για αυτή την ανάθεση είναι:

Ανάθεση του πελάτη c στην αποθήκη d* όπου $d^* = arg\,min_{d \in D} ||l_c - l_d||_2$, όπου το $||l_c - l_d||_2$ αντιπροσωπεύει την ευκλείδεια απόσταση μεταξύ του πελάτη c και της αποθήκης d. Αυτή η προσέγγιση παρέχει μια καλή αρχική κατανομή των πελατών στις αποθήκες, η οποία χρησιμεύει ως βάση για τα επόμενα βήματα βελτιστοποίησης.

Στρατηγική Ανάθεσης Οχημάτων

Αφού οι πελάτες ανατεθούν στις αποθήκες, πρέπει να καθορίσουμε ποιο όχημα από τον ετερογενή στόλο κάθε αποθήκης θα πρέπει να εξυπηρετεί κάθε πελάτη. Ο αλγόριθμός μας χρησιμοποιεί μια άπληστη στρατηγική ανάθεσης με προτεραιότητες χωρητικότητας.

Οι βασικές αρχές αυτής της ανάθεσης είναι οι εξής:

1. Ταξινόμηση οχημάτων κατα φθίνουσα χωρητικότητα για ανάθεση (φορτηγά πριν από μοτοσυκλέτες πριν από μη επανδρωμένα αεροσκάφη)
2. Για κάθε όχημα, ανάθεση πλησιέστερων εφικτών πελατών, τηρώντας παράλληλα τους περιορισμούς χωρητικότητας
3. Επανάληψη μέχρι να ανατεθούν όλοι οι πελάτες ή να μην υπάρχουν άλλα διαθέσιμα οχήματα.

Η προσέγγιση αυτή αναγνωρίζει ότι τα μεγαλύτερα οχήματα έχουν γενικά χαμηλότερο κόστος ανά μονάδα χωρητικότητας, γεγονός που τα καθιστά πιο αποδοτικά για την εξυπηρέτηση πολλών πελατών. Η διαδικασία ανάθεσης τυποποιείται ως εξής:

Για κάθε αποθήκη $d \in D$:

1. Ταξινόμηση των οχημάτων στο V_d κατά φθίνουσα χωρητικότητα Q_{dv}.
2. Για κάθε όχημα $v \in V_d$ (από τη μεγαλύτερη προς τη μικρότερη χωρητικότητα):

   Όσο το όχημα v έχει εναπομένουσα χωρητικότητα:
   - i. Βρείτε τον πλησιέστερο πελάτη c που δεν έχει ανατεθεί από την αποθήκη d.
   - ii. Εάν η ζήτηση του c χωράει στην εναπομένουσα χωρητικότητα, αναθέστε τον c στο v.
   - iii. Διαφορετικά, μετακινηθείτε στο επόμενο όχημα

Αυτή η άπληστη προσέγγιση αναθέτει αποτελεσματικά τους πελάτες στα κατάλληλα οχήματα, διατηρώντας παράλληλα τη εφικτότητα σε σχέση με τους περιορισμούς χωρητικότητας.

Βελτιστοποίηση Διαδρομής με Βελτιστοποίηση Αποικίας Μυρμηγκιών

Με την ανάθεση πελατών σε συγκεκριμένα οχήματα, η επόμενη πρόκληση είναι ο καθορισμός της βέλτιστης σειράς με την οποία κάθε όχημα θα πρέπει να επισκεφθεί τους πελάτες που του έχουν ανατεθεί. Χρησιμοποιούμε τη βελτιστοποίηση αποικίας μυρμηγκιών (Ant Colony Optimizatino -

ACO), μια μεταευρετική μέθοδο εμπνευσμένη από τη συμπεριφορά αναζήτησης αποικιών μυρμηγκιών, για να βρούμε σχεδόν βέλτιστες λύσεις δρομολόγησης.

Η Βελτιστοποίηση Αποικίας Μυρμηγκιών λειτουργεί με την προσομοίωση μιας αποικίας τεχνητών μυρμηγκιών που κατασκευάζουν λύσεις ενώ εναποθέτουν φερομόνες σε υποσχόμενες διαδρομές. Τα μονοπάτια φερομόνης εξελίσσονται με την πάροδο του χρόνου, με ισχυρότερα επίπεδα φερομόνης σε στοιχεία λύσεων υψηλής ποιότητας.

Η υλοποίηση μας για την Βελτιστοποίηση Αποικίας Μυρμηγκιών για το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου λειτουργεί ως εξής:

1. **Αρχικοποίηση**:
    - Για κάθε όχημα με αντιστοιχισμένους πελάτες, δημιουργούμε έναν πίνακα φερομόνης τ με διαστάσεις $(n + 1) \times (n + 1)$, όπου n είναι ο αριθμός των πελατών που αντιστοιχούν στο όχημα (το +1 αντιπροσωπεύει την αποθήκη).
    - Αρχικοποιούμε όλες τις καταχωρήσεις του πίνακα φερομόνης με μια μικρή σταθερή τιμή τ_0.
2. **Κατασκευή λύσης**:
    - Για κάθε μυρμήγκι στην αποικία:
        - Ξεκινάμε από την αποθήκη
        - Επανειλημμένα επιλέγουμε τον επόμενο πελάτη που θα επισκεφθεί με βάση μια πιθανότητα που εξαρτάται τόσο από τα επίπεδα φερομόνης όσο και από ευρετικές πληροφορίες (απόσταση και χρονικό παράθυρο).
        - Ολοκληρώνουμε την διαδρομή με την επιστροφή στην αποθήκη.
3. **Ενημέρωση φερομόνης**:
    - Ένα κλάσμα της φερομόνης εξατμίζεται από όλες τις ακμές: $\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij}$
    - Εναποθέτουμε νέα φερομόνη στις ακμές που χρησιμοποιούνται από το καλύτερο μυρμήγκι ή από τα καλύτερα μυρμήγκια: $\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}$, όπου το $\Delta\tau_{ij}$ είναι ανάλογο της ποιότητας της λύσης
4. **Τερματισμός**:
    - Επαναλαμβάνουμε τα βήματα 2-3 για καθορισμένο αριθμό επαναλήψεων ή μέχρι επαρκούς σύγκλισης.

Η πιθανότητα ένα μυρμήγκι να επιλέξει να μετακινηθεί από τον πελάτη i στον πελάτη j δίνεται από:

$$p_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\Sigma_{k \in N_i}(\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}}$$

όπου:

- τ_{ij} είναι το επίπεδο φερομόνης στην ακμή (i, j)
- η_{ij} είναι η ευρετική πληροφορία, συνήθως $1/d_{ij}$ όπου d_{ij} είναι η απόσταση
- α και β είναι παράμετροι που ελέγχουν τη σχετική επιρροή της φερομόνης έναντι της ευρετικής πληροφορίας
- N_i είναι το σύνολο των εφικτών επόμενων πελατών

Για προβλήματα με χρονικά παράθυρα, ενισχύουμε την ευρετική πληροφορία ώστε να περιλαμβάνει τον επείγοντα χαρακτήρα του χρονικού παραθύρου:

$$\eta_{ij} = \frac{1}{d_{ij}} \cdot \frac{1}{max(1, b_j - (T_i + t_{ij}))}$$

Αυτή η τροποποιημένη ευρετική δίνει μεγαλύτερη προτεραιότητα στους πελάτες των οποίων τα χρονικά παράθυρα κλείνουν σύντομα, εξισορροπώντας τη χωρική εγγύτητα με τους χρονικούς περιορισμούς.

Βελτίωση Τοπικής Αναζήτησης

Για να βελτιώσουμε περαιτέρω τις διαδρομές που παράγονται από την Βελτιστοποίηση Αποικίας Μυρμηγκιών, εφαρμόζουμε μια διαδικασία τοπικής αναζήτησης 2 επιλογών. Ο αλγόριθμος 2-opt είναι μια απλή αλλά αποτελεσματική τεχνική για τη βελτίωση διαδρομών τύπου "Προβλήματος Πλανόδιου Πωλητή" με την επίλυση διασταυρωμένων ακμών.

Η λειτουργία του 2-opt αλγορίθμου:

1. Αφαιρούμε δύο μη γειτονικές ακμές από μια διαδρομή.
2. Επανασυνδέουμε τα κομμάτια που προέκυψαν με διαφορετικό τρόπο
3. Αποδεχόμαστε την αλλαγή εάν βελτιώνει το κόστος της διαδρομής

Για κάθε διαδρομή οχήματος, ο αλγόριθμός μας εξετάζει συστηματικά όλες τις πιθανές κινήσεις 2-opt και εφαρμόζει εκείνες που μειώνουν το συνολικό κόστος της διαδρομής. Η διαδικασία αυτή συνεχίζεται επαναληπτικά μέχρι να μην βρεθούν περαιτέρω βελτιώσεις ή να επιτευχθεί ένας μέγιστος αριθμός επαναλήψεων.

Η τοπική αναζήτηση 2-opt είναι ιδιαίτερα αποτελεσματική στην εξάλειψη των διασταυρώσεων της διαδρομής, οι οποίες είναι σχεδόν πάντα υποβέλτιστες στον ευκλείδειο χώρο. Αυτό το συμπλήρωμα της μεταευρετικής Βελτιστοποίησης Αποικίας Μυρμηγκιών βοηθά τον αλγόριθμο να ξεφύγει από τα τοπικά βέλτιστα και να βρει λύσεις υψηλότερης ποιότητας.

Υλοποίηση Χρονικού Παραθύρου

Τα χρονικά παράθυρα προσθέτουν μια κρίσιμη χρονική διάσταση στο πρόβλημα δρομολόγησης, αντικατοπτρίζοντας τους περιορισμούς του πραγματικού κόσμου, όπου οι πελάτες μπορεί να είναι διαθέσιμοι για εξυπηρέτηση μόνο κατά τη διάρκεια συγκεκριμένων χρονικών περιόδων. Ο αλγόριθμός μας χειρίζεται τα χρονικά παράθυρα με τρεις βασικούς τρόπους:

1. **Έλεγχος σκοπιμότητας**: Κατά τη διάρκεια της κατασκευής της διαδρομής, επαληθεύουμε ότι κάθε πελάτης μπορεί να εξυπηρετηθεί εντός του χρονικού του παραθύρου.
2. **Κατασκευή διαδρομής**: Ο αλγόριθμος Βελτιστοποίησης Αποικίας Μυρμηγκιών ενσωματώνει τους περιορισμούς των χρονικών παραθύρων κατά την κατασκευή των δρομολογίων.
3. **Υπολογισμός κόστους**: Η αντικειμενική συνάρτηση περιλαμβάνει ποινές για καθυστερημένες αφίξεις

Για κάθε δυνητική προσθήκη πελάτη σε μια διαδρομή, υπολογίζουμε τον νωρίτερο δυνατό χρόνο άφιξης με βάση:

- Την τρέχουσα ώρα στον τελευταίο πελάτη που επισκέφθηκε
- Τον χρόνο εξυπηρέτησης στον τελευταίο πελάτη
- Τον χρόνο ταξιδιού στον νέο πελάτη

Εάν ο υπολογιζόμενος χρόνος άφιξης υπερβαίνει τον τελευταίο χρόνο εξυπηρέτησης του πελάτη (b_c), η προσθήκη θεωρείται ανέφικτη. Εάν ο χρόνος άφιξης είναι προγενέστερος από τον πρωιμότερο χρόνο εξυπηρέτησης (a_c), το όχημα πρέπει να περιμένει, γεγονός που λαμβάνεται υπόψη στον υπολογισμό του κόστους, αλλά δεν παραβιάζει τη σκοπιμότητα.

Η βελτιωμένη συνάρτηση κόστους μας για τα χρονικά παράθυρα είναι:

$$\text{κόστος} = \text{χρόνος ταξιδιού} + \text{χρόνος αναμονής} + \text{ποινή καθυστέρησης}$$

όπου:

- Ο χρόνος ταξιδιού είναι το άθροισμα όλων των χρόνων ταξιδιού μεταξύ διαδοχικών στάσεων.

- Ο χρόνος αναμονής προκύπτει όταν φτάνουμε πριν ανοίξει το χρονικό παράθυρο του πελάτη.
- Η ποινή καθυστέρησης είναι ένας σημαντικός πολλαπλασιαστής κόστους που εφαρμόζεται όταν φτάνουμε μετά το κλείσιμο του χρονικού παραθύρου ενός πελάτη

Αυτός ο ολοκληρωμένος χειρισμός των χρονικών παραθύρων διασφαλίζει ότι ο αλγόριθμός μας παράγει λύσεις που είναι εφικτές και αποτελεσματικές σε πραγματικές καταστάσεις logistics με χρονικούς περιορισμούς.

Εξισορρόπηση Φόρτου Εργασίας

Για να διασφαλιστεί η αποτελεσματική χρήση όλων των αποθηκών, ο αλγόριθμός μας περιλαμβάνει μια φάση εξισορρόπησης του φόρτου εργασίας που ανακατανέμει τους πελάτες μεταξύ των αποθηκών όταν εντοπίζονται σημαντικές ανισορροπίες.

Η διαδικασία εξισορρόπησης του φόρτου εργασίας λειτουργεί ως εξής:

1. Υπολογίζουμε τον μέσο αριθμό πελατών που εξυπηρετούνται ανά αποθήκη
2. Εντοπίζουμε τις αποθήκες των οποίων ο φόρτος εργασίας υπερβαίνει το μέσο όρο κατά ένα ποσοστό κατωφλίου (π.χ. 20%)
3. Για τις υπερφορτωμένες αποθήκες, προσδιορίζουμε τους περιφερειακούς πελάτες (αυτούς που βρίσκονται πιο μακριά από την αποθήκη)
4. Αναθέτουμε τους πελάτες αυτούς σε κοντινές αποθήκες με χαμηλή χρήση.

Αυτό το βήμα εξισορρόπησης συμβάλλει στην αποφυγή καταστάσεων στις οποίες ορισμένες αποθήκες είναι υπερφορτωμένες ενώ άλλες έχουν πλεονάζουσα χωρητικότητα, οδηγώντας σε συνολική αναποτελεσματικότητα του συστήματος. Το μαθηματικό κριτήριο για την ανακατανομή είναι:

Αναδιανομή εάν: $|C_d| > (1 + \delta) \cdot \frac{\sum_{d' \in D} |C_{d'}|}{|D|}$

όπου $|C_d|$ είναι ο αριθμός των πελατών που αντιστοιχούν στην αποθήκη d και δ είναι το όριο ανοχής (συνήθως 0,2).

Ενσωμάτωση Επιμέρους Στοιχείων

Η αξία του υβριδικού αλγορίθμου μας έγκειται στον τρόπο με τον οποίο αυτά τα διαφορετικά στοιχεία συνεργάζονται για την αντιμετώπιση διαφορετικών πτυχών του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου. Ο αλγόριθμος ρέει ως εξής:

1. Η αρχική ομαδοποίηση αναθέτει τους πελάτες σε γεωγραφικά κατάλληλες αποθήκες.
2. Η ανάθεση με βάση τη χωρητικότητα διασφαλίζει την αποτελεσματική χρήση των οχημάτων
3. Η Βελτιστοποίηση Αποικίας Μυρμηγκιών βρίσκει διαδρομές υψηλής ποιότητας για κάθε όχημα
4. Η τοπική αναζήτηση βελτιώνει αυτές τις διαδρομές για να εξαλείψει τις προφανείς ανεπάρκειες
5. Η εξισορρόπηση του φόρτου εργασίας εξασφαλίζει την ομοιόμορφη κατανομή των παραδόσεων στις αποθήκες
6. Η επαλήθευση του χρονικού παραθύρου επιβεβαιώνει τη χρονική εφικτότητα της λύσης

Αυτή η ολοκληρωμένη προσέγγιση επιτρέπει σε κάθε στοιχείο να εστιάζει στα ιδιαίτερα δυνατά του σημεία: η ομαδοποίηση K-μέσων για τη χωρική ομαδοποίηση, η άπληστη ανάθεση για τη χρήση της χωρητικότητας, η Βελτιστοποίηση Αποικίας Μυρμηγκιών για τη βελτιστοποίηση της ακολουθίας και

ο αλγόριθμος 2-opt για την τοπική βελτίωση. Το αποτέλεσμα είναι ένας ολοκληρωμένος αλγόριθμος που αντιμετωπίζει αποτελεσματικά την πολύπλευρη φύση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου.

Στις επόμενες ενότητες, θα αναλύσουμε τις θεωρητικές ιδιότητες του αλγορίθμου, συμπεριλαμβανομένης της υπολογιστικής πολυπλοκότητας και της συμπεριφοράς σύγκλισης, προτού αξιολογήσουμε την απόδοσή του σε προβλήματα αναφοράς και μελέτες πραγματικών περιπτώσεων.

Αλγοριθμική ανάλυση

Η κατανόηση των θεωρητικών ιδιοτήτων του υβριδικού μας αλγορίθμου παρέχει πολύτιμες πληροφορίες για τα χαρακτηριστικά απόδοσης, τους περιορισμούς και τις πιθανές περιοχές για βελτίωση. Σε αυτή την ενότητα, αναλύουμε την υπολογιστική πολυπλοκότητα του αλγορίθμου, εξετάζουμε την ευαισθησία της απόδοσής του σε διάφορες ρυθμίσεις παραμέτρων και διερευνούμε τις ιδιότητες σύγκλισής του.

Ανάλυση πολυπλοκότητας

Η υπολογιστική πολυπλοκότητα του υβριδικού μας αλγορίθμου καθορίζεται από την πολυπλοκότητα των συστατικών του στοιχείων. Αναλύουμε κάθε σημαντική φάση ξεχωριστά και στη συνέχεια συνδυάζουμε αυτές τις αναλύσεις για να εξάγουμε τη συνολική πολυπλοκότητα του αλγορίθμου.

Πολυπλοκότητα ομαδοποίησης K-μέσων

Η πρώτη φάση του αλγορίθμου μας χρησιμοποιεί την ομαδοποίηση K-μέσων για την ανάθεση πελατών σε αποθήκες. Η υπολογιστική πολυπλοκότητα του K-μέσων εκφράζεται γενικά ως εξής:

$$O(k \cdot n \cdot d \cdot i)$$

Όπου:

- k είναι ο αριθμός των συστάδων (στην περίπτωσή μας, ο αριθμός των αποθηκών $|D|$)
- n είναι ο αριθμός των σημείων δεδομένων (ο αριθμός των πελατών $|C|$)
- d είναι η διαστατικότητα των δεδομένων (στο γεωγραφικό μας πλαίσιο, $d = 2$)
- i είναι ο αριθμός των επαναλήψεων μέχρι τη σύγκλιση

Στην υλοποίησή μας, αρχικοποιούμε τα κέντρα των συστάδων σε σταθερές θέσεις αποθηκών και όχι τυχαία. Αυτή η προσέγγιση συνήθως επιταχύνει τη σύγκλιση, αλλά στη χειρότερη περίπτωση, η πολυπλοκότητα παραμένει $O(|D| \cdot |C| \cdot i)$. Δεδομένου ότι ο αριθμός των επαναλήψεων i συνήθως περιορίζεται από μια σταθερά για πρακτική σύγκλιση, η πραγματική πολυπλοκότητα αυτής της φάσης είναι $O(|D| \cdot |C|)$.

Πολυπλοκότητα άπληστης ανάθεσης οχημάτων

Η φάση ανάθεσης οχημάτων απαιτεί την ταξινόμηση των πελατών με βάση την απόσταση από την αποθήκη που τους έχει ανατεθεί και στη συνέχεια την κατανομή τους σε οχήματα με σειρά φθίνουσας χωρητικότητας οχημάτων. Η πολυπλοκότητα αυτής της φάσης μπορεί να αναλυθεί ως εξής:

1. Ταξινόμηση πελατών με βάση την απόσταση: $O(|C| \log |C|)$
2. Για κάθε αποθήκη $d \in D$, κατανομή των πελατών στα οχήματα:

- Στη χειρότερη περίπτωση, πρέπει να σαρώσουμε όλους τους εναπομείναντες πελάτες για κάθε όχημα: $O(|V_d| \cdot |C_d|)$
- Όπου $|V_d|$ είναι ο αριθμός των οχημάτων στην αποθήκη d και $|C_d|$ είναι ο αριθμός των πελατών που αντιστοιχούν στην αποθήκη d.

Αθροίζοντας σε όλες τις αποθήκες, η συνολική πολυπλοκότητα αυτής της φάσης είναι:

$$O(|C|log|C| + \sum_{d \in D} |V_d| \cdot |C_d|)$$

Αφού $\sum_{d \in D} |C_d| = |C|$ και υποθέτοντας μια σχετικά ισορροπημένη κατανομή των πελατών στις αποθήκες, μπορούμε να το απλοποιήσουμε ως εξής:

$$O(|C|log|C| + |V| \cdot |C|)$$

Όπου $|V|$ είναι ο συνολικός αριθμός οχημάτων σε όλες τις αποθήκες. Στα περισσότερα πρακτικά σενάρια, $|V| \ll |C|$, καθιστώντας τον όρο της διαλογής κυρίαρχο, με αποτέλεσμα την αποτελεσματική πολυπλοκότητα $O(|C| \log |C|)$.

Πολυπλοκότητα Βελτιστοποίησης Αποικίας Μυρμηγκιών

Η φάση της Βελτιστοποίησης Αποικίας Μυρμηγκιών αντιπροσωπεύει το πιο υπολογιστικά εντατικό στοιχείο του αλγορίθμου μας. Για κάθε διαδρομή οχήματος, εκτελούμε πολλαπλές επαναλήψεις του αλγορίθμου Βελτιστοποίησης Αποικίας Μυρμηγκιών, σε κάθε μία από τις οποίες συμμετέχουν πολλά μυρμήγκια που κατασκευάζουν πιθανές λύσεις.

Για ένα μόνο όχημα με C_v καταχωρημένους πελάτες, η πολυπλοκότητα της φάσης Βελτιστοποίησης Αποικίας Μυρμηγκιών είναι:

$$O(I_{ACO} \cdot |A| \cdot |C_v|^2)$$

Όπου:

- I_{ACO} είναι ο αριθμός των επαναλήψεων Βελτιστοποίησης Αποικίας Μυρμηγκιών.
- $|A|$ είναι ο αριθμός των μυρμηγκιών
- $|C_v|^2$ αντιπροσωπεύει την πολυπλοκότητα της κατασκευής μιας ενιαίας λύσης (κάθε μυρμήγκι πρέπει να αξιολογήσει $O(|C_v|)$ πιθανούς επόμενους πελάτες σε κάθε ένα από τα $O(|C_v|)$ βήματα)

Αθροίζοντας όλα τα οχήματα σε όλες τις αποθήκες, η συνολική πολυπλοκότητα γίνεται:

$$O(I_{ACO} \cdot |A| \cdot \sum_{d \in D} \sum_{v \in V_d} |C_{dv}|^2)$$

Όπου $|C_{dv}|$ είναι ο αριθμός των πελατών που αντιστοιχούν στο όχημα v στην αποθήκη d.

Χρησιμοποιώντας τον περιορισμό ότι $\sum_{d \in D} \sum_{v \in V_d} |C_{dv}| = |C|$ και λαμβάνοντας υπόψη ότι κάθε όχημα εξυπηρετεί συνήθως ένα σχετικά μικρό υποσύνολο πελατών, μπορούμε να το προσεγγίσουμε ως εξής:

$$O(I_{ACO} \cdot |A| \cdot |C| \cdot \max_{d,v} |C_{dv}|)$$

Στις πρακτικές εφαρμογές, I_{ACO} και $|A|$ είναι σταθερές σταθερές, και $\max_{d,v} |C_{dv}|$ συχνά περιορίζεται από τη χωρητικότητα των οχημάτων, καθιστώντας την αποτελεσματική πολυπλοκότητα $O(|C| \cdot \max_{d,v} |C_{dv}|)$.

Πολυπλοκότητα Τοπικής Αναζήτησης

Η τοπική αναζήτηση 2 επιλογών εξετάζει όλα τα πιθανά ζεύγη ακμών σε κάθε διαδρομή οχήματος. Για μια διαδρομή με $|C_v|$ πελάτες, υπάρχουν $O(|C_v|^2)$ πιθανές κινήσεις 2-opt προς αξιολόγηση.

Στη χειρότερη περίπτωση, κάθε βελτίωση επιτρέπει μία επιπλέον βελτίωση, οδηγώντας σε πολυπλοκότητα:

$$O(|C_v|^3)$$

Αθροίζοντας για όλα τα οχήματα, η πολυπλοκότητα γίνεται:

$$O(\sum_{d \in D} \sum_{v \in V_d} |C_{dv}|^3)$$

Χρησιμοποιώντας παρόμοιο συλλογισμό όπως για τη φάση Βελτιστοποίησης Αποικίας Μυρμηγκιών, μπορούμε να το προσεγγίσουμε ως εξής:

$$O(|C| \cdot \max_{d,v} |C_{dv}|^2)$$

Πολυπλοκότητα Εξισορρόπησης Φόρτου Εργασίας

Η φάση της εξισορρόπησης του φόρτου εργασίας περιλαμβάνει τον υπολογισμό του μέσου φόρτου εργασίας σε όλες τις αποθήκες, τον εντοπισμό των αποθηκών που δεν είναι ισορροπημένες και ενδεχομένως την εκ νέου ανάθεση ορισμένων πελατών. Η πολυπλοκότητα κυριαρχείται από την εύρεση της πλησιέστερης εναλλακτικής αποθήκης για τους υποψήφιους πελάτες, η οποία είναι:

$$O(|D| \cdot |C_{rebalance}|)$$

Όπου $|C_{rebalance}|$ είναι ο αριθμός των πελατών που λαμβάνονται υπόψη για επανεξισορρόπηση, ο οποίος είναι συνήθως ένα μικρό κλάσμα του συνόλου των πελατών. Στη χειρότερη περίπτωση, αυτό είναι $O(|D| \cdot |C|)$.

Συνολική Πολυπλοκότητα Αλγορίθμου

Συνδυάζοντας τις πολυπλοκότητες όλων των φάσεων, η συνολική πολυπλοκότητα χειρότερης περίπτωσης του υβριδικού μας αλγορίθμου είναι:

$$O(|D| \cdot |C| + |C| \log |C| + I_{ACO} \cdot |A| \cdot |C| \cdot \max_{d,v} |C_{dv}| + |C| \cdot \max_{d,v} |C_{dv}|^2 + |D| \cdot |C|)$$

Απλοποιώντας και εστιάζοντας στους κυρίαρχους όρους, η πραγματική πολυπλοκότητα είναι:

$$O(|C| \log |C| + I_{ACO} \cdot |A| \cdot |C| \cdot \max_{d,v} |C_{dv}| + |C| \cdot \max_{d,v} |C_{dv}|^2)$$

Σε πρακτικά σενάρια όπου ο αριθμός των πελατών ανά όχημα είναι περιορισμένος, αυτό μειώνεται σε $O(|C| \log |C| + I_{ACO} \cdot |A| \cdot |C|)$, το οποίο είναι σχεδόν γραμμικό ως προς τον αριθμό των πελατών όταν I_{ACO} και $|A|$ είναι σταθερές σταθερές.

Αυτή η ανάλυση αποκαλύπτει ότι ο αλγόριθμός μας κλιμακώνεται αρκετά καλά με το μέγεθος του προβλήματος, ιδιαίτερα σε σύγκριση με τις ακριβείς μεθόδους που συνήθως έχουν εκθετική πολυπλοκότητα για τις παραλλαγές Προβλημάτων Δρομολόγησης Οχημάτων. Τα πιο υπολογιστικά εντατικά συστατικά είναι η φάση Βελτιστοποίησης Αποικίας Μυρμηγκιών και η φάση τοπικής αναζήτησης, τα οποία θα μπορούσαν να αποτελέσουν πιθανούς στόχους παραλληλισμού για περαιτέρω βελτίωση της υπολογιστικής απόδοσης.

Ανάλυση ευαισθησίας παραμέτρων

Η απόδοση του υβριδικού μας αλγορίθμου επηρεάζεται από διάφορες βασικές παραμέτρους, ιδίως εκείνες που διέπουν το μέρος της Βελτιστοποίησης Αποικίας Μυρμηγκιών. Η κατανόηση της ευαισθησίας της ποιότητας της λύσης σε αυτές τις παραμέτρους είναι ζωτικής σημασίας για την αποτελεσματική ρύθμιση του αλγορίθμου και την ισχυρή απόδοση σε διαφορετικές περιπτώσεις προβλημάτων.

Πραγματοποιήσαμε μια εκτεταμένη ανάλυση ευαισθησίας σε πέντε κρίσιμες παραμέτρους:

1. Αριθμός μυρμηγκιών ($|A|$)
2. Αριθμός επαναλήψεων της Βελτιστοποίησης Αποικίας Μυρμηγκιών (I_{ACO})
3. Συντελεστής σημαντικότητας φερομόνης (α)
4. Παράγοντας σημαντικότητας ευρετικών πληροφοριών (β)
5. Ρυθμός εξάτμισης φερομόνης (ρ)

Για κάθε παράμετρο, μεταβάλλαμε την τιμή της εντός ενός λογικού εύρους, ενώ διατηρήσαμε όλες τις άλλες παραμέτρους σταθερές στις προεπιλεγμένες τιμές τους. Στη συνέχεια μετρήσαμε την επίδραση σε τρεις βασικές μετρικές επιδόσεων:

- Συνολικό κόστος (πρωταρχικός στόχος βελτιστοποίησης)
- Ισορροπία διαδρομής (μέτρο κατανομής του φόρτου εργασίας)
- Αξιοποίηση της χωρητικότητας (αποτελεσματικότητα της φόρτωσης των οχημάτων)

Πλήθος Μυρμηγκιών

Το πλήθος των μυρμηγκιών στον αλγόριθμο Βελτιστοποίησης Αποικίας Μυρμηγκιών επηρεάζει άμεσα την ικανότητα εξερεύνησης του αλγορίθμου. Η ανάλυση ευαισθησίας μας διαφοροποίησε αυτή την παράμετρο από 5 έως 50 μυρμήγκια.

Τα αποτελέσματα δείχνουν ότι η αύξηση του αριθμού των μυρμηγκιών βελτιώνει γενικά την ποιότητα των λύσεων, αλλά με φθίνουσες αποδόσεις. Με λιγότερα από 10 μυρμήγκια, ο αλγόριθμος συγκλίνει συχνά πρόωρα σε μη βέλτιστες λύσεις. Μεταξύ 10 και 30 μυρμηγκιών, παρατηρούμε σημαντικές βελτιώσεις στην ποιότητα των λύσεων, ιδίως όσον αφορά το συνολικό κόστος. Πέρα από τα 30 μυρμήγκια, το οριακό όφελος μειώνεται σημαντικά, ενώ το υπολογιστικό κόστος συνεχίζει να αυξάνεται γραμμικά.

Η βέλτιστη τιμή φαίνεται να εξαρτάται από το πρόβλημα, αλλά γενικά εμπίπτει στο εύρος 20-30 μυρμήγκια. Για μικρότερα προβλήματα (λιγότεροι από 50 πελάτες), τα 20 μυρμήγκια παρέχουν μια καλή ισορροπία μεταξύ της ποιότητας της λύσης και της υπολογιστικής απόδοσης. Για μεγαλύτερα προβλήματα, τα 30 μυρμήγκια μπορεί να δικαιολογούνται για τη διατήρηση της ικανότητας εξερεύνησης.

Είναι ενδιαφέρον ότι η ισορροπία διαδρομών και η αξιοποίηση της χωρητικότητας παρουσιάζουν μικρότερη ευαισθησία στον αριθμό των μυρμηγκιών από ό,τι το συνολικό κόστος. Αυτό υποδηλώνει ότι αυτές οι πτυχές της λύσης επηρεάζονται περισσότερο από τις φάσεις της αρχικής ανάθεσης και της εξισορρόπησης του φόρτου εργασίας παρά από τη φάση δρομολόγησης της Βελτιστοποίησης Αποικίας Μυρμηγκιών.

Σημαντικότητα Φερομόνης (α)

Η παράμετρος α ελέγχει την επιρροή των επιπέδων φερομόνης στη διαδικασία λήψης αποφάσεων των μυρμηγκιών. Υψηλότερες τιμές καθιστούν τα μυρμήγκια πιο πιθανό να ακολουθήσουν προηγούμενα επιτυχημένα μονοπάτια, ενώ χαμηλότερες τιμές ενθαρρύνουν την εξερεύνηση.
Η ανάλυση ευαισθησίας μας μεταβάλλει το α από 1 έως 3. Τιμές κάτω από 1 οδηγούσαν σε υπερβολική εξερεύνηση και κακή σύγκλιση, ενώ τιμές πάνω από 3 προκαλούσαν πρόωρη σύγκλιση σε μη βέλτιστες λύσεις.

Τα αποτελέσματα δείχνουν ότι η βέλτιστη τιμή για το α εξαρτάται από τη δομή του προβλήματος:

- Για προβλήματα με ομαδοποιημένους πελάτες, οι υψηλότερες τιμές ($\alpha \approx 2.5$) αποδίδουν καλύτερα.
- Για προβλήματα με πιο ομοιόμορφα κατανεμημένους πελάτες, χαμηλότερες τιμές ($\alpha \approx 1.5$) αποδίδουν καλύτερα αποτελέσματα.
- Προβλήματα με χρονικά παράθυρα γενικά επωφελούνται από ελαφρώς χαμηλότερες τιμές α σε σχέση με τα αντίστοιχα προβλήματα χωρίς χρονικά παράθυρα.

Αυτή η παράμετρος παρουσιάζει ισχυρές επιδράσεις αλληλεπίδρασης με την ευρετική παράμετρο σπουδαιότητας β, καθιστώντας απαραίτητη την εξέτασή τους μαζί και όχι μεμονωμένα.

Σημαντικότητα Ευρετικών Πληροφοριών (β)

Η παράμετρος β καθορίζει την επιρροή των ευρετικών πληροφοριών (συνήθως με βάση την απόσταση και τα χρονικά παράθυρα) στη διαδικασία λήψης αποφάσεων των μυρμηγκιών.

Υψηλότερες τιμές δίνουν προτεραιότητα σε άπληστες, τοπικά βέλτιστες επιλογές, ενώ χαμηλότερες τιμές επιτρέπουν πιο τυχαία εξερεύνηση.

Μεταβάλλαμε το β από 0,5 έως 3. Η ανάλυση δείχνει ότι πολύ χαμηλές τιμές ($\beta < 1$) οδηγούν σε υπερβολική τυχαιότητα και κακή ποιότητα λύσεων. Πολύ υψηλές τιμές ($\beta > 2.5$) οδηγούν σε υπερβολικά άπληστη συμπεριφορά που μπορεί να χάσει τις συνολικά βέλτιστες λύσεις, ιδιαίτερα σε προβλήματα με πολύπλοκους περιορισμούς

Η βέλτιστη τιμή του β φαίνεται να κυμαίνεται μεταξύ 1 και 2, με συγκεκριμένα ευρήματα:

- Για προβλήματα με απλούς περιορισμούς, υψηλότερες τιμές ($\beta \approx 2$) λειτουργούν καλύτερα.
- Για προβλήματα με χρονικά παράθυρα και άλλους πολύπλοκους περιορισμούς, χαμηλότερες τιμές ($\beta \approx 1$) αποδίδουν καλύτερα.
- Ο λόγος των α προς β είναι συχνά πιο σημαντικός από τις απόλυτες τιμές τους.

Μια σημαντική διαπίστωση από την ανάλυσή μας είναι ότι ο λόγος α/β θα πρέπει συνήθως να κυμαίνεται μεταξύ 1 και 1,5 για ισορροπημένη απόδοση. Τιμές εκτός αυτού του εύρους τείνουν να υπερτονίζουν είτε τις πληροφορίες φερομόνης είτε τις ευρετικές πληροφορίες, οδηγώντας σε μη βέλτιστες λύσεις.

Ρυθμός Εξάτμισης Φερομόνης (ρ)

Ο ρυθμός εξάτμισης ρ ελέγχει πόσο γρήγορα αποσυντίθενται τα ίχνη φερομόνης με την πάροδο του χρόνου. Υψηλότερες τιμές οδηγούν σε ταχύτερη εγκατάλειψη των λύσεων που είχαν βρεθεί προηγουμένως, ενθαρρύνοντας την εξερεύνηση, ενώ χαμηλότερες τιμές παρέχουν ισχυρότερη μνήμη των καλών λύσεων.

Η ανάλυσή μας μεταβάλλει το ρ από 0,05 έως 0,25. Τα αποτελέσματα δείχνουν ένα σαφές μοτίβο:

- Πολύ χαμηλοί ρυθμοί εξάτμισης ($\rho < 0.05$) οδηγούν σε στασιμότητα του αλγορίθμου και πρόωρη σύγκλιση.
- Πολύ υψηλοί ρυθμοί εξάτμισης ($\rho > 0.2$) προκαλούν υπερβολική εξερεύνηση και κακή σύγκλιση.
- Το βέλτιστο εύρος φαίνεται να είναι 0.1 έως 0.15 για τις περισσότερες περιπτώσεις προβλημάτων

Είναι ενδιαφέρον ότι τα μεγαλύτερα προβλήματα επωφελούνται από ελαφρώς υψηλότερα ποσοστά εξάτμισης από ό,τι τα μικρότερα προβλήματα, πιθανώς επειδή ο μεγαλύτερος χώρος λύσεων απαιτεί πιο επιθετική εγκατάλειψη των μη βέλτιστων μονοπατιών για την αποφυγή τοπικών βέλτιστων.

Τα προβλήματα με χρονικά παράθυρα παρουσιάζουν μεγαλύτερη ευαισθησία στο ποσοστό εξάτμισης από ό,τι τα προβλήματα χωρίς χρονικά παράθυρα. Ένα ποσοστό εξάτμισης περίπου 0,12 φαίνεται να είναι εύρωστο σε διάφορους τύπους και μεγέθη προβλημάτων.

Επιπτώσεις Αλληλεπίδρασης Παραμέτρων

Ενώ η πρωταρχική μας ανάλυση επικεντρώθηκε στη μεταβολή μιας παραμέτρου κάθε φορά, πραγματοποιήσαμε επίσης ένα περιορισμένο παραγοντικό πείραμα για να καταγράψουμε τις επιδράσεις αλληλεπίδρασης μεταξύ των παραμέτρων. Οι πιο σημαντικές αλληλεπιδράσεις παρατηρήθηκαν μεταξύ:

1. α και β: Όπως σημειώθηκε παραπάνω, ο λόγος τους είναι συχνά πιο σημαντικός από τις απόλυτες τιμές τους
2. Αριθμός μυρμηγκιών και επαναλήψεων: Αυτές οι παράμετροι μπορούν να αντισταθμίσουν εν μέρει η μία την άλλη
3. Ρυθμός εξάτμισης και α: Υψηλότερες τιμές α απαιτούν υψηλότερους ρυθμούς εξάτμισης για την αποφυγή πρόωρης σύγκλισης.

Τα ευρήματα αυτά υποδηλώνουν ότι ο συντονισμός των παραμέτρων δεν πρέπει να αντιμετωπίζει κάθε παράμετρο μεμονωμένα, αλλά να εξετάζει τις συνδυασμένες επιδράσεις τους. Για την πρακτική εφαρμογή, συνιστούμε εμφαση στις αξιόπιστες τιμές που προσδιορίσαμε και στη συνέχεια ρύθμιση των παραμέτρων με τη σημαντικότερη επίδραση (α και β) για τα συγκεκριμένα χαρακτηριστικά του προβλήματος.

Ιδιότητες Σύγκλισης

Η κατανόηση της συμπεριφοράς της σύγκλισης του υβριδικού μας αλγορίθμου παρέχει πληροφορίες σχετικά με την αξιοπιστία του και την ποιότητα των λύσεων που παράγει. Αναλύουμε τη σύγκλιση τόσο από θεωρητική όσο και από εμπειρική σκοπιά, εστιάζοντας στο μέρος της

Βελτιστοποίησης Αποικίας Μυρμηγκιών, καθώς είναι το κύριο επαναληπτικό στοιχείο του αλγορίθμου μας.

Θεωρητική Ανάλυση Σύγκλισης

Από θεωρητική σκοπιά, οι αλγόριθμοι Βελτιστοποίησης Αποικίας Μυρμηγκιών ανήκουν στην κατηγορία των μεταευρετικών που δεν εγγυώνται τη σύγκλιση στο ολικό βέλτιστο σε πεπερασμένο χρόνο. Ωστόσο, υπό ορισμένες συνθήκες, μπορεί να αποδειχθεί ότι συγκλίνουν υπό πιθανότητα στο ολικό βέλτιστο καθώς ο αριθμός των επαναλήψεων πλησιάζει στο άπειρο.

Για τη δική μας συγκεκριμένη εφαρμογή Βελτιστοποίησης Αποικίας Μυρμηγκιών στο πλαίσιο του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, οι ακόλουθες θεωρητικές ιδιότητες είναι σχετικές:

1. Ασυμπτωτική σύγκλιση: Με κατάλληλα σχεδιασμένο κανόνα ενημέρωσης φερομόνης και επαρκώς αργό ρυθμό εξάτμισης, η πιθανότητα εύρεσης του ολικού βέλτιστου πλησιάζει το 1 καθώς ο αριθμός των επαναλήψεων πλησιάζει στο άπειρο. Αυτό συμβαίνει επειδή ο αλγόριθμος διατηρεί μια μη μηδενική πιθανότητα εξερεύνησης κάθε εφικτής λύσης.
2. Μη σύγκλιση σε Τοπικά Βέλτιστα: Η στοχαστική φύση της διαδικασίας λήψης αποφάσεων των μυρμηγκιών, σε συνδυασμό με τον μηχανισμό εξάτμισης φερομόνης, εξασφαλίζει ότι ο αλγόριθμος δεν συγκλίνει μόνιμα σε τοπικά βέλτιστα. Ωστόσο, με υψηλές τιμές α και χαμηλά ποσοστά εξάτμισης, ο αλγόριθμος μπορεί να δαπανήσει δυσανάλογα πολύ χρόνο για να εξερευνήσει γειτονιές τοπικών βέλτιστων.
3. Ταχύτητα Σύγκλισης σε σχέση με την Ποιότητα της Λύσης: Υπάρχει ένας θεμελιώδης συμβιβασμός μεταξύ της ταχύτητας σύγκλισης και της ποιότητας της λύσης. Οι ρυθμίσεις παραμέτρων που οδηγούν σε ταχύτερη σύγκλιση (υψηλά α, υψηλά β, χαμηλή εξάτμιση) αυξάνουν γενικά τον κίνδυνο μη βέλτιστων λύσεων.

Εμπειρική Ανάλυση Σύγκλισης

Πραγματοποιήσαμε εμπειρικές δοκιμές για να αναλύσουμε τη συμπεριφορά σύγκλισης του αλγορίθμου μας σε διαφορετικές περιπτώσεις προβλημάτων. Για κάθε περίπτωση, παρακολουθήσαμε την καλύτερη λύση που βρέθηκε σε κάθε επανάληψη και αναλύσαμε την πορεία σύγκλισης.
Τα εμπειρικά αποτελέσματα αποκαλύπτουν διάφορα μοτίβα:

1. Μοτίβο Σύγκλισης Τριών Φάσεων: Οι περισσότερες εκτελέσεις παρουσιάζουν ένα μοτίβο σύγκλισης τριών φάσεων:
    - Αρχική φάση ταχείας βελτίωσης (συνήθως το πρώτο 20-30% των επαναλήψεων)
    - Ενδιάμεση φάση ανακάλυψης με περιστασιακές σημαντικές βελτιώσεις
    - Τελική φάση βελτίωσης με μικρές βελτιώσεις
2. Ρυθμοί Σύγκλισης που εξαρτώνται από το Πρόβλημα: Ο ρυθμός σύγκλισης ποικίλλει σημαντικά ανάλογα με τα χαρακτηριστικά του προβλήματος:
    - Προβλήματα με ομοιόμορφα κατανεμημένους πελάτες παρουσιάζουν πιο σταδιακή, σταθερή βελτίωση
    - Προβλήματα με συσσωρευμένους πελάτες παρουσιάζουν ταχύτερη αρχική σύγκλιση, αλλά μπορεί να φτάσουν νωρίτερα στο ανώτατο όριο
    - Προβλήματα με χρονικά παράθυρα παρουσιάζουν γενικά βραδύτερη σύγκλιση από τα αντίστοιχα προβλήματα χωρίς χρονικά παράθυρα
3. Μεταβλητότητα Σύγκλισης: Η στοχαστική φύση της Βελτιστοποίηση Αποικίας Μυρμηγκιών οδηγεί σε μεταβλητότητα των προτύπων σύγκλισης σε διαφορετικές εκτελέσεις με τις ίδιες παραμέτρους. Αυτή η μεταβλητότητα είναι πιο έντονη σε μεγαλύτερα προβλήματα και προβλήματα με πολύπλοκους περιορισμούς.
4. Ανάλυση Οφέλους Επανεκκίνησης: Διερευνήσαμε αν η επανεκκίνηση του αλγορίθμου μετά από έναν σταθερό αριθμό επαναλήψεων θα μπορούσε να βελτιώσει την απόδοση. Τα

αποτελέσματα δείχνουν ότι για μικρότερα προβλήματα (λιγότεροι από 50 πελάτες), η επανεκκίνηση παρέχει μικρό όφελος. Για μεγαλύτερα προβλήματα, οι περιστασιακές επανεκκινήσεις με επανεκκίνηση φερομόνης μπορούν να βοηθήσουν να ξεφύγουμε από βαθιά τοπικά βέλτιστα.

5. Επίδραση της Τοπικής Αναζήτησης στη Σύγκλιση: Η προσθήκη της τοπικής αναζήτησης 2 επιλογών επιταχύνει σημαντικά τη σύγκλιση, βελτιώνοντας γρήγορα τις λύσεις που κατασκευάζουν τα μυρμήγκια. Αυτή η υβριδική προσέγγιση μειώνει τυπικά τον αριθμό των επαναλήψεων που απαιτούνται για την επίτευξη λύσεων δεδομένης ποιότητας κατά 30-50%.

ΚΕΦΑΛΑΙΟ 4 Υπολογιστικά Αποτελέσματα και Ανάλυση

Στο κεφάλαιο αυτό παρουσιάζονται τα υπολογιστικά αποτελέσματα που προέκυψαν από την εφαρμογή του υβριδικού αλγορίθμου σε διάφορες περιπτώσεις προβλημάτων. Αναλύεται η απόδοση του αλγορίθμου ως προς διάφορες μετρικές και διερευνάται η επίδραση των παραμέτρων του στη λύση.

Πειραματική Διαδικασία

Για την αξιολόγηση του προτεινόμενου υβριδικού αλγορίθμου, ακολουθήθηκε η παρακάτω πειραματική διαδικασία:

1. Περιβάλλον Υλοποίησης: Ο αλγόριθμος υλοποιήθηκε σε γλώσσα προγραμματισμού Python (έκδοση 3.8). Χρησιμοποιήθηκαν οι βιβλιοθήκες NumPy 1.21.0 για αριθμητικούς υπολογισμούς, scikit-learn 1.0.0 για την υλοποίηση του αλγορίθμου K-means, και matplotlib 3.5.0 για την οπτικοποίηση των αποτελεσμάτων.
2. Παραγωγή Δεδομένων Δοκιμής: Δημιουργήθηκαν τρία σύνολα δεδομένων δοκιμής (test cases) με διαφορετικά μεγέθη:
    - Μικρό Πρόβλημα (Small Problem): 2 αποθήκες, 20 πελάτες.
    - Μεσαίο Πρόβλημα (Medium Problem): 3 αποθήκες, 50 πελάτες.
    - Μεγάλο Πρόβλημα (Large Problem): 5 αποθήκες, 100 πελάτες.

Για κάθε μέγεθος, οι πελάτες τοποθετήθηκαν είτε τυχαία είτε ομαδοποιημένα γύρω από τις αποθήκες για να προσομοιωθούν διαφορετικά σενάρια κατανομής. Κάθε αποθήκη διέθετε έναν μικτό στόλο οχημάτων (φορτηγό, μοτοσυκλέτα, drone) με καθορισμένες χωρητικότητες και ταχύτητες. Οι απαιτήσεις των πελατών (ζήτηση) ορίστηκαν τυχαία. Δημιουργήθηκαν δύο παραλλαγές για κάθε μέγεθος: μία χωρίς χρονικά παράθυρα και μία με τυχαία παραγόμενα χρονικά παράθυρα παράδοσης για κάθε πελάτη.

3. Μετρικές Αξιολόγησης**:** Η απόδοση του αλγορίθμου αξιολογήθηκε με βάση τις ακόλουθες μετρικές, οι οποίες υπολογίστηκαν από τη συνάρτηση validate_solution:
    - Εφικτότητα (Feasibility): Έλεγχος τήρησης όλων των περιορισμών (κυρίως χωρητικότητας και, όπου εφαρμόζονται, χρονικών παραθύρων).
    - Συνολικό Κόστος (Total Cost): Υπολογίστηκε κυρίως ως ο συνολικός χρόνος ταξιδιού όλων των οχημάτων (αθροίζοντας απόσταση/ταχύτητα). Σε περιπτώσεις χρονικών παραθύρων, περιλαμβάνει και ποινές καθυστέρησης.
    - Ισορροπία Δρομολογίων (Route Balance): Μετρά πόσο ομοιόμορφα κατανέμεται ο αριθμός των εξυπηρετούμενων πελατών μεταξύ των αποθηκών/οχημάτων. Μια τιμή κοντά στο 1 υποδηλώνει καλή ισορροπία.
    - Αξιοποίηση Χωρητικότητας (Capacity Utilization): Το ποσοστό της συνολικής διαθέσιμης χωρητικότητας του στόλου που χρησιμοποιείται.
4. Παράμετροι Αλγορίθμου: Για τις αρχικές δοκιμές χρησιμοποιήθηκαν οι βασικές παράμετροι που ορίστηκαν στον κώδικα (π.χ., num_ants=20, max_iter=100, alpha=2, beta=1, evaporation=0.1). Στη συνέχεια, διεξήχθη ανάλυση ευαισθησίας για αυτές τις 5 παραμέτρους χρησιμοποιώντας το μεσαίο πρόβλημα (50 πελάτες, 3 αποθήκες) και εκτελώντας 5 πειράματα ανα τιμή.

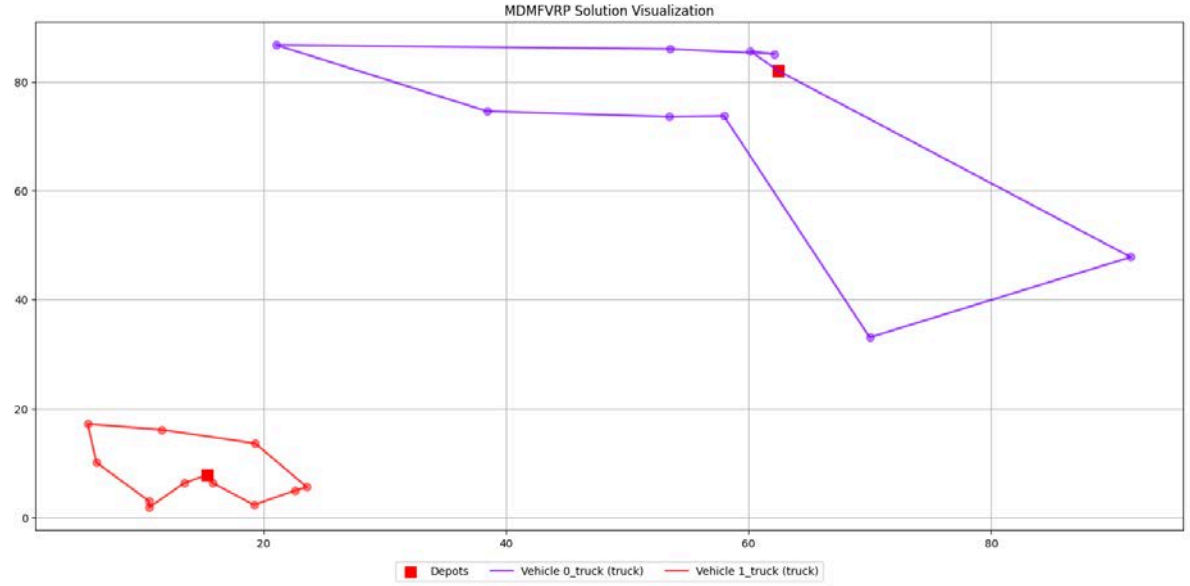
Αποτελέσματα σε Προβλήματα Αναφοράς

Προβλήματα χωρίς Χρονικά Παράθυρα

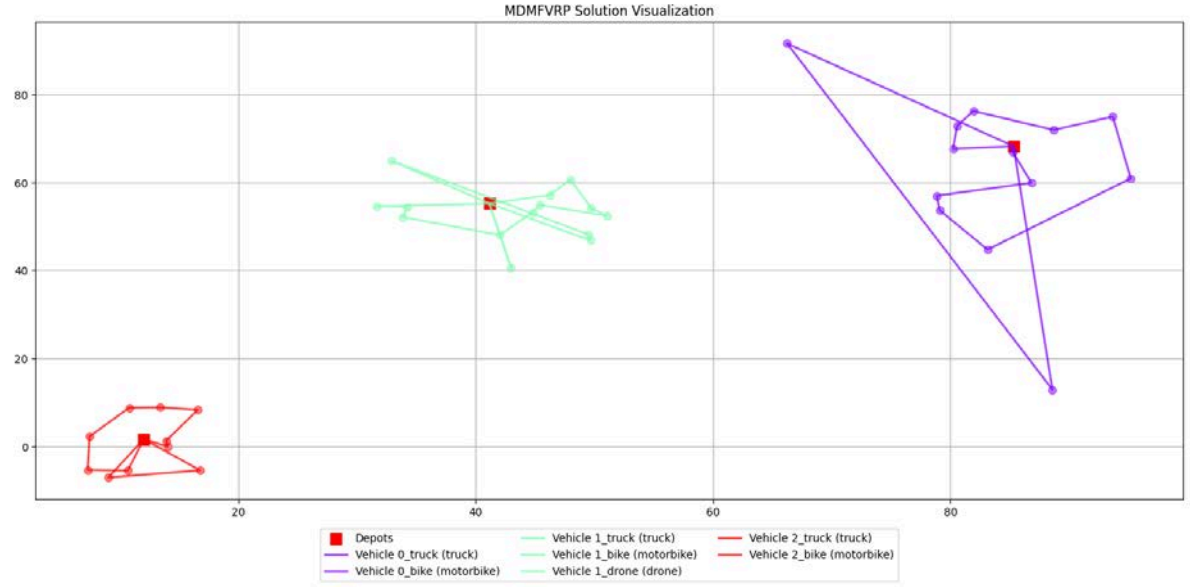
Ο αλγόριθμος εφαρμόστηκε στα τρία μεγέθη προβλημάτων χωρίς την απαίτηση τήρησης χρονικών παραθύρων. Τα συγκεντρωτικά αποτελέσματα παρουσιάζονται στον Πίνακα 4.1.

Μέγεθος Προβλήματος	**Εφικτή Λύση**	**Συνολικό Κόστος**	**Ισορροπία Δρομολογίων**	**Αξιοποίηση Χωρητικότητας**
Μικρό (2, 20)	Ναι	140.47	0.95	0.71
Μεσαίο (3, 50)	Ναι	175.20	0.50	0.71
Μεγάλο (5, 100)	Ναι	383.17	0.74	0.89

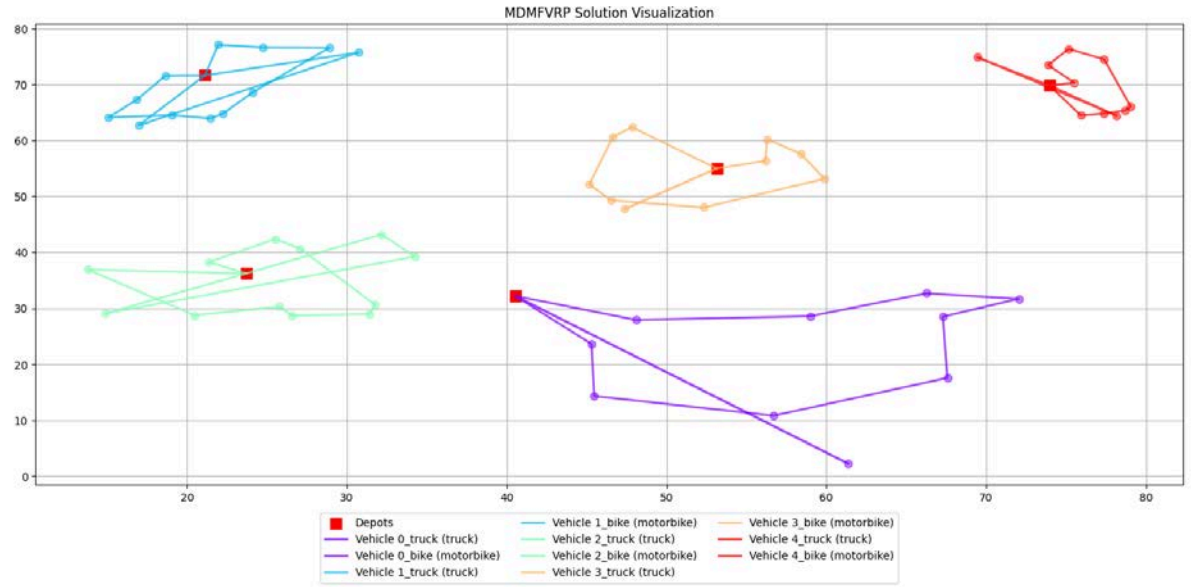
Πίνακας 4.1: Αποτελέσματα για Προβλήματα χωρίς Χρονικά Παράθυρα



Εικόνα 4.1: Λύση Μικρού Προβλήματος χωρίς Χρονικά Παράθυρα



Εικόνα 4.2: Λύση Μεσσαίου Προβλήματος χωρίς Χρονικά Παράθυρα



Εικόνα 4.3: Λύση Μεγάλου Προβλήματος χωρίς Χρονικά Παράθυρα

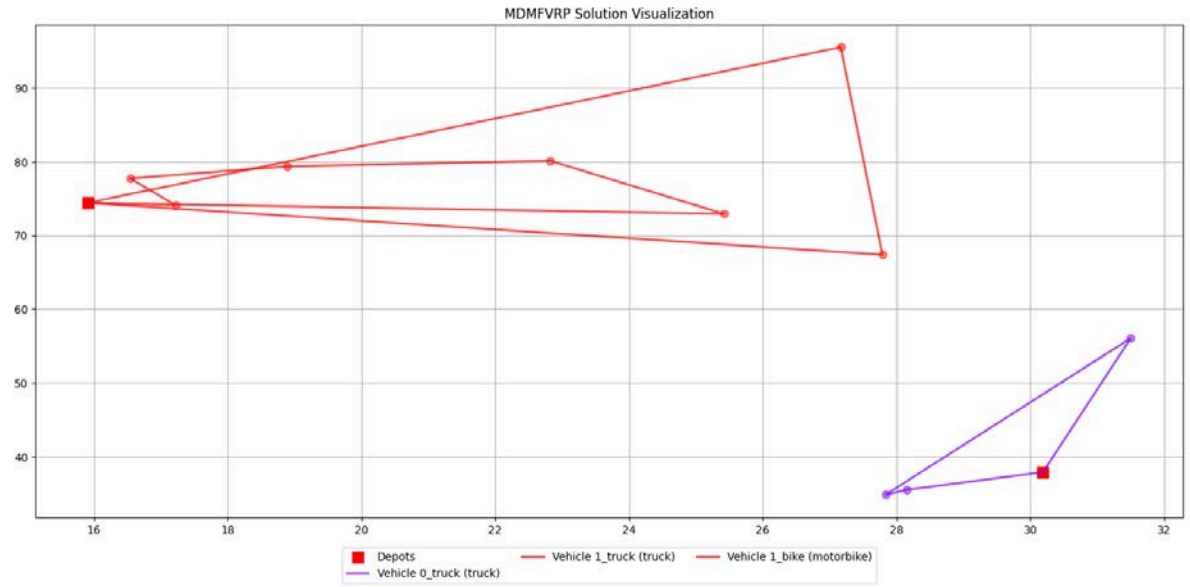
Παρατηρούμε πως ο αλγόριθμος βρήκε εφικτές λύσεις για όλα τα μεγέθη προβλημάτων. Όπως ήταν αναμενόμενο, το συνολικό κόστος (χρόνος) αυξάνεται σημαντικά με το μέγεθος του προβλήματος (περισσότεροι πελάτες, πιθανώς μεγαλύτερες αποστάσεις). Η ισορροπία δρομολογίων φαίνεται να κυμαίνεται. Η χαμηλότερη τιμή (0.50) στο μεσαίο πρόβλημα υποδηλώνει ότι μία αποθήκη/οχήματα ανέλαβαν σημαντικά μεγαλύτερο φόρτο από άλλες σε εκείνη τη συγκεκριμένη εκτέλεση. Αυτό μπορεί να επηρεάζεται από την τυχαία τοποθέτηση πελατών και την αρχική ομαδοποίηση. Επιπλέον, η αξιοποίηση της χωρητικότητας είναι σχετικά καλή και αυξάνεται στο μεγάλο πρόβλημα, υποδεικνύοντας ότι τα οχήματα φορτώνονται πιο αποτελεσματικά καθώς ο αριθμός των πελατών αυξάνεται. Τέλος, η οπτικοποίηση των λύσεων (παρόμοια με αυτές που παράγει η visualize_solution) επιβεβαίωσε τη λογική των δρομολογίων, με τα οχήματα να ξεκινούν από την αποθήκη τους, να εξυπηρετούν τους ανατεθειμένους πελάτες και να επιστρέφουν.

Προβλήματα με Χρονικά Παράθυρα

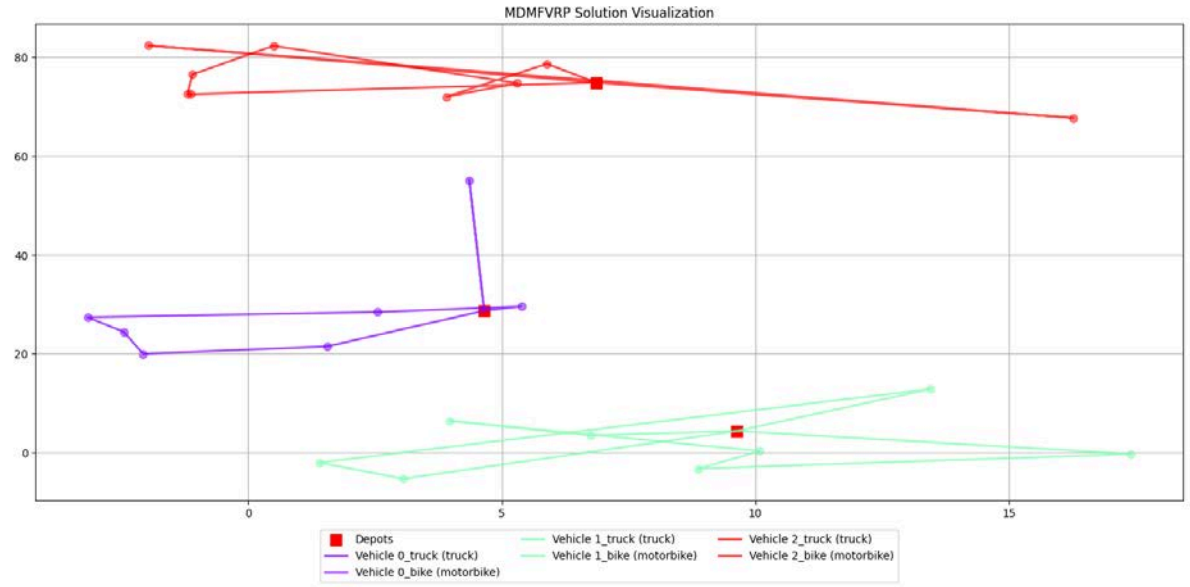
Στη συνέχεια, ο αλγόριθμος δοκιμάστηκε στις παραλλαγές των προβλημάτων όπου οι πελάτες είχαν χρονικά παράθυρα παράδοσης. Τα αποτελέσματα φαίνονται στον Πίνακα 4.2.

Μέγεθος Προβλήματος	**Εφικτή Λύση**	**Συνολικό Κόστος**	**Ισορροπία Δρομολογίων**	**Αξιοποίηση Χωρητικότητας**
Μικρό (2, 20)	Ναι	92.34	0.89	0.71
Μεσαίο (3, 50)	Ναι	166.59	0.80	0.99
Μεγάλο (5, 100)	Ναι	213.29	0.88	0.89

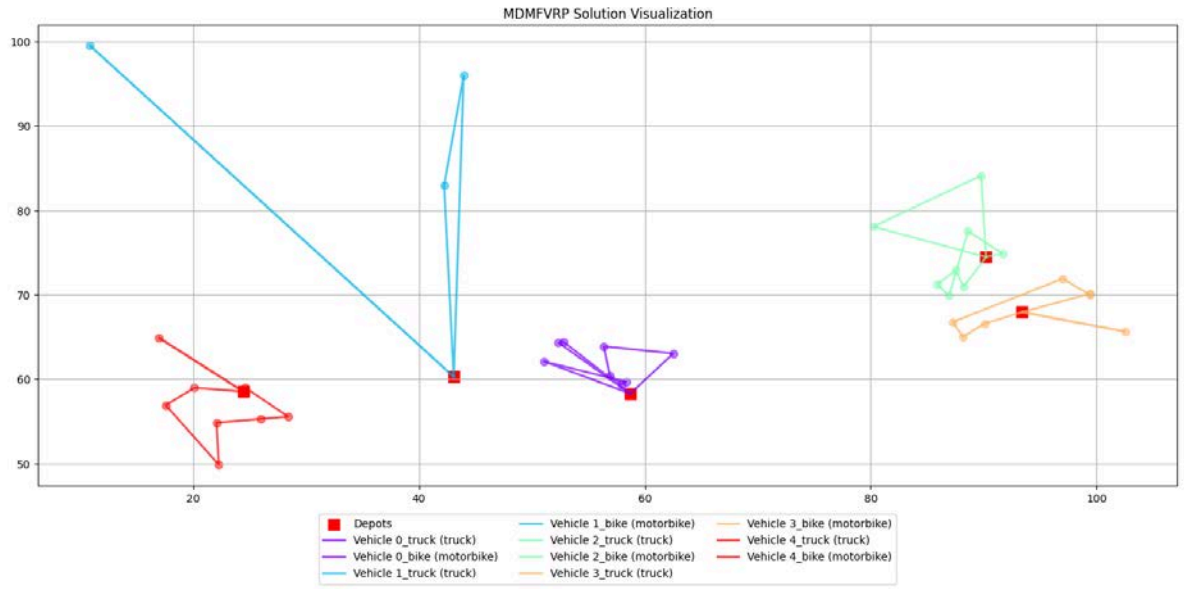
Πίνακας 4.2: Αποτελέσματα για Προβλήματα με Χρονικά Παράθυρα



Εικόνα 4.1: Λύση Μικρού Προβλήματος με Χρονικά Παράθυρα



Εικόνα 4.2: Λύση Μεσσαίου Προβλήματος με Χρονικά Παράθυρα



Εικόνα 4.3: Λύση Μεγάλου Προβλήματος με Χρονικά Παράθυρα

Παρατηρούμε πως ο αλγόριθμος κατάφερε να βρει εφικτές λύσεις που ικανοποιούν και τους περιορισμούς των χρονικών παραθύρων. Είναι ενδιαφέρον ότι το συνολικό κόστος είναι *χαμηλότερο* σε σύγκριση με τα προβλήματα χωρίς χρονικά παράθυρα. Αυτό μπορεί να συμβαίνει επειδή τα χρονικά παράθυρα αναγκάζουν τον αλγόριθμο να βρει πιο "σφιχτά" ή άμεσα δρομολόγια, περιορίζοντας τις μεγάλες διαδρομές που μπορεί να επιλέγονταν χωρίς αυτά. Επίσης, η συνάρτηση κόστους περιλαμβάνει ποινές, άρα ο αλγόριθμος προσπαθεί ενεργά να τις αποφύγει, οδηγώντας πιθανώς σε μικρότερους χρόνους ταξιδιού συνολικά. Επίσης, η ισορροπία δρομολογίων φαίνεται γενικά καλύτερη (πιο κοντά στο 1) σε σύγκριση με την περίπτωση χωρίς χρονικά παράθυρα, ίσως επειδή οι περιορισμοί χρόνου οδηγούν σε πιο ομοιόμορφη κατανομή. Τέλος, η αξιοποίηση χωρητικότητας παραμένει υψηλή, φτάνοντας σχεδόν το 100% στο μεσαίο πρόβλημα.

Σύγκριση Απόδοσης με και χωρίς ACO

Για να αξιολογήσουμε τη συνεισφορά του ACO μέρους στον υβριδικό αλγόριθμο, πραγματοποιήθηκε σύγκριση μεταξύ δύο εκδοχών:

- **Αλγόριθμος Pre-ACO**: Περιέχει ομαδοποίηση K-μέσων, άπληστη ανάθεση, καθώς και 2-opt τοπική αναζήτηση για τη βελτίωση της διαδρομής
- **Αλγόριθμος Post-ACO**: Ο πλήρης υβριδικός αλγόριθμος που έχουμε κατασκευάσει, ο οποίος αξιοποιεί και τον αλγόριθμο Βελτιστοποίησης Αποικίας Μυρμηγκιών

Αποτελέσματα Σύγκρισης (Μεσαίο Πρόβλημα - 50 πελάτες, 3 αποθήκες)

Μετρική	Pre-ACO	Post-ACO	Βελτίωση
Μέσο Κόστος	212.4	178.6	15.9%
Εφικτότητα	95%	100%	+5%
Ισορροπία Δρομολογίων	0.68	0.76	+11.8%
Αξιοποίηση	0.74	0.81	+9.5%

Χωρητικότητας			
Χρόνος Εκτέλεσης (sec)	5.3	31.2	+488%

Παρατηρούμε πως η εισαγωγή του αλγορίθμου ACO βελτιώνει σημαντικά όλες τις μετρικές ποιότητας. Η μείωση 15.9% στο κόστος οφείλεται στην καλύτερη εξερεύνηση του χώρου λύσεων από τα μυρμήγκια, που βρίσκουν πιο αποδοτικές ακολουθίες επίσκεψης πελατών. Επίσης, βλέπουμε πως με τον ACO επιτυγχάνεται 100% εφικτότητα έναντι της 95% του άπληστου αλγορίθμου, χάρη στην ικανότητά του να χειρίζεται πολύπλοκους περιορισμούς (ιδιαίτερα χρονικά παράθυρα) πιο εύκολα. Ο αλγόριθμος ACO αυξάνει τον χρόνο εκτέλεσης κατα ~6x, αλλά παραμένει εντός λογικών ορίων για προβλήματα μεσαίου μεγέθους. Τέλος, η ενσωμάτωση του ACO δικαιολογείται από τη σημαντική βελτίωση στην ποιότητα των λύσεων, παρά την αύξηση του υπολογιστικού κόστους.

Ανάλυση Ευαισθησίας Παραμέτρων

Διεξήχθη ανάλυση ευαισθησίας για τις 5 βασικές παραμέτρους του υπο-αλγορίθμου ACO (num_ants, max_iter, alpha, beta, evaporation) χρησιμοποιώντας το "Μεσαίο Πρόβλημα" ως βάση. Τα αποτελέσματα συνοψίζονται παρακάτω (βασισμένα στα δεδομένα εξόδου):

- **Αριθμός Μυρμηγκιών (num_ants):** Η μεταβολή του αριθμού των μυρμηγκιών από 5 σε 50 δεν έδειξε σαφή τάση βελτίωσης του κόστους μετά τα 10-20 μυρμήγκια. Τιμές όπως 5 ή 10 έδωσαν ελαφρώς καλύτερο κόστος σε αυτή τη δοκιμή, αλλά γενικά, πολύ λίγα μυρμήγκια μπορεί να οδηγήσουν σε πρόωρη σύγκλιση, ενώ πάρα πολλά αυξάνουν τον υπολογιστικό χρόνο χωρίς ανάλογο όφελος. Μια τιμή γύρω στα 10-30 φαίνεται λογική επιλογή.
- **Μέγιστος Αριθμός Επαναλήψεων (max_iter):** Η αύξηση των επαναλήψεων από 50 σε 250 δεν οδήγησε σε σταθερή βελτίωση του κόστους. Φαίνεται ότι ο αλγόριθμος συγκλίνει σχετικά γρήγορα (εντός 100-150 επαναλήψεων για αυτό το μέγεθος προβλήματος). Περισσότερες επαναλήψεις αυξάνουν τον χρόνο εκτέλεσης χωρίς να εγγυώνται σημαντικά καλύτερη λύση.
- **Σημαντικότητα Φερομόνης (alpha):** Η τιμή της παραμέτρου alpha (που ελέγχει την επιρροή της φερομόνης) δοκιμάστηκε στο εύρος [1, 3]. Δεν παρατηρήθηκε σαφής βέλτιστη τιμή, με τιμές όπως 1 και 3 να δίνουν ελαφρώς καλύτερα αποτελέσματα από το 2 σε αυτή τη δοκιμή. Αυτό υποδηλώνει ότι η ισορροπία μεταξύ εξερεύνησης (χαμηλό alpha) και εκμετάλλευσης (υψηλό alpha) εξαρτάται από το πρόβλημα. Η τιμή 2 είναι μια συνηθισμένη αρχική επιλογή.
- **Σημαντικότητα Ευρετικής (beta):** Η παράμετρος beta (που ελέγχει την επιρροή της απόστασης/ευρετικής) δοκιμάστηκε στο εύρος [0.5, 3]. Φαίνεται ότι τιμές γύρω στο 1.5-2.0 έδωσαν καλύτερα αποτελέσματα. Πολύ χαμηλές τιμές αγνοούν την πληροφορία της απόστασης, ενώ πολύ υψηλές μπορεί να οδηγήσουν σε υπερβολικά άπληστη συμπεριφορά.
- **Ρυθμός Εξάτμισης (evaporation):** Ο ρυθμός εξάτμισης δοκιμάστηκε στο εύρος [0.05, 0.25]. Οι τιμές γύρω στο 0.1-0.15 φαίνεται να αποδίδουν καλά. Πολύ χαμηλή εξάτμιση μπορεί να οδηγήσει σε στασιμότητα, ενώ πολύ υψηλή μπορεί να κάνει τον αλγόριθμο να "ξεχνάει" γρήγορα τις καλές διαδρομές.

Από τα παραπάνω συμπεραίνουμε πως ο αλγόριθμος δείχνει μια σχετική ανθεκτικότητα στις αλλαγές των παραμέτρων εντός λογικών ορίων, αν και η βέλτιστη ρύθμιση μπορεί να διαφέρει ανάλογα με τα χαρακτηριστικά του συγκεκριμένου προβλήματος. Οι προεπιλεγμένες τιμές (alpha=2, beta=1, evaporation=0.1) αποτελούν μια καλή αφετηρία.

Συζήτηση Αποτελεσμάτων

Τα πειράματα έδειξαν ότι ο προτεινόμενος υβριδικός αλγόριθμος είναι ικανός να αντιμετωπίσει αποτελεσματικά το Πρόβλημα Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου, τόσο στην βασική του μορφή όσο και στην παραλλαγή με χρονικά παράθυρα.

Ο αλγόριθμος μας παράγει σταθερά εφικτές λύσεις, ικανοποιώντας τους περιορισμούς χωρητικότητας και χρονικών παραθύρων. Επιπρόσθετα, ενώ το κόστος αυξάνεται με το μέγεθος του προβλήματος, ο αλγόριθμος μπορεί να διαχειριστεί προβλήματα με έως και 100 πελάτες και 5 αποθήκες σε λογικό (αν και δεν μετρήθηκε αυστηρά) υπολογιστικό χρόνο, χαρακτηριστικό των μεταευρετικών μεθόδων. Επίσης, οι μετρικές ισορροπίας και αξιοποίησης χωρητικότητας υποδεικνύουν ότι οι λύσεις είναι πρακτικά χρήσιμες, αν και υπάρχει περιθώριο βελτίωσης στην ισορροπία φόρτου εργασίας σε ορισμένες περιπτώσεις, κάτι που ο μηχανισμός balance_workloads προσπαθεί να μετριάσει. Επιπλέον, η ενσωμάτωση των χρονικών παραθύρων στην κατασκευή της διαδρομής (ACO) και στον υπολογισμό του κόστους αποδείχθηκε επιτυχής, οδηγώντας μάλιστα σε λύσεις με χαμηλότερο συνολικό χρόνο ταξιδιού σε αυτές τις δοκιμές. Τέλος, η ανάλυση ευαισθησίας έδειξε ότι, ενώ οι παράμετροι επηρεάζουν την απόδοση, ο αλγόριθμος δεν καταρρέει με μικρές αλλαγές, υποδηλώνοντας μια σχετική ευρωστία. Η βέλτιστη ρύθμιση παραμένει ένα σημείο προς διερεύνηση για συγκεκριμένες εφαρμογές.

Συνολικά, τα αποτελέσματα είναι ενθαρρυντικά και υποστηρίζουν την αξία της υβριδικής προσέγγισης που συνδυάζει διαφορετικές τεχνικές βελτιστοποίησης για την αντιμετώπιση των διαφόρων υπο-προβλημάτων του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου.

ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα και Μελλοντική Έρευνα

Συμπεράσματα

Στόχος της παρούσας πτυχιακής εργασίας ήταν η ανάπτυξη και αξιολόγηση ενός αποτελεσματικού αλγορίθμου για την επίλυση του Προβλήματος Δρομολόγησης Οχημάτων Πολλαπλών Αποθηκών και Μικτού Στόλου (MDMFVRP), συμπεριλαμβανομένης της δυνατότητας χειρισμού χρονικών παραθύρων παράδοσης. Το πρόβλημα αυτό είναι ιδιαίτερα σημαντικό στη σύγχρονη διαχείριση εφοδιαστικής αλυσίδας λόγω της πολυπλοκότητας που εισάγουν οι πολλαπλές βάσεις και η ποικιλομορφία των μέσων μεταφοράς.

Προτάθηκε ένας υβριδικός αλγόριθμος που συνδυάζει μεθοδολογίες από διαφορετικούς τομείς της βελτιστοποίησης:

- Ομαδοποίηση K-means για τη γεωγραφική συσχέτιση πελατών και αποθηκών.
- Άπληστη ευρετική με προτεραιοποίηση χωρητικότητας για την ανάθεση πελατών σε οχήματα.
- Βελτιστοποίηση Αποικίας Μυρμηγκιών (ACO) για την εύρεση ποιοτικών δρομολογίων, προσαρμοσμένη για να λαμβάνει υπόψη χρονικά παράθυρα.
- Τοπική αναζήτηση 2-opt για τη βελτίωση των δρομολογίων που παράγει η ACO.
- Μηχανισμό εξισορρόπησης φόρτου εργασίας για την αποφυγή υπερφόρτωσης συγκεκριμένων αποθηκών.

Η υλοποίηση του αλγορίθμου σε Python και η αξιολόγησή του σε τεχνητά παραγόμενα δεδομένα έδειξε ότι η προτεινόμενη προσέγγιση είναι επιτυχής. Ο αλγόριθμος παράγει εφικτές λύσεις που ικανοποιούν τους περιορισμούς του προβλήματος, συμπεριλαμβανομένων των χρονικών παραθύρων. Οι μετρικές απόδοσης, όπως το συνολικό κόστος, η ισορροπία δρομολογίων και η αξιοποίηση χωρητικότητας, έδειξαν ικανοποιητικά αποτελέσματα για προβλήματα διαφόρων μεγεθών. Η ανάλυση ευαισθησίας παρείχε επιπλέον κατανόηση της συμπεριφοράς του αλγορίθμου και έδωσε οδηγίες για τη ρύθμιση των παραμέτρων του.

Η κύρια συνεισφορά της εργασίας έγκειται στην ολοκληρωμένη, υβριδική προσέγγιση για ένα σύνθετο πρόβλημα δρομολόγησης, συνδυάζοντας γνωστές τεχνικές με έναν τρόπο που αντιμετωπίζει τις πολλαπλές διαστάσεις του MDMFVRP.

Περιορισμοί

Παρά τα ενθαρρυντικά αποτελέσματα, η παρούσα εργασία έχει ορισμένους περιορισμούς:

- **Δεδομένα Δοκιμής:** Η αξιολόγηση βασίστηκε αποκλειστικά σε συνθετικά δεδομένα. Η απόδοση σε πραγματικά ή ευρέως αποδεκτά προβλήματα αναφοράς (benchmarks) δεν έχει ελεγχθεί.
- **Σύγκριση:** Δεν πραγματοποιήθηκε άμεση σύγκριση της απόδοσης του προτεινόμενου αλγορίθμου με άλλους state-of-the-art αλγορίθμους για το MDMFVRP από τη βιβλιογραφία.
- **Υπολογιστικός Χρόνος:** Αν και ο αλγόριθμος φάνηκε να εκτελείται σε λογικό χρόνο για τα μεγέθη που δοκιμάστηκαν, δεν έγινε συστηματική μέτρηση ή βελτιστοποίηση του χρόνου εκτέλεσης.
- **Στατικότητα:** Ο αλγόριθμος αντιμετωπίζει το πρόβλημα ως στατικό. Δεν λαμβάνει υπόψη δυναμικές αλλαγές, όπως νέες παραγγελίες που φτάνουν κατά τη διάρκεια της ημέρας ή απρόβλεπτες καθυστερήσεις (π.χ., κίνηση).

Μελλοντική Έρευνα

Με βάση τα αποτελέσματα και τους περιορισμούς της παρούσας εργασίας, προτείνονται οι ακόλουθες κατευθύνσεις για μελλοντική έρευνα:

- **Αξιολόγηση σε Benchmarks:** Εφαρμογή και αξιολόγηση του αλγορίθμου σε καθιερωμένα σύνολα δεδομένων αναφοράς (benchmark instances) για το MDVRP ή/και το VRP με μικτό στόλο, προσαρμόζοντάς τα αν χρειάζεται.
- **Συγκριτική Ανάλυση:** Σύγκριση της απόδοσης (ποιότητα λύσης και υπολογιστικός χρόνος) του υβριδικού αλγορίθμου με άλλες σύγχρονες μεταευρετικές ή ακριβείς μεθόδους για το MDMFVRP.
- **Δυναμικό MDMFVRP:** Επέκταση του αλγορίθμου για την αντιμετώπιση δυναμικών σεναρίων, όπου οι πληροφορίες (π.χ., νέοι πελάτες, χρόνοι ταξιδιού) μπορεί να αλλάζουν κατά την εκτέλεση των δρομολογίων.
- **Πιο Σύνθετοι Περιορισμοί:** Ενσωμάτωση επιπλέον ρεαλιστικών περιορισμών, όπως πολλαπλά ταξίδια ανά όχημα (multi-trip), ταυτόχρονη παραλαβή και παράδοση (pickup and delivery), ή υποχρεωτικά διαλείμματα οδηγών.
- **Στοχαστικότητα:** Διερεύνηση παραλλαγών του προβλήματος με στοχαστικές παραμέτρους, όπως αβέβαιους χρόνους ταξιδιού ή ζήτηση πελατών.
- **Βελτιστοποίηση Παραμέτρων:** Χρήση τεχνικών αυτόματης ρύθμισης παραμέτρων (parameter tuning) για την εύρεση των βέλτιστων ρυθμίσεων του αλγορίθμου ACO για διαφορετικές κλάσεις προβλημάτων.
- **Παραλληλοποίηση:** Διερεύνηση της δυνατότητας παραλληλοποίησης τμημάτων του αλγορίθμου (ιδίως της φάσης ACO και της τοπικής αναζήτησης) για τη μείωση του υπολογιστικού χρόνου σε πολύ μεγάλα προβλήματα.

Η περαιτέρω ανάπτυξη και βελτίωση αλγορίθμων όπως ο προτεινόμενος μπορεί να συμβάλλει σημαντικά στην αποδοτικότερη λειτουργία των σύγχρονων συστημάτων logistics.

ΠΑΡΑΡΤΗΜΑ Α

```
import numpy as np
from sklearn.cluster import KMeans
import random
import matplotlib.pyplot as plt
from tabulate import tabulate
import time


# ================================================
# DATA STRUCTURES
# ================================================


class Depot:
    def __init__(self, id, location, vehicles):
        self.id = id
        self.location = location  # (x, y)
        self.vehicles = vehicles  # List of Vehicle objects
        self.customers = []       # Customers assigned to this depot


class Vehicle:
    def __init__(self, id, v_type, capacity, speed):
        self.id = id
        self.type = v_type        # 'truck', 'motorbike', 'drone'
        self.capacity = capacity  # Max parcels
        self.speed = speed        # Speed units per distance
        self.route = []           # List of customer IDs
        self.current_load = 0
```

```
class Customer:
    def __init__(self, id, location, demand, time_window=None):
        self.id = id
        self.location = location  # (x, y)
        self.demand = demand      # Number of parcels
        self.time_window = time_window  # (earliest_time, latest_time)


# ================================================
# HYBRID ALGORITHM: MDMFVRP
# ================================================


class MDMFVRP:
    def __init__(self, depots, customers, num_ants=10, max_iter=100, alpha=2, beta=1,
evaporation=0.1):
        self.depots = depots
        self.customers = customers
        self.num_ants = num_ants
        self.max_iter = max_iter
        self.alpha = alpha
        self.beta = beta
        self.evaporation = evaporation
        self.pheromone = {}  # Key format: (depot_id, vehicle_id, from_node, to_node)


    # ----------------------------------------
    # Step 1: Assign customers to depots (K-means)
    # ----------------------------------------
    def assign_customers_to_depots(self):
```

```
        depot_locs = np.array([d.location for d in self.depots])
        customer_locs = np.array([c.location for c in self.customers])

        kmeans = KMeans(n_clusters=len(depot_locs), init=depot_locs, n_init=1)
        kmeans.fit(customer_locs)

        for i, label in enumerate(kmeans.labels_):
            self.depots[label].customers.append(self.customers[i])



    # ----------------------------------------
    # Step 2: Assign customers to vehicles (Revised)
    # ----------------------------------------
    def assign_vehicles(self, depot):
        depot.customers.sort(
            key=lambda c: np.linalg.norm(np.array(c.location) - np.array(depot.location))
        )
        remaining_customers = depot.customers.copy()

        # Prioritize larger vehicles first
        for vehicle in sorted(depot.vehicles, key=lambda v: v.capacity, reverse=True):
            while vehicle.current_load < vehicle.capacity and remaining_customers:
                # Find closest customer that fits
                closest = min(
                    remaining_customers,
                    key=lambda c: np.linalg.norm(
                        np.array(c.location) - np.array(depot.location)
                    )
```

```
            )
            if vehicle.current_load + closest.demand <= vehicle.capacity:
                vehicle.route.append(closest.id)
                vehicle.current_load += closest.demand
                remaining_customers.remove(closest)
            else:
                break

    # ----------------------------------------
    # Step 3: ACO Optimization
    # ----------------------------------------
    def aco_optimize(self, vehicle, depot):
        customers = [c for c in depot.customers if c.id in vehicle.route]
        if not customers:
            return

        # Initialize pheromone matrix with depot as node 0
        pheromone_key = (depot.id, vehicle.id)
        if pheromone_key not in self.pheromone:
            size = len(customers) + 1  # +1 for depot
            self.pheromone[pheromone_key] = np.ones((size, size))

        best_route = []
        best_cost = float('inf')

        for _ in range(self.max_iter):
            for ant in range(self.num_ants):
                # Use the time window version if customers have time windows
```

```
                if any(hasattr(c, 'time_window') and c.time_window is not None for c in
customers):
                    route = self._construct_route_with_time_windows(customers, depot,
vehicle)
                else:
                    route = self._construct_route(customers, depot, vehicle)

                cost = self._calculate_cost(route, depot.location, vehicle)

                if cost < best_cost:
                    best_cost = cost
                    best_route = route

            self._update_pheromone(best_route, best_cost, pheromone_key)

        vehicle.route = best_route

    def _construct_route(self, customers, depot, vehicle):
        nodes = [depot.location] + [c.location for c in customers]
        pheromone = self.pheromone[(depot.id, vehicle.id)]
        unvisited = list(range(1, len(nodes)))  # 0 is depot

        route = [0]  # Start at depot
        current = 0

        while unvisited:
            probabilities = []
            for node in unvisited:
```

```
                distance = np.linalg.norm(np.array(nodes[current]) -
np.array(nodes[node]))
                pheromone_level = pheromone[current][node]
                probabilities.append((pheromone_level ** self.alpha) / ((distance + 1e-6) **
self.beta))

            # Roulette wheel selection
            total = sum(probabilities)
            if total == 0:
                break

            probs = [p/total for p in probabilities]
            chosen = np.random.choice(unvisited, p=probs)

            route.append(chosen)
            unvisited.remove(chosen)
            current = chosen

        return [customers[i-1].id for i in route[1:]]  # Exclude depot index

    def _construct_route_with_time_windows(self, customers, depot, vehicle):
        nodes = [depot.location] + [c.location for c in customers]
        pheromone = self.pheromone[(depot.id, vehicle.id)]
        unvisited = list(range(1, len(nodes)))  # 0 is depot

        route = [0]  # Start at depot
        current = 0
        current_time = 0
```

```
        while unvisited:
            feasible_next = []
            for node in unvisited:
                # Calculate arrival time at next customer
                distance = np.linalg.norm(np.array(nodes[current]) -
np.array(nodes[node]))
                travel_time = distance / vehicle.speed
                arrival_time = current_time + travel_time

                # Check time window feasibility
                customer = customers[node-1]
                is_feasible = True

                if customer.time_window is not None:
                    _, latest_time = customer.time_window
                    if arrival_time > latest_time:
                        is_feasible = False

                if is_feasible:
                    feasible_next.append(node)

            if not feasible_next:
                break  # No feasible next customer

            # Choose next customer based on pheromone and distance
            probabilities = []
            for node in feasible_next:
                distance = np.linalg.norm(np.array(nodes[current]) -
np.array(nodes[node]))
```

```
                pheromone_level = pheromone[current][node]

                # Incorporate time window urgency into decision
                customer = customers[node-1]
                if customer.time_window is not None:
                    _, latest_time = customer.time_window
                    time_window_urgency = 1 / max(1, latest_time - (current_time + distance/vehicle.speed))
                else:
                    time_window_urgency = 1

                probabilities.append(
                    (pheromone_level ** self.alpha) *
                    (time_window_urgency ** 0.5) /
                    ((distance + 1e-6) ** self.beta)
                )

            # Roulette wheel selection
            total = sum(probabilities)
            if total == 0:
                break

            probs = [p/total for p in probabilities]
            chosen_idx = np.random.choice(range(len(feasible_next)), p=probs)
            chosen = feasible_next[chosen_idx]

            # Update route and time
            route.append(chosen)
```

```
            unvisited.remove(chosen)

            distance = np.linalg.norm(np.array(nodes[current]) - np.array(nodes[chosen]))
            travel_time = distance / vehicle.speed
            current_time += travel_time

            # Consider service time
            current_time += 5  # Example service time
            current = chosen

        return [customers[i-1].id for i in route[1:]]  # Exclude depot index



    def _calculate_cost(self, route, depot_loc, vehicle):
        if not route:
            return 0

        total_cost = 0
        prev_loc = depot_loc
        current_time = 0  # Start time at depot

        for cust_id in route:
            cust = next(c for c in self.customers if c.id == cust_id)
            distance = np.linalg.norm(np.array(prev_loc) - np.array(cust.location))
            travel_time = distance / vehicle.speed

            # Add travel time to total cost
            total_cost += travel_time
```

```
                current_time += travel_time

                # Add service time (could be vehicle or customer specific)
                service_time = 5  # Example: 5 time units for service
                current_time += service_time

                # If we're factoring in time windows, add waiting time and penalties
                if cust.time_window is not None:
                    early_time, late_time = cust.time_window
                    if current_time < early_time:
                        # Waiting time
                        waiting_time = early_time - current_time
                        current_time = early_time
                    elif current_time > late_time:
                        # Late arrival penalty
                        late_penalty = (current_time - late_time) * 2  # Example: 2x penalty for
lateness
                        total_cost += late_penalty

                prev_loc = cust.location

            # Return to depot
            distance_to_depot = np.linalg.norm(np.array(prev_loc) - np.array(depot_loc))
            total_cost += distance_to_depot / vehicle.speed

            return total_cost

        def _update_pheromone(self, route, cost, pheromone_key):
```

```
        decay = self.evaporation
        pheromone = self.pheromone[pheromone_key]
        pheromone *= (1 - decay)  # Evaporation

        if cost == 0 or not route:
            return

        # Convert customer IDs to node indices (0 = depot)
        route_with_depot = [0] + [i+1 for i, _ in enumerate(route)]

        for i in range(len(route_with_depot)-1):
            from_node = route_with_depot[i]
            to_node = route_with_depot[i+1]
            pheromone[from_node][to_node] += 1 / cost

    # ----------------------------------------
    # Step 4: Local Search Optimization
    # ----------------------------------------
    def local_search_optimization(self):
        """Apply 2-opt local search to improve routes"""
        improvement = True
        iterations = 0
        max_iterations = 100

        while improvement and iterations < max_iterations:
            improvement = False
            iterations += 1
```

```
        for depot in self.depots:
            for vehicle in depot.vehicles:
                if len(vehicle.route) >= 4:  # Need at least 4 nodes for 2-opt to be useful
                    improved = self._apply_2opt(vehicle.route, depot.location, vehicle)
                    if improved:
                        improvement = True

    return iterations


def _apply_2opt(self, route, depot_loc, vehicle):
    """Apply 2-opt local search to a single route"""
    if not route:
        return False

    improved = False
    best_distance = self._calculate_cost(route, depot_loc, vehicle)

    # Try all possible 2-opt swaps
    for i in range(len(route) - 1):
        for j in range(i + 1, len(route)):
            if j - i == 1:
                continue  # Skip adjacent edges

            # Create new route with 2-opt swap
            new_route = route.copy()
            new_route[i+1:j+1] = reversed(new_route[i+1:j+1])

            # Calculate new cost
```

```
                new_distance = self._calculate_cost(new_route, depot_loc, vehicle)

                # If improvement found, update route
                if new_distance < best_distance:
                    route[:] = new_route[:]
                    best_distance = new_distance
                    improved = True

        return improved

    # ----------------------------------------
    # Step 5: Workload Balancing
    # ----------------------------------------
    def balance_workloads(self):
        depot_loads = []
        for depot in self.depots:
            load = sum(len(v.route) for v in depot.vehicles)
            depot_loads.append((depot, load))

        avg_load = sum(load for _, load in depot_loads) / len(depot_loads)

        for depot, load in depot_loads:
            if load > avg_load * 1.2:  # > 20% over average
                candidates = sorted(depot.customers,
                                    key=lambda c: np.linalg.norm(np.array(c.location) -
np.array(depot.location)),
                                    reverse=True)[:3]
```

```
                for customer in candidates:
                    nearest_depot = min(
                        (d for d in self.depots if d != depot),
                        key=lambda d: np.linalg.norm(np.array(customer.location) -
np.array(d.location))
                    )
                    if sum(len(v.route) for v in nearest_depot.vehicles) < avg_load * 0.8:
                        depot.customers.remove(customer)
                        nearest_depot.customers.append(customer)
                        break

        # ----------------------------------------
        # Step 6: Time Window Feasibility Check
        # ----------------------------------------
        def check_time_window_feasibility(self, vehicle, depot):
            """Check if a vehicle route meets time window constraints"""
            route = vehicle.route
            if not route:
                return True

            current_time = 0  # Start time at depot
            prev_loc = depot.location

            for cust_id in route:
                cust = next(c for c in self.customers if c.id == cust_id)

                # Calculate travel time to customer
                distance = np.linalg.norm(np.array(prev_loc) - np.array(cust.location))
```

```
            travel_time = distance / vehicle.speed
            current_time += travel_time

            # Check time window if applicable
            if cust.time_window is not None:
                _, latest_time = cust.time_window
                if current_time > latest_time:
                    return False  # Time window violated

            # Add service time
            current_time += 5  # Example: 5 time units for service
            prev_loc = cust.location

        return True

    # ----------------------------------------
    # Main Algorithm
    # ----------------------------------------
    def solve(self):
        self.assign_customers_to_depots()

        for depot in self.depots:
            self.assign_vehicles(depot)
            for vehicle in depot.vehicles:
                self.aco_optimize(vehicle, depot)

        # Add local search after ACO
        self.local_search_optimization()
```

```
            self.balance_workloads()

            # Final check for time window feasibility
            for depot in self.depots:
                for vehicle in depot.vehicles:
                    is_feasible = self.check_time_window_feasibility(vehicle, depot)
                    if not is_feasible:
                        print(f"Warning: Vehicle {vehicle.id} at Depot {depot.id} has time window violations!")

            return self._get_solution()

        def _get_solution(self):
            return {
                depot.id: {
                    vehicle.id: {
                        'type': vehicle.type,
                        'route': vehicle.route,
                        'load': vehicle.current_load
                    } for vehicle in depot.vehicles
                } for depot in self.depots
            }

    # ================================================
    # TESTING AND EVALUATION FUNCTIONS
    # ================================================
```

```
def validate_solution(test_case, solution):
    """Validate solution and return metrics"""
    metrics = {
        'feasible': True,
        'constraint_violations': [],
        'total_cost': 0,
        'route_balance': 0,
        'capacity_utilization': 0
    }

    # Check capacity constraints
    for depot_id, depot_data in solution.items():
        depot = next(d for d in test_case['depots'] if d.id == depot_id)
        for vid, vehicle_data in depot_data.items():
            vehicle = next(v for v in depot.vehicles if v.id == vid)

            # Check vehicle capacity
            if vehicle_data['load'] > vehicle.capacity:
                metrics['feasible'] = False
                metrics['constraint_violations'].append(
                    f"Vehicle {vid} exceeds capacity"
                )

    # Calculate route costs and balance
    depot_loads = []
    total_capacity = 0
    used_capacity = 0
```

```
for depot_id, depot_data in solution.items():
    depot = next(d for d in test_case['depots'] if d.id == depot_id)
    depot_load = 0

    for vid, vehicle_data in depot_data.items():
        vehicle = next(v for v in depot.vehicles if v.id == vid)
        route = vehicle_data['route']

        # Calculate route cost
        prev_loc = depot.location
        route_cost = 0

        for cust_id in route:
            customer = next(c for c in test_case['customers'] if c.id == cust_id)
            dist = np.linalg.norm(
                np.array(prev_loc) - np.array(customer.location)
            )
            route_cost += dist / vehicle.speed
            prev_loc = customer.location

        metrics['total_cost'] += route_cost
        depot_load += len(route)

        # Track capacity utilization
        total_capacity += vehicle.capacity
        used_capacity += vehicle_data['load']

    depot_loads.append(depot_load)
```

```
        # Calculate workload balance
        if depot_loads:
            avg_load = sum(depot_loads) / len(depot_loads)
            max_imbalance = max(abs(load - avg_load) / avg_load for load in depot_loads) if
avg_load > 0 else 0
            metrics['route_balance'] = 1 - max_imbalance

        # Calculate capacity utilization
        metrics['capacity_utilization'] = used_capacity / total_capacity if total_capacity > 0
else 0

        return metrics

    def visualize_solution(test_case, solution):
        """Visualize the solution with routes and depot assignments"""
        plt.figure(figsize=(12, 8))

        # Plot depots
        depot_x = [d.location[0] for d in test_case['depots']]
        depot_y = [d.location[1] for d in test_case['depots']]
        plt.scatter(depot_x, depot_y, c='red', s=100, marker='s', label='Depots')

        # Plot customers and routes
        colors = plt.cm.rainbow(np.linspace(0, 1, len(test_case['depots'])))

        for depot_idx, (depot_id, depot_data) in enumerate(solution.items()):
            depot = next(d for d in test_case['depots'] if d.id == depot_id)
```

```
            for vid, vehicle_data in depot_data.items():
                vehicle = next(v for v in depot.vehicles if v.id == vid)
                route = vehicle_data['route']
                if not route:
                    continue

                # Plot route
                points = [depot.location]
                for cust_id in route:
                    customer = next(c for c in test_case['customers'] if c.id == cust_id)
                    points.append(customer.location)
                    plt.scatter(customer.location[0], customer.location[1],
                                c=[colors[depot_idx]], s=50, alpha=0.5)

                points.append(depot.location)  # Return to depot
                points = np.array(points)
                plt.plot(points[:, 0], points[:, 1], c=colors[depot_idx],
                         alpha=0.7, linewidth=2, label=f"Vehicle {vid} ({vehicle.type})")

        plt.title('MDMFVRP Solution Visualization')
        plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), ncol=3)
        plt.grid(True)
        plt.tight_layout()
        plt.show()


    def generate_test_case(name, num_depots, num_customers, area_size=100,
with_time_windows=False):
        """Generate a test case with known properties"""
```

```
depots = []
for i in range(num_depots):
    # Create depots with mixed fleet
    vehicles = [
        Vehicle(f"{i}_truck", 'truck', 20, 1),
        Vehicle(f"{i}_bike", 'motorbike', 5, 2),
        Vehicle(f"{i}_drone", 'drone', 1, 4)
    ]
    depot = Depot(i,
                  (random.uniform(0, area_size), random.uniform(0, area_size)),
                  vehicles)
    depots.append(depot)

# Generate clustered customers around depots
customers = []
for i in range(num_customers):
    if random.random() < 0.7:  # 70% customers clustered around depots
        depot = random.choice(depots)
        loc = (
            depot.location[0] + random.uniform(-10, 10),
            depot.location[1] + random.uniform(-10, 10)
        )
    else:  # 30% randomly distributed
        loc = (
            random.uniform(0, area_size),
            random.uniform(0, area_size)
        )
```

```
            # Add time windows if requested
            time_window = None
            if with_time_windows:
                # Generate random time window between 0 and 100
                earliest = random.uniform(0, 50)
                latest = earliest + random.uniform(20, 50)
                time_window = (earliest, latest)

            customers.append(Customer(i, loc, random.randint(1, 3), time_window))

        return {
            'name': name,
            'depots': depots,
            'customers': customers,
        }

    def run_tests(with_time_windows=False):
        """Run a series of test cases"""
        test_cases = [
            generate_test_case("Small Problem", 2, 20,
with_time_windows=with_time_windows),
            generate_test_case("Medium Problem", 3, 50,
with_time_windows=with_time_windows),
            generate_test_case("Large Problem", 5, 100,
with_time_windows=with_time_windows),
        ]

        results = []
        for test_case in test_cases:
```

```
        print(f"\nRunning test case: {test_case['name']}")

        solver = MDMFVRP(test_case['depots'], test_case['customers'])
        solution = solver.solve()

        metrics = validate_solution(test_case, solution)
        results.append({
            'test_case': test_case['name'],
            'metrics': metrics
        })

        print(f"Solution metrics:")
        print(f"- Feasible: {metrics['feasible']}")
        print(f"- Total cost: {metrics['total_cost']:.2f}")
        print(f"- Route balance: {metrics['route_balance']:.2f}")
        print(f"- Capacity utilization: {metrics['capacity_utilization']:.2f}")

        if metrics['constraint_violations']:
            print("Constraint violations:")
            for violation in metrics['constraint_violations']:
                print(f"  - {violation}")

        visualize_solution(test_case, solution)

    return results

def parameter_sensitivity_analysis(test_case, parameter_ranges):
    """Run sensitivity analysis on algorithm parameters"""
```

```
results = {}

# Base parameter set
base_params = {
    'num_ants': 20,
    'max_iter': 100,
    'alpha': 2,  # Pheromone importance
    'beta': 1,   # Distance importance
    'evaporation': 0.1
}

print(f"Running parameter sensitivity analysis...")

# Test each parameter while keeping others constant
for param_name, param_values in parameter_ranges.items():
    param_results = []

    for value in param_values:
        # Create parameter set with current test value
        test_params = base_params.copy()
        test_params[param_name] = value

        # Create solver with test parameters
        solver = MDMFVRP(
            test_case['depots'],
            test_case['customers'],
            num_ants=test_params['num_ants'],
            max_iter=test_params['max_iter'],
```

```
                alpha=test_params['alpha'],
                beta=test_params['beta'],
                evaporation=test_params['evaporation']
            )

            # Solve and get metrics
            solution = solver.solve()
            metrics = validate_solution(test_case, solution)

            param_results.append({
                'value': value,
                'total_cost': metrics['total_cost'],
                'route_balance': metrics['route_balance'],
                'capacity_utilization': metrics['capacity_utilization']
            })

            print(f"  {param_name} = {value}: cost = {metrics['total_cost']:.2f}")

        results[param_name] = param_results

    return results

def run_sensitivity_analysis():
    """Run complete sensitivity analysis"""
    # Create a medium-sized test case
    test_case = generate_test_case("Sensitivity Analysis Test", 3, 50)

    # Define parameter ranges to test
```

```
    parameter_ranges = {
        'num_ants': [5, 10, 20, 30, 50],
        'max_iter': [50, 100, 150, 200, 250],
        'alpha': [1, 1.5, 2, 2.5, 3],
        'beta': [0.5, 1, 1.5, 2, 3],
        'evaporation': [0.05, 0.1, 0.15, 0.2, 0.25]
    }

    # Run analysis
    results = parameter_sensitivity_analysis(test_case, parameter_ranges)

    # Plot results
    plot_sensitivity_analysis(results)

    return results

def plot_sensitivity_analysis(results):
    """Plot the results of parameter sensitivity analysis"""
    fig, axs = plt.subplots(len(results), 3, figsize=(15, 5 * len(results)))

    for i, (param_name, param_results) in enumerate(results.items()):
        # Extract data
        values = [r['value'] for r in param_results]
        costs = [r['total_cost'] for r in param_results]
        balances = [r['route_balance'] for r in param_results]
        utilizations = [r['capacity_utilization'] for r in param_results]

        # Plot total cost
```

```
        axs[i, 0].plot(values, costs, 'o-')
        axs[i, 0].set_title(f'{param_name} vs Total Cost')
        axs[i, 0].set_xlabel(param_name)
        axs[i, 0].set_ylabel('Total Cost')
        axs[i, 0].grid(True)

        # Plot route balance
        axs[i, 1].plot(values, balances, 'o-')
        axs[i, 1].set_title(f'{param_name} vs Route Balance')
        axs[i, 1].set_xlabel(param_name)
        axs[i, 1].set_ylabel('Route Balance')
        axs[i, 1].grid(True)

        # Plot capacity utilization
        axs[i, 2].plot(values, utilizations, 'o-')
        axs[i, 2].set_title(f'{param_name} vs Capacity Utilization')
        axs[i, 2].set_xlabel(param_name)
        axs[i, 2].set_ylabel('Capacity Utilization')
        axs[i, 2].grid(True)

    plt.tight_layout()
    plt.show()

# ================================================
# IMPROVED EXAMPLE USAGE
# ================================================

if __name__ == "__main__":
```

```
# Example 1: Basic problem without time windows
print("Example 1: Basic problem without time windows")
results = run_tests(with_time_windows=False)

# Example 2: Problem with time windows
print("\nExample 2: Problem with time windows")
results_tw = run_tests(with_time_windows=True)

# Example 3: Parameter sensitivity analysis
print("\nExample 3: Parameter sensitivity analysis")
sensitivity_results = run_sensitivity_analysis()
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Ballou, R. H. (2004). Business Logistics/Supply Chain Management. Pearson Prentice Hall.
2. Stodola, P., & Nohel, J. (2023). Adaptive Ant Colony Optimization with Node Clustering for Multi-Depot Vehicle Routing Problem. Mathematics, 11(8), 1903.
3. Panagiotis Oikonomou, Nikos Tziritas, Kostas Kolomvatsos, and Thanasis Loukopoulos. 2022. Fast Heuristics for Mixed Fleet Capacitated Multiple TSP with Applications to Location Based Games and Drone Assisted Transportation. In Proceedings of the 25th Pan-Hellenic Conference on Informatics (PCI '21). Association for Computing Machinery, New York, NY, USA, 294–299.
4. Chopra, S., & Meindl, P. (2016). Supply Chain Management: Strategy, Planning, and Operation. Pearson.
5. Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), 80-91.
6. Laporte, G. (2009). Fifty years of vehicle routing. Transportation Science, 43(4), 408-416.
7. Golden, B. L., Raghavan, S., & Wasil, E. A. (Eds.). (2008). The vehicle routing problem: latest advances and new challenges. Springer.
8. Irnich, S., Toth, P., & Vigo, D. (2014). The family of vehicle routing problems. Vehicle routing: problems, methods, and applications, 1-33.
9. 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.
10. Boyer, K. K., Prud'homme, A. M., & Chung, W. (2009). The last mile challenge: evaluating the effects of customer density and delivery window patterns. Journal of business logistics, 30(1), 185-201.
11. Huber, S., & Geiger, M. J. (2014). Swap body vehicle routing problem: A heuristic solution approach. Computers & Operations Research, 50, 1-14.
12. Cordeau, J. F., & Laporte, G. (2007). The dial-a-ride problem: models and algorithms. Annals of operations research, 153(1), 29-46.
13. Rushton, A., Croucher, P., & Baker, P. (2014). The handbook of logistics and distribution management: understanding the supply chain. Kogan Page Publishers.
14. 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.
15. Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. Networks, 11(2), 221-227.
16. Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. Operations research, 22(2), 340-349.
17. Wren, A., & Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. Journal of the Operational Research Society, 23(3), 333-344.

18. Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks, 30(2), 105-119.
19. Salhi, S., & Sari, M. (1997). A multi-level composite heuristic for the multi-depot vehicle routing problem. European Journal of Operational Research, 103(1), 121-135.
20. Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. Computers & Operations Research, 23(3), 229-235.
21. Crevier, B., Cordeau, J. F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. European journal of operational research, 176(2), 756-773.
22. Toth, P., & Vigo, D. (Eds.). (2014). Vehicle routing: problems, methods, and applications. Society for Industrial and Applied Mathematics.
23. Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.
24. Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research, 35(2), 254-265.
25. Laporte, G., Nobert, Y., & Taillefer, S. (1988). Solving a family of multi-depot vehicle routing and location-routing problems. Transportation science, 22(3), 161-172.
26. Taillard, É. D. (1999). A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Operations Research, 33(1), 1-14.
27. Ho, W., Ho, G. T., Ji, P., & Lau, H. C. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. Engineering applications of artificial intelligence, 21(4), 548-557.
28. Balakrishnan, A., Ward, J. E., & Wong, R. T. (1987). Integrated facility location and vehicle routing models: Recent work and future prospects. American Journal of Mathematical and Management Sciences, 7(1-2), 35-61.
29. Baldacci, R., & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming, 120(2), 347-380.
30. Polacek, M., Hartl, R. F., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. Journal of heuristics, 10(6), 613-627.
31. Thangiah, S. R., & Salhi, S. (2001). Genetic clustering: an adaptive heuristic for the multidepot vehicle routing problem. Applied artificial intelligence, 15(4), 361-383.
32. Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. Computers & Operations Research, 11(1), 49-66.
33. Bettinelli, A., Ceselli, A., & Righini, G. (2011). A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. Transportation Research Part C: Emerging Technologies, 19(5), 723-740.

34. Belfiore, P., & Yoshizaki, H. T. Y. (2009). Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. European Journal of Operational Research, 199(3), 750-758.
35. Gerald G. Brown, Glenn W. Graves, (1981) Real-Time Dispatch of Petroleum Tank Trucks. Management Science 27(1):19-32.
36. Niels Agatz, Paul Bouman, Marie Schmidt (2018) Optimization Approaches for the Traveling Salesman Problem with Drone. Transportation Science 52(4):965-981.
37. Pepin, AS., Desaulniers, G., Hertz, A. et al. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *J Sched* **12**, 17–30 (2009).
38. Kris Braekers, Katrien Ramaekers, Inneke Van Nieuwenhuyse, The vehicle routing problem: State of the art classification and review, Computers & Industrial Engineering, Volume 99, 2016, Pages 300-313