ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Το Συσωρευτικό Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (Cum-CVRPTW) και Χρήση Αλγορίθμου Περιορισμένης Αναζήτησης (Tabu Search)

The Cumulative Vehicle Routing Problem with Time Windows constraints (CUM-CVRPTW) based on the frameworks of Tabu Search

Συγγραφέας:
ΣΕΒΑΣΤΟΠΟΥΛΟΣ ΙΩΑΝΝΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
ΔΡ. ΜΑΡΙΝΑΚΗΣ ΙΩΑΝΝΗΣ

Χανιά 2020

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Μαρινάκη Ιωάννη, για όλη τη βοήθεια του, και την καθοδήγηση του πάνω στην παρούσα διπλωματική εργασία.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Νικόλαο-Αντώνιο Κυριακάκη και την κ. Ανδρομάχη Ταξίδου που ως υποψήφιοι διδάκτορες, ήταν πάντα διαθέσιμοι να λύσουν όλες μου τις απορίες.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου, τους γονείς μου και τα αδέρφια μου, για τους κόπους, τις θυσίες και τη διαρκή και αδιαμαρτύρητη στήριξη τους όλα αυτά τα χρόνια.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή

1.1 Η Εφοδιαστική Αλυσίδα

Στη σύγχρονη κοινωνία όπου τα ακατέργαστα υλικά και ημικατεργασμένα προϊόντα, τα τελικά προϊόντα καθώς και η σχετιζόμενη πληροφορία διακινούνται από και προς κάθε γωνιά του κόσμου, η επιστήμη της επιχειρησιακής έρευνας και η διαχείριση της εφοδιαστικής αλυσίδας παίζουν σημαντικό ρόλο στη μεγιστοποίηση της απόδοσης της εταιρίας. Η επιχείρηση πρέπει να δίνει στον πελάτη το προϊόν που ζητάει, όταν το ζητάει, όσο συχνά το επιθυμεί και σε μία λογική τιμή. Η σωστή διαχείριση της εφοδιαστικής αλυσίδας δημιουργεί ανταγωνιστικό πλεονέκτημα. Η διαχείριση της εφοδιαστικής αλυσίδας ορίζεται ως η διαχείριση ενός δικτύου εσωτερικά συνδεδεμένων επιχειρήσεων που συμμετέχουν στην απώτερη παροχή πακέτων προϊόντων και υπηρεσιών, τα οποία απευθύνονται στους τελικούς καταναλωτές (Harland, 1996) [12]. Η εφοδιαστική αλυσίδα εκτείνεται σε όλη τη διαδικασία μεταφοράς και αποθήκευσης των αγαθών και των πρώτων υλών. Προγραμματίζει και συντονίζει αποτελεσματικά και αποδοτικά τις αναγκαίες εσωτερικές διαδικασίες με σκοπό την παραγωγή και τη διανομή ολοκληρωμένων αγαθών στον τελικό καταναλωτή. Διαχειρίζεται τους προμηθευτές, τα αποθέματα και τα υλικά των αποθηκών, τα κέντρα διανομής, τους μεταφορείς, τα καταστήματα καθώς και τους καταναλωτές.

Τα ενδιάμεσα στάδια διανομής δημιουργούν επιμέρους προβλήματα που η επιστήμη της επιχειρησιακής έρευνας καλείται να επιλύσει κατά το βέλτιστο δυνατό τρόπο. Τέτοια προβλήματα αποτελούν η διαμόρφωση του δικτύου διανομής, η στρατηγική διανομής, η διακίνηση των πληροφοριών μέσα στην αλυσίδα, η διαχείριση αποθεμάτων και οι χρηματικές ροές. Μία εφοδιαστική αλυσίδα οφείλει να συνδυάζει αρμονικά τις αποφάσεις που πρέπει να ληφθούν στα ενδιάμεσα στάδια διανομής.

Τα δίκτυα διανομής επηρεάζουν σε μεγάλο βαθμό τις πωλήσεις της εταιρίας αφού καθορίζουν τη διαθεσιμότητα των προϊόντων. Η διαμόρφωση των δικτύων διανομής χωρίζεται σε επιμέρους προβλήματα. Τα προβλήματα μεταφοράς και χωροθέτησης, χρονοπρογραμματισμού παραγωγής, συσκευασίας, μεταφορών και διανομής και τα προβλήματα καθορισμού διαδρομών ή αλλιώς προβλήματα δρομολόγησης οχημάτων. Κάθε ομάδα προβλημάτων αποτελείται από ένα σύνολο προβλημάτων που διαφοροποιούνται μεταξύ τους σύμφωνα με τους περιορισμούς. Μία εφοδιαστική αλυσίδα πρέπει να συνδυάζει κατάλληλα τις λύσεις των επιμέρους προβλημάτων ώστε να μεγιστοποιηθεί η απόδοση της εταιρίας.

Η εργασία αυτή αφορά το πρόβλημα δρομολόγησης οχημάτων. Στα προβλήματα αυτά έχοντας ένα συγκεκριμένο αριθμό φορτηγών αναζητείται η βέλτιστη διαδρομή που μπορεί να πραγματοποιήσει το εκάστοτε όχημα, με αντικειμενικό στόχο την ελαχιστοποίηση του συνολικού κόστους των διαδρομών. Για να πραγματοποιηθεί η δρομολόγηση οχημάτων πρέπει να ικανοποιούνται ταυτόχρονα ο περιορισμός της ζήτησης του κάθε πελάτη και της χωρητικότητας του οχήματος. Επίσης ένας επιπρόσθετος περιορισμός σε αυτήν την εργασία είναι ότι κάθε πελάτης πρέπει να εξυπηρετηθεί εντός ενός συγκεκριμένου χρονικού διαστήματος, αυτό το χρονικό διάστημα αποτελεί και το χρονικό παράθυρο του κάθε πελάτη. Για το πρόβλημα δρομολόγησης οχημάτων έχουν αναφερθεί στη βιβλιογραφία πολλές παραλλαγές που αφορούν κυρίως στον τρόπο κοστολόγησης μίας λύσης αλλά και στους περιορισμούς

που θα πρέπει να ληφθούν υπόψη. Η εργασία αυτή μελετά και επιλύει μια τέτοια παραλλαγή, το Συσσωρευτικό Πρόβλημα Δρομολόγησης Οχημάτων με τον περιορισμό της χωρητικότητας καθώς και τον περιορισμό των χρονικών παραθύρων. Το πρόβλημα παρουσιάζεται αναλυτικά στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 2: Το Συσσωρευτικό Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (CCVRPTW)

2.1 Το Συσσωρευτικό Πρόβλημα Δρομολόγησης Οχημάτων (CCVRP)

Ένα από τα βασικά προβλήματα της εφοδιαστικής αλυσίδας είναι το πρόβλημα δρομολόγησης οχημάτων. Αναφέρεται στη βιβλιογραφία ένα εύρος παραλλαγών αυτών των προβλημάτων. Ωστόσο δυο είναι τα πιο διαδεδομένα και ευρέως γνωστά. Το πρόβλημα του πλανόδιου πωλητή (TSP) και το πρόβλημα δρομολόγησης οχημάτων (CVRP). Στο πρόβλημα του πλανόδιου πωλητή, το ζητούμενο είναι η εύρεση της συντομότερης διαδρομής, όπου ένα όχημα έχοντας ως αφετηρία έναν κόμβο περνάει από τους υπόλοιπους κόμβους ακριβώς μία φορά και επιστρέφει στον αρχικό. Το πρόβλημα δρομολόγησης οχημάτων αποτελεί μία παραλλαγή του TSP με την προσθήκη του περιορισμού χωρητικότητας, ενώ απαιτείται παραπάνω από ένα όχημα για την εξυπηρέτηση των πελατών. Αρχικά προτάθηκε από τους Dantzig and Ramser (1959) [2] και πάνω σε αυτό βασίζονται τα περισσότερα προβλήματα δρομολόγησης. Σύμφωνα με αυτό η εξυπηρέτηση των πελατών γίνεται από έναν στόλο από οχήματα με στόχο την ελαχιστοποίηση του συνολικού κόστους διαδρομής.

Στη βιβλιογραφία έχουν διατυπωθεί επίσης πολλές παραλλαγές στα προβλήματα δρομολόγησης οχημάτων βασιζόμενες στο CVRP. Γνωστά προβλήματα είναι: το πρόβλημα με χρονικά παράθυρα (VRPTW), το πρόβλημα με παραλαβή και παράδοση (VRPPD), το πρόβλημα με πολλαπλές διαδρομές (VRPMT), το πρόβλημα με πολλαπλές επιστροφές στην αποθήκη (MVRP) και το ανοιχτό πρόβλημα δρομολόγησης (OVRP) [29]. Η επίλυση αυτών των προβλημάτων μπορεί να καθορίσει τη βιωσιμότητα μιας επιχείρησης αφού αυτά συναντώνται καθημερινά.

Το συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων αποτελεί μία παραλλαγή του κλασικού προβλήματος δρομολόγησης οχημάτων το οποίο έχει ως στόχο την ελαχιστοποίηση του συνολικού κόστους μεταφοράς και τη μεγιστοποίηση του κέρδους. Το συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων δίνει έμφαση στην ανάγκη για γρήγορη παροχή κρίσιμων αγαθών σε περίπτωση έκτακτης ανάγκης μετά από κάποια καταστροφή. Για το λόγο αυτό είναι ανθρωποκεντρικό και εστιάζει στην ελαχιστοποίηση των ανθρώπινων απωλειών, σε αντίθεση με τις εμπορικές εφοδιαστικές αλυσίδες που εστιάζουν στην ποιότητα και την κερδοφορία. Οι διαδρομές που δημιουργούνται λαμβάνοντας υπόψη τον στόχο του CCVRP εξυπηρετούν τις κοινότητες σε συντομότερο χρόνο σε σχέση με το απλό CVRP και οι βέλτιστες λύσεις τους διαφέρουν σημαντικά. Στα προβλήματα αυτά δίνεται βαρύτητα στον χρόνο στον οποίο θα εξυπηρετηθεί ο κάθε πελάτης ενώ λαμβάνεται υπόψη στην κοστολόγηση της λύσης η σειρά εξυπηρέτησης του κάθε πελάτη. Αυτό είναι το Συσσωρευτικό Πρόβλημα δρομολόγησης το οποίο αναλύεται στη συνέχεια [29].

Το συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων (Cummulative Capacitated Vehicle Routing Problem (CCVRP)) ανήκει στην κατηγορία των NP-Hard προβλημάτων. Είναι ουσιαστικά μια παραλλαγή του Traveling Repairman Problem (TRP), με την προσθήκη του περιορισμού χωρητικότητας και της ύπαρξης ομοιογενούς στόλου οχημάτων. Στόχος είναι η ελαχιστοποίηση του αθροίσματος των χρόνων άφιξης στους πελάτες αντί του συνολικού κόστους δρομολόγησης [17 , 18 , 23 , 29].

2.2 Το Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (VRPTW)

Είναι γνωστό ότι η φόρτωση και η εκφόρτωση εμπορευμάτων ιδίως σε κεντρικά σημεία πόλεων περιορίζονται συχνά από ειδικές διατάξεις του νόμου σε κάποιες πρωινές ώρες. Επιπρόσθετα, οι περισσότερες επιχειρήσεις θέτουν οι ίδιες ανάλογα με το προϊόν που διακινούν χρονικά παράθυρα μέσα στα οποία δέχονται και παραδίδουν εμπορεύματα [29].

Το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα μπορεί να καθοριστεί ως εξής: ένα σύνολο πελατών είναι διεσπαρμένοι σε μία γεωγραφική περιοχή και πρέπει να εξυπηρετηθούν από συγκεκριμένο αριθμό οχημάτων. Τα οχήματα είναι αρχικά τοποθετημένα στην αποθήκη. Σε κάθε ένα από τα οχήματα ορίζεται ένα φορτίο που πρέπει να διανεμηθεί στους πελάτες. Η εξυπηρέτηση του πελάτη πρέπει να αρχίσει εντός της χρονικής περιόδου που έχει προκαθορίσει (time window – χρονικό παράθυρο ή περιθώριο), και στη συνέχεια να ολοκληρωθεί. Τα οχήματα που εξυπηρετούν τους πελάτες έχουν περιορισμένη χωρητικότητα, και η συνολική ζήτηση των πελατών που έχει ανατεθεί σε κάθε όχημα δεν πρέπει να υπερβαίνει τη χωρητικότητα των οχημάτων. Αντικείμενο του προβλήματος είναι η εύρεση ενός συνόλου διαδρομών, τέτοιο ώστε τα οχήματα να εξυπηρετούν όλους τους πελάτες. Κάθε διαδρομή θα πρέπει να ξεκινάει και να τελειώνει στην αποθήκη, να εξυπηρετεί ένα υποσύνολο πελατών χωρίς το άθροισμα της ζήτησής τους να υπερβαίνει τη χωρητικότητα του οχήματος καθώς επίσης θα πρέπει να τηρούνται οι περιορισμοί των χρονικών παραθύρων. Στόχος είναι η ελαχιστοποίηση του συνολικού μήκους των διαδρομών [8 , 9 , 21 , 25 , 26 , 27 , 29].

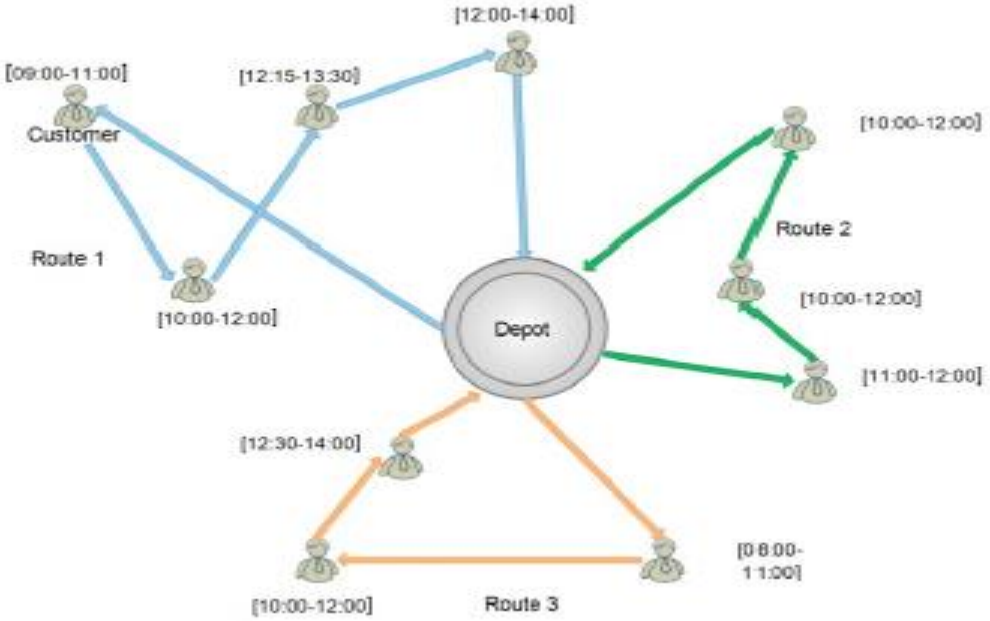
Το πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα αποτελεί μία επέκταση του προβλήματος περιορισμένης χωρητικότητας, στην οποία εξακολουθούν να ισχύουν οι περιορισμοί χωρητικότητας ακριβώς όπως ισχύουν και για το πρόβλημα περιορισμένης χωρητικότητας, με την προσθήκη των χρονικών περιορισμών. Σύμφωνα με αυτούς, κάθε πελάτης ορίζει μία χρονική περίοδο $[\alpha_i, \beta_i]$ μέσα στην οποία πρέπει να αρχίσει η εξυπηρέτηση του. Η χρονική αυτή περίοδος ονομάζεται χρονικό παράθυρο και δίνεται από τα δεδομένα του προβλήματος όπως επίσης και ο χρόνος εξυπηρέτησής s_i του κάθε πελάτη. Ο χρόνος ταξιδιού t_{ij} για κάθε τόξο (i, j) ορίζεται ως η απόσταση d_{ij} των δύο κόμβων. Επιπλέον ως δεδομένα του προβλήματος δίνονται η χρονική στιγμή κατά την οποία τα οχήματα ξεκινούν την εκτέλεση της διαδρομής. Η εξυπηρέτηση του κάθε πελάτη πρέπει να ξεκινήσει μέσα στο χρονικό παράθυρο που έχει ορισθεί και το όχημα πρέπει να παραμείνει στο σημείο αυτό έως ότου ολοκληρωθεί η εξυπηρέτησή του. Σε περίπτωση που το όχημα φτάσει στον πελάτη νωρίτερα από τον προκαθορισμένο χρόνο α_i, τότε το όχημα πρέπει να περιμένει έως την έναρξη του χρονικού παραθύρου [29].

Για να καθορίσουμε τα χρονικά παράθυρα θέτουμε ως χρονική στιγμή μηδέν τη στιγμή που τα οχήματα φεύγουν απ' την αποθήκη. Επιπλέον, ακόμη και αν οι αρχικοί πίνακες είναι συμμετρικοί, λόγω του ότι τα χρονικά παράθυρα απαιτούν έναν πλήρη προσανατολισμό της κάθε διαδρομής, τις περισσότερες φορές το πρόβλημα προτυποποιείται ως μη συμμετρικό πρόβλημα [29].

Τα χρονικά παράθυρα μπορούν να χωριστούν σε δύο κατηγορίες ανάλογα με το πόσο αυστηρά είναι, τα χαλαρά χρονικά παράθυρα και τα σκληρά χρονικά παράθυρα. Στα χαλαρά χρονικά παράθυρα, η εξυπηρέτηση ενός πελάτη μπορεί να ξεκινήσει οποιαδήποτε στιγμή, ακόμη και αν η άφιξη του φορτηγού σε αυτόν δεν είναι εντός χρονικών παραθύρων. Αντίθετα

στα σκληρά χρονικά παράθυρα δεν επιτρέπεται το όχημα να φτάσει στον πελάτη μετά απ' τον αργότερο χρόνο εξυπηρέτησης β_i. Εάν η άφιξη του οχήματος στον πελάτη γίνει νωρίτερα απ' τον ενωρίτερο χρόνο εξυπηρέτησής του α_i, τότε το όχημα θα πρέπει να περιμένει και η εξυπηρέτηση του πελάτη θα γίνει τη χρονική στιγμή α_i [29].

Τα προβλήματα δρομολόγησης οχημάτων με χρονικά παράθυρα ανήκουν στην κατηγορία των NP-Hard προβλημάτων της συνδυαστικής βελτιστοποίησης της επιχειρησιακής έρευνας. Η διπλωματική αυτή αναφέρεται στη δεύτερη κατηγορία των χρονικών παραθύρων, που χρησιμοποιεί τα σκληρά χρονικά παράθυρα.



Σχήμα 2.1 Απεικόνιση VRPTW (Πηγή: https://ieeexplore.ieee.org/abstract/document/8342810/figures#figures)

2.3 Το Συσσωρευτικό Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (CCVRPTW)

2.3.1 Περιγραφή

Το συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα μπορεί να οριστεί ως ο συνδυασμός των δύο προηγούμενων NP-Hard προβλημάτων. Είναι μία επέκταση του συσσωρευτικού προβλήματος δρομολόγησης οχημάτων όπου το κόστος της αντικειμενικής συνάρτησης υπολογίζεται ως το άθροισμα των χρόνων άφιξης σε όλους τους πελάτες με τον επιπλέον περιορισμό ότι οι πελάτες πρέπει να εξυπηρετηθούν εντός του χρονικού περιθωρίου που έχει καθοριστεί, είτε απ' τους πελάτες, είτε επειδή η κίνηση στην περιοχή υπόκειται σε κάποια νομοθεσία, είτε απ' τις ανάγκες της ίδιας της επιχείρησης.

2.3.2 Μορφοποίηση προβλήματος

Το πρόβλημα αναφέρεται σε ένα γράφημα G=(V,A), όπου V = {0} ∪ {1,…,n} ∪ {n+1} είναι το σύνολο των κόμβων και A = {(I,j): i,j ∈ V, i≠j} το σύνολο των τόξων. Το σύνολο N = (1,…,n) είναι το σύνολο των πελατών, ενώ οι κόμβοι 0 και n+1 αντίστοιχα αντιπροσωπεύουν την αποθήκη προέλευσης και προορισμού. Στην πράξη υπάρχει μόνο μία αποθήκη που συμβολίζεται με 0 ή n+1, αναλόγως αν είναι ο αρχικός ή ο τερματικός κόμβος μίας διαδρομής ενός φορτηγού. Ένας στόλος από K φορτηγά αρχικά τοποθετείται στην αποθήκη και είναι διαθέσιμος να εξυπηρετήσει τους πελάτες. Κάθε φορτηγό έχει χωρητικότητα Q και ένα σταθερό κόστος F. Κάθε πελάτης έχει μία μη αρνητική ζήτηση q_i που πρέπει να ικανοποιηθεί από την αποθήκη και για λόγους ευκολίας για τη ζήτηση της αποθήκης ισχύει $q_0 = q_{n+1} = 0$. Ένα χρονικό παράθυρο $[e_i, l_i]$ ορίζεται σε κάθε πελάτη i ∈ N = {1,….,n} και υποδεικνύει ότι η εξυπηρέτηση του κάθε πελάτη πρέπει να αρχίζει ενδιάμεσα του e_i και l_i . Ένα όχημα επιτρέπεται να φτάσει στον πελάτη πριν του ενωρίτερου χρόνου εξυπηρέτησης του, αλλά θα πρέπει να περιμένει για να ξεκινήσει την εξυπηρέτησή του. Κάθε τόξο (i, j) ∈ A ισούται με τον χρόνο μετάβασης $c_{ij.}$ Ο χρόνος εξυπηρέτησης για κάθε πελάτη συμπεριλαμβάνεται στον χρόνο μετάβασης από αυτόν τον κόμβο. Ο στόχος του Cum-CVRPTW είναι να καθορίσει ένα σύνολο ακριβώς K διαδρομών, και να ελαχιστοποιήσει το σταθερό κόστος των οχημάτων καθώς και αυτό των χρόνων μετάβασης σε όλους τους πελάτες υπό τους ακόλουθους περιορισμούς, όπως αναφέρονται στο [29].

1. Κάθε διαδρομή αρχίζει και τελειώνει στην αποθήκη.
2. Κάθε πελάτης εξυπηρετείται μόνο ένα φορτηγό.
3. Η συνολική ζήτηση των πελατών που εξυπηρετούνται από ένα φορτηγό δεν πρέπει να υπερβαίνει τη χωρητικότητα του οχήματος.
4. Η εξυπηρέτηση του κάθε πελάτη πρέπει να αρχίσει και να τελειώσει μέσα στο χρονικό παράθυρο $[e_i, l_i]$, ενώ το φορτηγό θα διαμείνει στην τοποθεσία του πελάτη για χρόνο s_i έως ότου ολοκληρωθεί η εκφόρτωση των εμπορευμάτων.

Στη συνέχεια παρουσιάζεται μία μορφοποίηση τριών δεικτών στην οποία χρησιμοποιούνται δύο μεταβλητές απόφασης [16] :

x_{ijk} είναι μία δυαδική μεταβλητή που ισούται με 1 αν το όχημα επισκέπτεται τον πελάτη j αμέσως μετά τον πελάτη I,

tik που καθορίζει το πότε θα ξεκινήσει η εξυπηρέτηση στον πελάτη i από το όχημα k.

Σύμφωνα με τα παραπάνω το πρόβλημα δρομολόγησης οχημάτων μπορεί να καθοριστεί ως εξής:

$$\text{Minimize}\left(\sum_{i\in N}\sum_{k\in K} t_{ik} + \sum_{j\in N}\sum_{k\in K} F * x_{0jk}\right) (1)$$

Υπό:

$$\sum_{j\in V} x_{ijk} = \sum_{j\in V} x_{jik} \dots \forall i \in N, \forall k \in K \ (2)$$

$$\sum_{k\in K}\sum_{j\in V} x_{jik} = 1 \dots \forall i \in N \ (3)$$

$$\sum_{j\in V} x_{0jk} = 1 \dots \ \forall k \in K \ (4)$$

$$\sum_{j \in V} x_{j,n+1,k} = 1 \ldots \ \forall k \in K \ (5)$$

$$\sum_{i \in N} \sum_{j \in V} q_i x_{ijk} \leq Q \ldots \forall k \in K (6)$$

$$t_{ik} + c_{ij} - \left(1 - x_{ijk}\right) M \leq t_{jk} \ldots \forall i \in V \setminus \{n+1\}, \forall j \in V\{0\}, k \in K \quad (7)$$

$$e_i \leq t_{ik} \leq l_i \ldots \forall i \in V, k \in K \quad (8)$$

$$t_{ik} \geq 0 \ldots \ i \in V, k \in K \quad (9)$$

$$x_{ijk} \in \{0,1\} \ldots i, j \in V, k \in K \quad (10)$$

Η αντικειμενική συνάρτηση στοχεύει στην ελαχιστοποίηση του συνολικού κόστους. Ο περιορισμός (2) δηλώνει ότι όταν το όχημα επισκεφθεί έναν πελάτη, στη συνέχεια πρέπει να αναχωρήσει απ' αυτόν. Ο περιορισμός (3) δηλώνει ότι κάθε πελάτης πρέπει να εξυπηρετηθεί από ένα φορτηγό ακριβώς μία φορά. Οι περιορισμοί (4) και (5) υποδεικνύουν ότι κάθε φορτηγό ξεκινάει και τερματίζει στην αποθήκη. Ο περιορισμός (6) υποδηλώνει τον περιορισμό χωρητικότητας του φορτηγού. Οι περιορισμοί (7) και (8) εγγυώνται την εφικτότητα των διαδρομών με βάση τους χρονικούς περιορισμούς, ενώ οι περιορισμοί (9) και (10) επιβάλλουν περιορισμούς στις μεταβλητές.

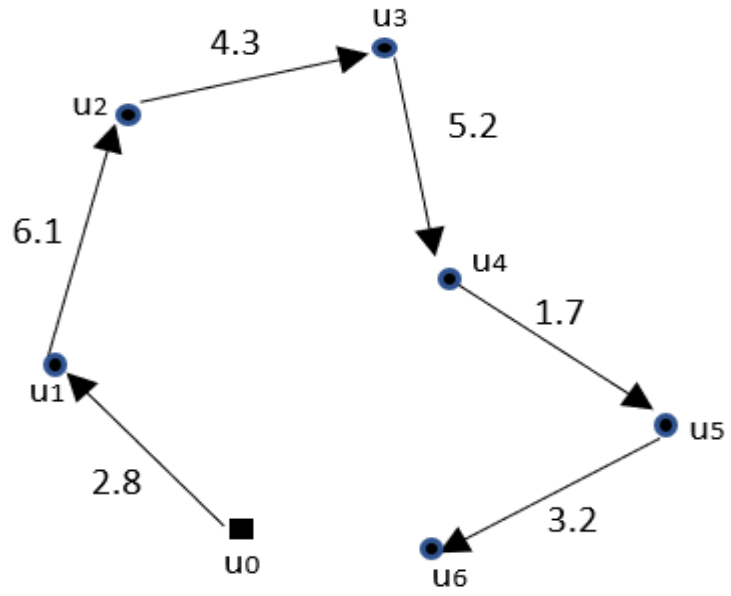
Ακολουθεί ένα παράδειγμα στο οποίο παρουσιάζουμε τον τρόπο κοστολόγησης μίας διαδρομής στο συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων (σχήμα 2.1).

- Έστω μία διαδρομή λύση U=[u0, u1, u2, u3, u4, u5, u6, u0].
- Ο υπολογισμός του κόστους της λύσης μπορεί να περιγραφεί με τον ακόλουθο τύπο:

$$f(x) = \sum_{i=1}^{n} (n - 1 + i) * c[x_{i-1}, x_i] \quad (11)$$

- Άρα σύμφωνα με τα προηγούμενα το τελικό κόστος της διαδρομής θα ισούται με :

$$f(x) = 6(2.8) + 5(6.1) + 4(4.3) + 3(5.2) + 2(1.7) + 1(3.2) = 86.7$$



Σχήμα 2.1: Υπολογισμός κόστους στο CCVRP

ΚΕΦΑΛΑΙΟ 3: Ο Αλγόριθμος Περιορισμένης Αναζήτησης (Tabu Search)

3.1 Αλγόριθμοι

3.1.1 Ευρετικοί Αλγόριθμοι

Η επίλυση ενός προβλήματος συνδυαστικής βελτιστοποίησης γίνεται συνεχώς δυσκολότερη όσο αυξάνεται το μέγεθος του προβλήματος και το πλήθος των μεταβλητών. Πολύ συχνά είναι αδύνατη η εύρεση της βέλτιστης λύσης σε λογικό χρόνο καθώς ο αριθμός των υπολογισμών αυξάνεται εκθετικά σύμφωνα με την αύξηση του μεγέθους του προβλήματος. Για την επίλυση των προβλημάτων αυτής της μορφής μπορούμε να καταφύγουμε σε διαφορετικές τεχνικές οι οποίες οδηγούν το πρόβλημα σε μία καλή λύση η οποία όμως μπορεί να αποκλίνει από τη βέλτιστη. Ως καλή λύση ενός ευρετικού αλγορίθμου μπορεί να ορισθεί μία λύση της οποίας το συνολικό κόστος έχει μικρή διαφορά από τη βέλτιστη. Ένας ευρετικός αλγόριθμος κρίνεται από την ευκολία απόκτησης μιας λύσης, τη λογική πάνω στην οποία στηρίζονται οι κανόνες που χρησιμοποιήθηκαν για να οδηγηθούμε στη λύση, καθώς και από τον χρόνο απόκτησης μιας λύσης. Έχουν αναπτυχθεί πολλοί ευρετικοί αλγόριθμοι οι οποίοι σε μικρό χρονικό διάστημα μπορούν να μας οδηγήσουν σε καλές λύσεις. Ο συνδυασμός αυτών των αλγορίθμων μπορεί να μας οδηγήσει ακόμη και στην βέλτιστη λύση. Οι ευρετικοί αλγόριθμοι κατηγοριοποιούνται ως εξής:

- Αλγόριθμοι απληστίας
- Προσεγγιστικοί αλγόριθμοι
- Αλγόριθμοι τοπικής αναζήτησης

Συνοπτικά μπορούμε να πούμε ότι με τους αλγόριθμους απληστίας μπορούμε να οδηγηθούμε σε μια εφικτή λύση του προβλήματος, σε πολυωνυμικό χρόνο ως προς το μέγεθος των δεδομένων. Γενικά η αρχή της απληστίας είναι η αρχή της ανάπτυξης μυωπικών αλγορίθμων οι οποίοι χρησιμοποιώντας ένα κριτήριο ως προς βελτιστοποίηση σε κάθε βήμα, μπορούν να βρουν μία βέλτιστη λύση τοπικά. Έτσι με βάση αυτό το κριτήριο ό,τι φαίνεται προς στιγμή καλύτερο ως επιλογή, υποθέτουμε ότι είναι και συνολικά η καλύτερη επιλογή. Το πιο συνηθισμένο κριτήριο είναι η απόσταση μεταξύ δύο πελατών. Οι προσεγγιστικοί αλγόριθμοι για να παρακάμψουν αυτό το πρόβλημα χρησιμοποιούν επιπλέον κριτήρια. Τέλος οι αλγόριθμοι τοπικής αναζήτησης κάνουν εξερεύνηση στην γειτονιά της λύσης, δηλαδή σε μία κοντινή στην ήδη υπάρχουσα λύση, προσπαθώντας να την βελτιώσουν [29].

Οι πιο γνωστοί αλγόριθμοι της πρώτης κατηγορίας είναι ο αλγόριθμος του πλησιέστερου γείτονα, ο αλγόριθμος της πλησιέστερης εκχώρησης και ο αλγόριθμος των εξοικονομήσεων (Clarke & Wright, 1964) [1]. Οι αλγόριθμοι αυτοί έχουν τη δυνατότητα να κατασκευάσουν μία εφικτή λύση σε πολύ λίγο χρόνο.

Οι στρατηγικές επίλυσης για το VRP μπορούν να ταξινομηθούν στις παρακάτω προσεγγίσεις :

1. Ομαδοποίηση πρώτα - δρομολόγηση έπειτα (Cluster first – route second).
2. Δρομολόγηση πρώτα – ομαδοποίηση έπειτα (Route first – cluster second).
3. Εξοικονομήσεις / καταχώρηση (Savings /Insertion).

4. Βελτίωση ή ανταλλαγή (Improvement or exchange).
5. Προσέγγιση μαθηματικού προγραμματισμού (Mathematical programming approach).
6. Αλληλοεπιδρούσα βελτίωση (Interactive optimization).
7. Ακριβής διαδικασία (exact procedure).

Για να εφαρμοστεί ένας αλγόριθμος τοπικής αναζήτησης πρέπει πρώτα να έχει δημιουργηθεί μία αρχική λύση είτε με κάποιον απ' τους αλγορίθμους απληστίας, είτε με τη χρήση κάποιας τυχαίας αρχικής λύσης. Οι αλγόριθμοι τοπικής αναζήτησης αναφέρονται στη βιβλιογραφία και ως γειτονιές αναζήτησης. Ο Watters [28], πρότεινε τέσσερις διαδικασίες τοπικής αναζήτησης, που μπορούν να εφαρμοστούν μεταξύ δύο ή περισσότερων διαδρομών, είτε με ανταλλαγή κόμβων είτε με επανατοποθέτηση ενός κόμβου σε μία άλλη διαδρομή. Οι διαδικασίες αυτές έχουν ως στόχο την βελτίωση μέρους της λύσης, με απώτερο σκοπό την βελτίωση του συνολικού κόστους της λύσης. Οι διαδικασίες αυτές είναι οι ακόλουθες :

- Η 1-0 επανατοποθέτηση (1-0 relocate), είναι η διαδικασία διαγραφής ενός κόμβου από μία διαδρομή και η επανατοποθέτησή του σε μία άλλη με καλύτερο συνολικό κόστος.
- Η 1-1 ανταλλαγή (1-1 exchange), σε αυτήν ανταλλάσσονται ένας πελάτης από την μία διαδρομή και ένας πελάτης από τη δεύτερη διαδρομή.
- Η 1-2 ανταλλαγή (1-2 exchange), στην οποία πραγματοποιείται μία συνδυασμένη ανταλλαγή πελατών έτσι ώστε όπως δηλώνει και η ονομασία της, ένας πελάτης από την πρώτη διαδρομή ανταλλάσσεται με δύο διαδοχικούς πελάτες από τη δεύτερη διαδρομή.
- Η 2-2 ανταλλαγή (2-2 exchange), στην οποία δύο γειτονικοί πελάτες μίας διαδρομής ανταλλάσσονται με δύο γειτονικούς πελάτες μίας άλλης διαδρομής.

Πολύ συνηθισμένο είναι επίσης οι διαδικασίες τοπικής αναζήτησης να διενεργούνται εντός της ίδιας διαδρομής. Τέτοιες μέθοδοι είναι οι 2-opt [15], 3-opt [15], Or-opt [20]. Ο αλγόριθμος 3-opt είναι στην ουσία μία επέκταση του αλγορίθμου 2-opt, ενώ ο αλγόριθμος Or-opt είναι μία περιορισμένη έκδοση του 3-opt. Στη μέθοδο 2-opt διαγράφονται δύο τόξα και επανασυνδέονται τα δύο μονοπάτια με διαφορετικό τρόπο ώστε να δημιουργηθεί μία νέα διαδρομή. Αν το κόστος της διαδρομής βελτιώνεται τότε ως τρέχουσα διαδρομή ορίζεται το νέο μονοπάτι που δημιουργήθηκε. Υπάρχει μόνο ένας τρόπος να επανασυνδέσουμε μονοπάτια. Η διαδικασία της μεθόδου είναι η ακόλουθη:

Βήμα 1. Έστω T η τρέχουσα διαδρομή.

Βήμα 2. Για κάθε κόμβο i = 1, . . . , n: Εξετάζουμε όλες τις πιθανές 2-opt κινήσεις που μπορεί να γίνουν από την I και την επόμενή της μέσα στη διαδρομή. Αν με αυτόν τον τρόπο μπορούμε να μειώσουμε το κόστος της διαδρομής, τότε επιλέγουμε την καλύτερη 2-opt κίνηση και εφαρμόζουμε τις αλλαγές στη διαδρομή T.

Βήμα 3. Αν δεν μπορούμε να βρούμε επιπλέον βελτίωση σταματάμε.

3.1.2 Εξελικτικοί και Γενετικοί Αλγόριθμοι

Η ιδέα της μίμησης της διαδικασίας της εξέλιξης στην αναζήτηση για καλύτερες λύσεις οδήγησαν στην ανάπτυξη του πεδίου των Γενετικών Αλγορίθμων. Το πρώτο μαθηματικό

μοντέλο γενετικού αλγορίθμου αναπτύχθηκε από τον John H. Holland [14] τη δεκαετία του 1960, ο οποίος αρχικά εφάρμοσε τις ιδέες του σε απλά καθορισμένα συστήματα περιορισμένων παραμέτρων, ενώ στη συνέχεια επικεντρώνοντας σε συστήματα που έχουν πολλούς παράγοντες που αλληλοεπιδρούν μη γραμμικά μεταξύ τους, έδωσε τη δυνατότητα στους ηλεκτρονικούς υπολογιστές να παράγουν λύσεις σε δύσκολα προβλήματα αναζήτησης και συνδυαστικής βελτιστοποίησης όπως η ελαχιστοποίηση συναρτήσεων και η μηχανική μάθηση [7, 14, 22 , 30].

Οι γενετικοί αλγόριθμοι μιμούνται τη διαδικασία εξέλιξης στη φύση. Βασίζονται στη μίμηση της βιολογικής διαδικασίας στην οποία νέοι και καλύτεροι πληθυσμοί μεταξύ διαφορετικών ειδών αναπτύσσονται κατά τη διάρκεια της εξέλιξης. Έτσι σε αντίθεση με τους ευρετικούς αλγορίθμους, οι γενετικοί κατά τη διάρκεια αναζήτησης μίας καλύτερης λύσης χρησιμοποιούν πληροφορίες από έναν πληθυσμό λύσεων που ονομάζονται άτομα (individuals). Κάθε άτομο στον πληθυσμό αντιπροσωπεύει μία υποψήφια λύση. Σε κάθε επανάληψη, ή αλλιώς γενιά (generation), ενός γενετικού αλγορίθμου διατηρείται το μέγεθος του πληθυσμού σταθερό. Η βασική λειτουργία των γενετικών αλγορίθμων είναι το ταίριασμα δύο λύσεων με στόχο να δημιουργηθεί μία νέα λύση. Για να δημιουργηθεί μία νέα λύση εφαρμόζονται δύο τελεστές, ένας δυαδικός τελεστής που ονομάζεται διασταύρωση (crossover), και ένας μοναδιαίος τελεστής που ονομάζεται μετάλλαξη (mutation). Η διασταύρωση ανταλλάσσει τμήματα από δύο άτομα που ονομάζονται γονείς (parents) και παράγει δύο νέα άτομα που ονομάζονται απόγονοι (offspring). Για τη διαδικασία εξέλιξης υπάρχουν δύο ειδών μοντέλα. Αυτά που χρησιμοποιούν διαφορικές εξισώσεις για τη δυναμική επιλογή και αυτά που χρησιμοποιούν «χειριστές εξέλιξης», διασταύρωση (crossover), μετάλλαξη (mutation) και επιλογή (selection), που επενεργούν στα χαρακτηριστικά. Μερικά από τα πλεονεκτήματα των γενετικών αλγορίθμων είναι [13]:

- Μπορούν να λύσουν εξίσου αποτελεσματικά ένα πρόβλημα που οι μεταβλητές παίρνουν είτε συνεχείς είτε διακριτές τιμές.
- Δεν χρειάζονται πληροφορίες από παραγώγους.
- Μπορούν να κάνουν ταυτόχρονη αναζήτηση σε ένα πολύ μεγάλο μέρος του χώρου των λύσεων.
- Μπορούν να λύσουν γρήγορα το πρόβλημα ανεξάρτητα από το μέγεθός του.
- Μπορούν να εφαρμοστούν σε παράλληλα υπολογιστικά συστήματα.
- Μπορούν εύκολα να ξεφύγουν από κάποιο τοπικό ελάχιστο.
- Μπορούν να δώσουν παραπάνω από μία τοπικά βέλτιστες λύσεις και όχι μόνο μία λύση.

Οι βασικοί παράγοντες κάθε γενετικού αλλά και γενικότερα κάθε εξελικτικού αλγορίθμου είναι:

1. Η κωδικοποίηση (encoding) των λύσεων έτσι ώστε οι λύσεις να αντιστοιχούν σε κάποιο χρωμόσωμα ή χρωμοσώματα.
2. Μία συνάρτηση που εκτιμά την ποιότητα-καταλληλόλητα (fitness) του κάθε ενός από τα άτομα του πληθυσμού.
3. Αρχικοποίηση του πληθυσμού.
4. Επιλογή τελεστών.
5. Τελεστές αναπαραγωγής.

3.1.3 **Νοημοσύνη Σμήνους και Αλγόριθμοι Εμπνευσμένοι από τη Φύση**

Σε αυτήν την κατηγορία ανήκουν αλγόριθμοι που στηρίζονται στη μοντελοποίηση συστημάτων και βασίζονται στη συνεργασία μεταξύ ατόμων από έναν πληθυσμό, στην περίπτωση που τα άτομα του πληθυσμού δεν έχουν πλήρη γνώση του προς επίλυση προβλήματος. Το κάθε μέλος του πληθυσμού ενεργεί με βάση τις πληροφορίες που έχει και στη συνέχεια μοιράζεται τη γνώση που απέκτησε με τα άλλα μέλη του πληθυσμού. Το σύνολο του πληθυσμού ονομάζεται σμήνος (swarm) και όλες αυτές οι μέθοδοι ονομάζονται αλγόριθμοι νοημοσύνης σμήνους (swarm intelligence). Οι πιο γνωστοί αλγόριθμοι είναι:

- Ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization).
- Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization).
- Αλγόριθμοι που βασίζονται σε συμπεριφορές μελισσών.
- Αλγόριθμοι τεχνητών ανοσοποιητικών συστημάτων (Artificial Immune Systems).
- Ο αλγόριθμος της πυγολαμπίδας (Fire-Fly Algorithm).

3.1.4 **Μεθευρετικοί Αλγόριθμοι**

Οι μεθευρετικοί αλγόριθμοι είναι μέθοδοι επίλυσης που χρησιμοποιούν τις διαδικασίες τοπικής αναζήτησης με μία στρατηγική ικανή να ξεφύγει από κάποιο τοπικό ελάχιστο, στο οποίο μπορεί να έχει συγκλίνει ή παγιδευτεί η λύση. Για να αντιμετωπισθεί η ευαισθησία που έχουν οι αλγόριθμοι τοπικής αναζήτησης στην αρχική λύση, έχουν προταθεί στη βιβλιογραφία πολλοί μεθευρετικοί αλγόριθμοι που κάνουν αναζήτηση στον χώρο τον λύσεων έτσι ώστε η λύση να απεγκλωβιστεί από ένα τοπικό ελάχιστο στο οποίο μπορεί να έχει παγιδευτεί.

Ένας πολύ καλός διαχωρισμός των μεθευρετικών αλγορίθμων είναι ανάλογα με το αν χρησιμοποιούν μία ή περισσότερες λύσεις. Η πρώτη κατηγορία αποτελείται από τους αλγορίθμους στους οποίους χρησιμοποιείται μία μόνο λύση και γίνεται αναζήτηση στη γειτονιά της. Αυτοί οι αλγόριθμοι έχουν ισχυρές δυνατότητες εκμετάλλευσης ή εντατικοποίησης (exploitation ή intensification) της περιοχής γύρω από τη λύση. Στη δεύτερη κατηγορία στην οποία χρησιμοποιείται ένας πληθυσμός λύσεων, γίνεται η αναζήτηση σε όλο τον χώρο των λύσεων. Έτσι αυτοί οι αλγόριθμοι έχουν ισχυρές δυνατότητες εξερεύνησης ή διάχυσης (exploration ή diversification) σε μεγάλο μέρος του χώρου των λύσεων. Επίσης έχουν προταθεί στρατηγικές που συνδυάζουν τα παραπάνω.

Οι μεθευρετικοί αλγόριθμοι μπορούν να κατηγοριοποιηθούν σύμφωνα με τη στρατηγική που χρησιμοποιείται για να ξεφύγει η λύση από κάποιο τοπικό ελάχιστο [30].

- Επαναληπτικές διαδικασίες που αρχίζουν από διαφορετικές αρχικές λύσεις. Χαρακτηριστικοί αλγόριθμοι σε αυτήν την κατηγορία είναι οι αλγόριθμοι πολυεναρκτήριας τοπικής αναζήτησης (multi-start local search), της επαναληπτικής τοπικής αναζήτησης (iterated local search) και η μέθοδος της διαδικασίας της άπληστης τυχαιοποιημένης αναζήτησης (Greedy Randomized Adaptive Search Procedure (GRASP)).
- Αλγόριθμοι στους οποίους μία κίνηση που χειροτερεύει τη λύση μπορεί να γίνει αποδεκτή υπό κάποιες συνθήκες. Με αυτόν τον τρόπο γίνεται αναζήτηση σε μεγαλύτερο εύρος λύσεων και έτσι μπορούμε να ξεφύγουμε από ένα τοπικό ελάχιστο και να οδηγηθούμε σε μία καλύτερη λύση απ' την προηγούμενη. Οι πιο χαρακτηριστικοί αλγόριθμοι σε αυτήν την κατηγορία είναι η προσομοιωμένη ανόπτηση (simulated

annealing), ο οποίος είναι και ο πρώτος μεθευρετικός αλγόριθμος που εμφανίστηκε, και ο αλγόριθμος της περιορισμένης αναζήτησης (tabu search), στον οποίο θα αναφερθούμε εκτενώς στη συνέχεια.

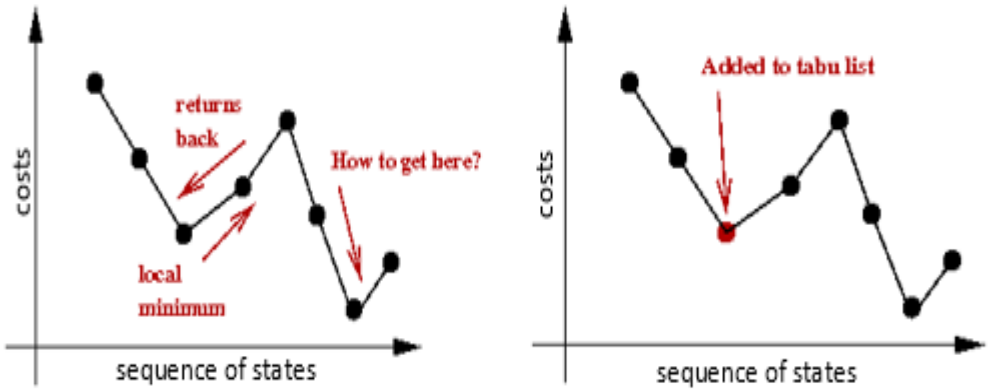
- Αλγόριθμοι που αλλάζουν τη γειτονιά αναζήτησης. Αυτή η κατηγορία αλγορίθμων αποτελείται από αλγόριθμους οι οποίοι όταν κολλήσουν σε κάποιο τοπικό ελάχιστο αλλάζουν τον αλγόριθμο που χρησιμοποιούν σε γειτονικά σημεία του χώρου λύσεων. Οι πιο χαρακτηριστικές μέθοδοι αυτής της κατηγορίας είναι ο αλγόριθμος μεταβλητής γειτονιάς αναζήτησης (Variable Neighborhood Search (VNS)) και ο αλγόριθμος επέκτασης της γειτονιάς αναζήτησης (Expanding Neighborhood Search (ENS)).
- Αλγόριθμοι που αλλάζουν την αντικειμενική συνάρτηση ή κάποια από τα δεδομένα του προβλήματος. Ο πιο χαρακτηριστικός αλγόριθμος αυτής της κατηγορίας είναι αυτός της καθοδηγούμενης τοπικής αναζήτησης (Guided Local Search).

3.2 Περιορισμένη Αναζήτηση (Tabu Search)

3.2.1 **Περιγραφή**

Ο μεθευρετικός αλγόριθμος της Περιορισμένης Αναζήτησης (Tabu Search – (TS)), προτάθηκε αρχικά από τον Glover [3 , 4]. Ο αλγόριθμος χρησιμοποιεί τους αλγορίθμους τοπικής αναζήτησης για να μετακινηθεί από μία λύση σε μία άλλη. Για να ξεφύγει ο αλγόριθμος από ένα πιθανό τοπικό ελάχιστο, μπορεί να αποδεχθεί κινήσεις που χειροτερεύουν τη λύση. Για να αποφευχθούν οι επαναλαμβανόμενες επιστροφές σε καλές λύσεις, χρησιμοποιείται μνήμη από τις προηγούμενες κινήσεις που πραγματοποιήθηκαν. Αυτή η διαδικασία ονομάζεται μνήμη μικρής περιόδου (short term memory) και με αυτόν τον τρόπο ο αλγόριθμος ¨θυμάται¨ τις τελευταίες κινήσεις που πραγματοποίησε. Οι τελευταίες κινήσεις καταγράφονται σε μία λίστα, η οποία ονομάζεται λίστα περιορισμένων κινήσεων (Tabu List). Οι συγκεκριμένες κινήσεις απαγορεύουν στον αλγόριθμο να επιστρέψει στην ίδια λύση για ένα συγκεκριμένο αριθμό επαναλήψεων. Αυτές οι απαγορευμένες κατευθύνσεις που έκανε ονομάζονται περιορισμένες κινήσεις (Tabu Moves) ενώ σαν k ορίζεται το μέγεθος της λίστας περιορισμένης αναζήτησης. Η τιμή του k μπορεί είτε να προσδιοριστεί απ’ την αρχή είτε έχει τη μορφή μεταβλητής που εξαρτάται από τις ιδιαιτερότητες του προβλήματος. Το k επίσης μπορεί να επιλεχθεί τυχαία σε ένα συγκεκριμένο διάστημα ή ακόμα και να αλλάζει δυναμικά κατά το πέρας των επαναλήψεων.

Το μέγεθος της λίστας είναι ένας πολύ σημαντικός παράγοντας επιτυχίας της μεθόδου. Μία μεγάλη τιμή στη μεταβλητή k μπορεί να απαγορεύσει την επίσκεψη μεγάλου αριθμού λύσεων, ενώ μία μικρή τιμή θα περιορίσει την αναζήτηση σε μία πολύ μικρή περιοχή. Κάθε φορά που βελτιώνεται η λύση προστίθεται στη λίστα η τελευταία κίνηση που πραγματοποιήθηκε, ενώ αν το μέγεθος της λίστας είναι μεγαλύτερο του k τότε μία λύση αφαιρείται από τη λίστα σύμφωνα με το πρωτόκολλο FIFO (First In First Out). Με αυτήν την λίστα ο αλγόριθμος μπορεί να αποφύγει να εξετάσει λύσεις που έχει ήδη επισκεφθεί (cycling).



Σχήμα 3.2.1 Μνήμη μικρής περιόδου
(Πηγή:https://www2.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html)

Μερικές φορές κάποιες κινήσεις που κανονικά θα έπρεπε να απαγορεύονται λόγω του ότι βρίσκονται στην tabu list, επιτρέπονται αν η κίνηση αυτή έχει ένα καλύτερο αποτέλεσμα στη συνάρτηση κόστους. Η διαδικασία αυτή ονομάζεται κριτήριο απενεργοποίησης των περιορισμών (aspiration criterion).

Σημαντικό ρόλο στην επιτυχία της μεθόδου παίζει η μνήμη μεσαίας περιόδου (medium term memory). Σε αυτήν τη μνήμη καταγράφονται οι έως τώρα καλύτερες λύσεις που έχουν βρεθεί. Εκμεταλλευόμενοι τη μνήμη μεσαίας περιόδου μπορούμε να χρησιμοποιήσουμε τις στρατηγικές εντατικοποίησης της μεθόδου (intensification strategies). Σύμφωνα με αυτές οι μεταβλητές που βρίσκονται σταθερά μετά από κάποιες κινήσεις μέσα στη λύση ή που εμφανίζονται πολύ συχνά πρέπει να παραμείνουν μέσα στη λύση. Η λογική που ακολουθεί αυτή η στρατηγική είναι ότι η βέλτιστη λύση δεν θα απέχει πολύ από το πιο υποσχόμενο σημείο και είναι πιθανόν να βρίσκεται κοντά στη γειτονιά της έως τώρα καλύτερης λύσης. Έτσι αποθηκεύονται στοιχεία από εκλεκτές λύσεις και χρησιμοποιούνται όταν ενεργοποιηθεί η στρατηγική.

Για να μην εγκλωβιστεί η αναζήτηση σε ένα απλώς υποσχόμενο σημείο χρησιμοποιείται η στρατηγική διάχυσης της αναζήτησης (diversification). Η στρατηγική διάχυσης βασιζόμενη συνήθως στη μνήμη μακράς διάρκειας (long term memory), δημιουργεί νέες λύσεις που θα πρέπει να εξερευνήσει ο αλγόριθμος. Στη μνήμη μακράς διάρκειας καταγράφεται το σύνολο των λύσεων καθώς επίσης και παράγοντες από τις περιοχές που έχουν εξερευνηθεί και από τις μεταβλητές που βρίσκονται τώρα στην τρέχουσα λύση ή που έχουν βρεθεί κατά τη διάρκεια των επαναλήψεων. Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου [29 , 30].

Αλγόριθμος Περιορισμένη Αναζήτηση (Tabu Search(TS))

Αρχικοποίηση

Κατασκευή μιας αρχικής λύσης S_0

Υπολογισμός της συνάρτησης κόστους της λύσης

$S^* = S_0$! Αρχικοποίηση της βέλτιστης λύσης

$f(S^*) = f(S_0)$

Κύρια Φάση

Do while κάποιο κριτήριο τερματισμού δεν έχει ικανοποιηθεί

Υπολογισμός μίας γειτονικής λύσης S'

If $f(S') < f(s^*)$ then

$S^* = S'$

$f^* = f(s')$

end if

Αποθήκευσε την τελευταία λύση στη λίστα των τελευταίων

υποψηφίων (ταυτόχρονα αν έχει συμπληρωθεί το μέγεθος της λίστα διέγραψε την

παλαιότερη εγγραφή)

Κάλεσε κάθε k1 επαναλήψεις τη στρατηγική εντατικοποίησης

If $f(S_{intensification}) < f(s^*)$ then

$S^* = S_{intensification}$

$f^* = f(S_{intensification})$

end if

Κάλεσε κάθε k2 επαναλήψεις τη στρατηγική διαφοροποίησης

If $f(S_{diversification}) < f(s^*)$ then

$S^* = S_{diversification}$

$f^* = f(S_{diversification})$

end if

End do

Επέστρεψε τη βέλτιστη λύση S^*.

3.2.2 Εφαρμογές Περιορισμένης Αναζήτησης

Ο αλγόριθμος περιορισμένης αναζήτησης (TSA) θεωρείται απ' τους ισχυρότερους μεθευρετικούς αλγορίθμους. Εφαρμόζεται σε πολλά προβλήματα συνδυαστικής βελτιστοποίησης και έχει επεκταθεί στην επίλυση δυναμικών προβλημάτων, στοχαστικών προβλημάτων και προβλημάτων όπου χρησιμοποιούνται συμπληρωματικά. Έχει σημαντικά πλεονεκτήματα σε σχέση με άλλους μεθευρετικούς αλγορίθμους, αφού για να ξεφύγει η λύση από ένα τοπικό ελάχιστο χρησιμοποιείται μία συγκεκριμένη στρατηγική (μνήμη) και δεν γίνεται τυχαία επιλογή της επόμενης λύσης. Από τις αρχές του 1990 ο TSA είναι πολύ δημοφιλής στην επίλυση προβλημάτων βελτιστοποίησης και πλέον έχει μία τεράστια γκάμα εφαρμογών σύμφωνα με το [5].

- Το πρόβλημα του πλανόδιου πωλητή, Traveling Salesman Problem (TSP).
- Το πρόβλημα του σακιδίου, The Knapsack Problem.
- Προγραμματισμός εργασιών με περιορισμούς πόρων, Resource-Constrained Project Scheduling Problem (RCPSP).
- Γενικευμένες απαιτήσεις χωρητικότητας στην παραγωγή, Generalized Capacity Requirements in Production.
- Προβλήματα Δρομολόγησης Οχημάτων, Vehicle Routing Problem (VRP).
- Συστήματα Βάσεων Δεδομένων, Database Systems.
- Προβλήματα Ικανοποίησης Περιορισμού, Constraint Satisfaction Problems (SAT).
- Προγραμματισμός σε Συστήματα Παραγωγής, Scheduling in Manufacturing Systems.
- Προγραμματισμός σε Γραμμή Ροής σε Βιομηχανία, Scheduling a Flow-Line Manufacturing Cell.
- Προγραμματισμός εργασιών πολλαπλών επεξεργαστών σε παράλληλα προγράμματα, Multiprocessor Task Scheduling in Parallel Programs.
- Σχεδιασμός δικτύου, Network Design.
- Χρωματισμός Γραφήματος, Graph Coloring.
- Πρόβλημα Διάταξης Ανάθεσης, Retrieval Layout Problem.
- Νευρωνικά Δίκτυα και Εκμάθηση, Neural Networks and Learning.
- Δίκτυα τηλεπικοινωνιών, Telecommunication Network.
- Ολική βελτιστοποίηση, Global Optimization.
- Βελτιστοποίηση Ηλεκτρονικής Δομής,Optimization of Electronic Structure.
- Συνεχείς και Στοχαστική Βελτιστοποίηση, Continuous and Stochastic Optimization.

Στο πλαίσιο των κανόνων του αλγορίθμου Περιορισμένης Αναζήτησης σε αυτήν την εργασία χρησιμοποιήθηκε μία υβριδική μορφή του αλγορίθμου, στην οποία συνδυάζεται με τον αλγόριθμο μεταβλητής γειτονιάς αναζήτησης (Variable Neighborhood Search (VNS)) [10 , 11], όπως θα περιγραφεί αναλυτικά στη συνέχεια.

3.3 Επιπλέον Αλγόριθμοι που Χρησιμοποιήθηκαν

3.3.1 Ο Αλγόριθμος του Πλησιέστερου Γείτονα (NN)

Για τη δημιουργία αρχικής λύσης χρησιμοποιήθηκε ο αλγόριθμος του πλησιέστερου γείτονα. Σε αυτόν τον αλγόριθμο η διαδρομή του οχήματος έχει ως αφετηρία την αποθήκη και έπειτα επισκέπτεται τον πλησιέστερο πελάτη σε αυτήν. Από εκεί πηγαίνει στον κοντινότερο πελάτη που δεν έχει ως τώρα επισκεφθεί. Έτσι εξελίσσεται η διαδικασία του αλγορίθμου μέχρι να επισκεφθεί όλους τους πελάτες ή να καλυφθεί η διαθέσιμη χωρητικότητα του

οχήματος. Έτσι με βάση τον αλγόριθμο αυτόν και λαμβάνοντας υπόψη τους περιορισμούς του προβλήματος, δημιουργείται μία αρχική εφικτή λύση.

Ο πλησιέστερος γείτονας είναι ένας απλός αποτελεσματικός αλγόριθμος που ανήκει στην ομάδα των ευρετικών αλγορίθμων. Η διαδικασία της μεθόδου αυτής είναι η ακόλουθη σύμφωνα με το άρθρο [24].

Βήμα 1. Ξεκινάμε από οποιονδήποτε κόμβο σαν αφετηρία της διαδρομής.

Βήμα 2. Βρίσκουμε τον πλησιέστερο κόμβο στον τελευταίο κόμβο που προστέθηκε στο μονοπάτι και τον προσθέτουμε.

Βήμα 3. Επαναλαμβάνουμε το βήμα 2, μέχρι όλες οι κορυφές να βρίσκονται στο μονοπάτι το οποίο σημαίνει και την ολοκλήρωση της διαδικασίας.

3.3.2 **Αλγόριθμος Επανασύνδεσης Διαδρομών, Path Relinking (PR)**

Ο αλγόριθμος επανασύνδεσης διαδρομών χρησιμοποιείται συνήθως ως στρατηγική εντατικοποίησης και δημιουργεί καινούριες λύσεις εξετάζοντας διαφορετικές τροχιές. Ο αλγόριθμος χρησιμοποιεί δύο λύσεις, την εναρκτήρια λύση (starting solution) και τη λύση στόχος (target solution) [6]. Τις περισσότερες φορές ως αρχική λύση χρησιμοποιείται η καλύτερη λύση, ενώ η χειρότερη λύση χρησιμοποιείται ως λύση στόχος. Οι ρόλοι των δύο λύσεων μπορούν να αλλάξουν. Σημαντικό πλεονέκτημα του αλγορίθμου επανασύνδεσης διαδρομών είναι ότι έχει τη δυνατότητα να προσαρμοσθεί σε οποιαδήποτε μέθοδο παράγει περισσότερες από δύο λύσεις. Έτσι μπορεί να χρησιμοποιηθεί και στον (TSA) αφού στη μνήμη μεσαίας περιόδου αποθηκεύει εκλεκτές λύσεις [30].

Η ιδέα είναι ότι πάντα ανάμεσα σε δύο καλές λύσεις υπάρχει μία καλύτερη λύση. Μία ακολουθία από γειτονικές λύσεις δημιουργούνται και από αυτές ο αλγόριθμος επιστρέφει την καλύτερη. Ο αλγόριθμος τερματίζεται όταν η αρχική λύση και λύση στόχος είναι ίδιες. Υπάρχουν πολλοί τρόποι για να επιλέξουμε την εναρκτήρια λύση καθώς και τη λύση στόχος. Οι πιο γνωστοί είναι [29 , 30]:

- Προς τα εμπρός επανασύνδεση διαδρομών (forward relinking): η λύση στόχος είναι καλύτερη από την εναρκτήρια λύση
- Προς τα πίσω επανασύνδεση διαδρομών (backward relinking): η λύση στόχος είναι χειρότερη από την εναρκτήρια λύση. Έχει αποδειχθεί πειραματικά ότι αυτή η μέθοδος δίνει καλύτερα αποτελέσματα γιατί εξερευνάει τη γειτονιά αναζήτησης γύρο απ' την καλύτερη λύση.
- Προς τα πίσω και προς τα εμπρός επανασύνδεση (back and forward relinking): σε αυτήν την περίπτωση και οι δύο διαδρομές παράγονται ταυτόχρονα. Αυτό αυξάνει τον υπολογιστικό φόρτο χωρίς να σημαίνει ότι θα οδηγηθεί σε καλύτερα αποτελέσματα.
- Ανάμεικτη επανασύνδεση διαδρομών (mixed relinking): οι δύο διαδρομές παράγουν μία ενδιάμεση λύση η οποία ισαπέχει από τις δύο λύσεις.
- Τυχαιοποιημένη επανασύνδεση διαδρομών (randomized relinking): εδώ επιλέγονται λύσεις απ' τη μνήμη με τις καλύτερες λύσεις.

- Περικοπτώμενη επανασύνδεση διαδρομών (truncated relinking): εξετάζεται ένα μέρος της διαδρομής από τη μία λύση στην άλλη.

Στη συνεχεία παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου.

Αλγόριθμος Επανασύνδεση Διαδρομών

Αρχικοποίηση

Επέλεγε Αρχική λύση s

Επέλεξε Τελική λύση t

x=s Υπολόγισε την απόσταση dist (x,t)

repeat

Βρες την καλύτερη κίνηση m που μειώνει την απόσταση (x,t)

Εφάρμοσε τη κίνηση m στη λύση x

Υπολόγισε την απόσταση dist (x,t)

Until dist(x, t) ≠ 0

Επέστρεψε τη βέλτιστη λύση στην τροχιά των s και t.

3.3.3 Αλγόριθμος Μεταβλητής Γειτονιάς Αναζήτησης (Variable Neighborhood Search (VNS))

Ο αλγόριθμος μεταβλητής γειτονιάς αναζήτησης (Variable Neighborhood Search (VNS)) προτάθηκε από τους Hansen και Mladenovic [10, 11]. Η βασική ιδέα της μεθόδου είναι η επιτυχής αναζήτηση ενός συνόλου από γείτονες για να βρεθεί μία καλύτερη λύση. Η αναζήτηση αυτή πραγματοποιείται είτε με τυχαίο τρόπο είτε με πιο συστηματικό ώστε να ξεφύγει η αναζήτηση από κάποιο τοπικό ελάχιστο. Για παράδειγμα μπορούν να χρησιμοποιηθούν οι μέθοδοι 1-1 ανταλλαγή, 1-0 επανατοποθέτηση, και 2-opt. Ο αλγόριθμος σταματάει όταν καμία μέθοδος δεν μπορεί να βελτιώσει άλλο τη λύση σε μία επανάληψη. Αυτή η μέθοδος εκμεταλλεύεται την ιδιότητα των διαφόρων μεθόδων τοπικής αναζήτησης οι οποίες μπορούν να οδηγήσουν τη λύση σε διαφορετικά τοπικά βέλτιστα.

Ο αλγόριθμος VNS βασίζεται στον αλγόριθμο Μεταβλητής Γειτονιάς Καθόδου (Variable Neighborhood Descent (VND)). Ο αλγόριθμος VND χρησιμοποιεί διαδοχικές γειτονιές αναζήτησης σε κατάβαση προς ένα τοπικό ελάχιστο. Αρχικά ορίζεται το σύνολο των γειτονιών (N_l, όπου l= 1, . . . , l_{max}). Εάν δεν υπάρχει κάποια βελτίωση στη λύση της s σε κάποια επανάληψη της N_l τότε η γειτονιά αλλάζει και γίνεται N_{l+1}.

ΚΕΦΑΛΑΙΟ 4: Εφαρμογή και Επίλυση

4.1 Συνάρτηση Τιμωρίας

Για την αποτελεσματικότερη επίλυση του προβλήματος χρησιμοποιήθηκε μία συνάρτηση τιμωρίας. Αυτή η συνάρτηση ορίστηκε ως:

$$J = cost + a_1 * \sum_{j=0}^{k} \max\left(load_j - capacity_j, 0\right) + \sum_{j=0}^{k}\sum_{i=0}^{n} a_2 * timeFlag_{ij} \quad (4.1)$$

Όπου, $\text{timeFlag}_{ij} = \begin{cases} 0, & \text{αν το φορτηγό } (j) \text{ φτάνει στον πελατη έγκαιρα(i)} \\ 1, & αλλιώς \end{cases}$

όπου (n) το πλήθος των κόμβων και (k) το πλήθος των φορτηγών.

Η μεταβλητή timeFlag είναι μία δυαδική μεταβλητή που παίρνει την τιμή 0 αν το φορτηγό (k) φτάνει στον πελάτη (i) πριν τη λήξη του χρονικού παραθύρου του (li), αλλιώς παίρνει την τιμή 1. Έτσι η συνάρτηση αυτή συνυπολογίζει το συνολικό κόστος (cost), και τιμωρεί το κόστος της διαδρομής αν η συνολική ζήτηση που έχει ανατεθεί στο όχημα (load) ξεπερνά τη χωρητικότητά του (capacity), με a_1 μονάδες επί το έξτρα φορτίο, ενώ για κάθε πελάτη που το φορτηγό δεν προλαβαίνει να μεταβεί, το κόστος τιμωρείται με a_2 μονάδες. Έτσι δίνεται η δυνατότητα στον αλγόριθμο να επισκεφθεί κάποιες λύσεις που στην πραγματικότητα απαγορεύεται, αλλά με κάποιο κόστος. Άρα στην ουσία επιτρέπει να επισκεφθεί μία λύση με μεγάλο κέρδος ακόμα και αν αυτό είναι ανέφικτο, αν αυτό δεν σημαίνει την αύξηση του κόστους της συνάρτησης τιμωρίας. Αυτό ονομάζεται κριτήριο απενεργοποίησης περιορισμών (aspiration criterion) και εξαρτάται από τις τιμές των παραμέτρων a1 και α2. Με αυτόν τον τρόπο ο αλγόριθμος επιδιώκει να ξεφύγει από κάποιο τοπικό ελάχιστο, με στόχο την εύρεση μίας καλύτερης ή και της βέλτιστης λύσης.

4.2 Αρχική Λύση

Στον αλγόριθμο αυτόν χρησιμοποιήθηκε ως αρχική λύση ο αλγόριθμος του πλησιέστερου γείτονα. Με αυτόν τον αλγόριθμο, η διαδικασία μας ξεκινάει με μία αρχική εφικτή λύση. Για να το πετύχει αυτό, ο αλγόριθμος ξεκινάει απ' την αποθήκη. Στη συνέχεια ελέγχει ποιος είναι ο κοντινότερος πελάτης και αν ικανοποιούνται οι περιορισμοί του προβλήματος. Αν ικανοποιούνται, προσθέτει τον πελάτη στη διαδρομή, αν όχι εξετάζει τον επόμενο πελάτη. Αν δεν υπάρχει πελάτης του οποίου η επίσκεψή του ικανοποιεί τους περιορισμούς, δημιουργεί καινούρια διαδρομή, άρα χρησιμοποιείται ένα ακόμη φορτηγό. Η διαδικασία συνεχίζεται μέχρι να εξυπηρετηθούν όλοι οι πελάτες ακριβώς μία φορά. Έτσι δημιουργείται μία αρχική εφικτή λύση ως προς τη χωρητικότητα και ως προς τους χρονικούς περιορισμούς, που όμως συνήθως χρησιμοποιούνται περισσότερα φορτηγά από ότι το ελάχιστο δυνατό μέγεθος του στόλου.

Στη συνέχεια χρησιμοποιείται μία διαδικασία επιδιόρθωσης της αρχικής λύσης (Repair), η οποία πραγματοποιεί μείωση των φορτηγών που θα χρησιμοποιηθούν στο ελάχιστο δυνατό, βάση της χωρητικότητας των φορτηγών. Στο απλό πρόβλημα δρομολόγησης

οχημάτων (VRP) αν προστεθεί ένα όχημα επιπλέον από τον ελάχιστο απαιτούμενο στόλο, αυτομάτως θα προστεθούν στη λύση και δύο τόξα, από και προς την αποθήκη. Αυτό σημαίνει ότι είναι πολύ πιθανό το κόστος να αυξάνεται. Στο συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων δεν ισχύει κάτι τέτοιο, καθώς το ότι είναι ανθρωποκεντρικό το κάνει να μειώνεται το κόστος με την αύξηση του στόλου. Έτσι, εφόσον δεν συνυπολογίστηκε το μέγεθος του στόλου στη συνάρτηση τιμωρίας, το οποίο έγινε εσκεμμένα, αφού μας δίνει την ευχέρεια να ρυθμίσουμε μόλις δύο μεταβλητές την a_1 και την a_2 , δημιουργήθηκε η ανάγκη για χρήση της συνάρτησης μείωσης στόλου.

Έτσι χρησιμοποιείται μία διαδικασία επιδιόρθωσης της αρχικής λύσης (Repair), η οποία έχει ως στόχο τη μείωση των φορτηγών που θα χρησιμοποιηθούν στο ελάχιστο δυνατό, βάση της χωρητικότητας των φορτηγών (Αλγόριθμος 3). Αυτή η λύση είναι πιθανόν να είναι ανέφικτη, αφού γίνεται μία απλή εισαγωγή κόμβων στα ήδη υπάρχοντα φορτηγά, με μοναδικό στόχο τη μείωση του στόλου στο ελάχιστο δυνατό. Με τη συνάρτηση αυτή ο αλγόριθμος ψάχνει μία εφικτή ή μη εφικτή λύση χρησιμοποιώντας τον ελάχιστο δυνατό στόλο.

4.3 Εφαρμογή Αλγορίθμου Περιορισμένης Αναζήτησης

Αφού η λύση έχει έρθει σε μία μορφή που χρησιμοποιεί το μικρότερο δυνατό στόλο, αλλά πολύ πιθανόν να αποτελείται από ανέφικτες διαδρομές, των οποίων το κόστος έχει τιμωρηθεί σύμφωνα με τη συνάρτηση τιμωρίας, χρησιμοποιείται ο μεθευρετικός αλγόριθμος Περιορισμένης Αναζήτησης (Tabu Search) (Αλγόριθμος 6). Για να μετακινηθεί από μία λύση σε μία άλλη, χρησιμοποιούνται διάφορες μέθοδοι τοπικής αναζήτησης. Για να ξεφύγει ο αλγόριθμος από ένα πιθανό τοπικό ελάχιστο, μπορεί να αποδεχθεί κινήσεις που χειροτερεύουν τη λύση. Για να αποφευχθεί η επαναλαμβανόμενη επίσκεψη των προηγούμενων λύσεων, χρησιμοποιείται η μνήμη μικρής περιόδου ή μνήμη πρόσφατων κινήσεων, στην οποία οι τελευταίες κινήσεις προστίθενται στη λίστα περιορισμένων κινήσεων (Tabu List). Σε αυτόν τον αλγόριθμο ως μνήμη χρησιμοποιείται το συνολικό κόστος της λύσης. Αυτό, λόγω και της ιδιαιτερότητας στον υπολογισμό του κόστους στο συσσωρευτικό πρόβλημα, είναι πρακτικά αδύνατο να είναι ίδιο σε δύο διαφορετικές λύσεις. Γι' αυτό θεωρούμε ότι το κόστος της λύσης, είναι μοναδικό για κάθε λύση, άρα την αντικατοπτρίζει πλήρως.

Σύμφωνα με όσα προαναφέραμε ο αλγόριθμος αποδέχεται τον καλύτερο γείτονα, του οποίου το κόστος δεν ανήκει στη λίστα περιορισμένης αναζήτησης, άρα και η λύση δεν είναι ταμπού.

Η μνήμη μεσαίας περιόδου είναι η πιο σημαντική, καθώς στοχεύει στην εντατικοποίηση της λύσης, δηλαδή στην εξερεύνηση των γειτονιών που βρίσκονται κοντά στην καλύτερη λύση. Αυτό υπάγεται στη λογική ότι η καλύτερη λύση θα βρίσκεται κοντά στο πιο υποσχόμενο σημείο. Γι' αυτόν το λόγο τη μνήμη μεσαίας περιόδου τη διαχειριζόμαστε με δύο διαφορετικές στρατηγικές, καθώς και με απλούς συνδυασμούς αυτών των στρατηγικών.

Η πρώτη στρατηγική είναι η χρήση του πίνακα συχνοτήτων (Αλγόριθμος 2). Για να φτιαχτεί αυτός ο πίνακας εξετάζονται οι καλύτερες λύσεις οι οποίες έχουν αποθηκευτεί στη μνήμη των καλύτερων λύσεων. Ο πίνακας που δημιουργείται έχει στις γραμμές και στις στήλες του τους κόμβους. Αρχικά σε όλα τα στοιχεία του πίνακα έχουμε την τιμή 0. Στη συνέχεια καταχωρείται η συχνότητα εμφάνισης κάθε τόξου. Αν για παράδειγμα ονομάσουμε τον πίνακα συχνοτήτων A και στο σύνολο των εκλεκτών λύσεων υπάρχει δύο φορές το τόξο

(i,j)=(0,1), το αντίστοιχο σημείο του πίνακα θα πάρει την τιμή A(0,1)= 2. Σύμφωνα με αυτόν τον πίνακα κατασκευάζεται η νέα λύση.

Ξεκινώντας από την αποθήκη βρίσκουμε τον κόμβο που επισκέπτεται πιο συχνά μετά απ' αυτήν. Αυτός ο κόμβος είναι ο κόμβος που θα τοποθετηθεί αμέσως μετά την αποθήκη. Με την ίδια διαδικασία συνεχίζουμε από κόμβο σε κόμβο μέχρι το φορτηγό να γεμίσει και να δημιουργηθεί ένα καινούριο φορτηγό. Αν δημιουργηθούν φορτηγά παραπάνω απ' αυτά του ελάχιστου στόλου, τότε χρησιμοποιείται η συνάρτηση επιδιόρθωσης της λύσης (Repair).

Η δεύτερη στρατηγική εκμετάλλευσης της μνήμης μεσαίας περιόδου είναι η μέθοδος επανασύνδεσης διαδρομών, Path Relinking (PR) (Αλγόριθμος 4). Σε αυτόν τον αλγόριθμο εισάγονται δύο καλές λύσεις από τη λίστα των καλύτερων λύσεων. Ο αλγόριθμος μετατρέπει τις δύο λύσεις έτσι ώστε η μία να είναι ίδια με την άλλη, άρα κριτήριο τερματισμού του αλγορίθμου μπορεί να θεωρηθεί όταν το συνολικό κόστος των δύο λύσεων είναι ίδιο. Αν ενδιάμεσα των δύο λύσεων υπάρχει μία λύση που έχει μικρότερο κόστος από την ως τώρα καλύτερη λύση, τότε αποθηκεύεται ως η νέα καλύτερη λύση. Με αυτήν τη στρατηγική εντατικοποίησης εξερευνώνται τα μονοπάτια που είναι κοντά στην καλύτερη λύση και η λύση μπορεί να ξεφύγει από κάποιο τοπικό ελάχιστο που έχει εγκλωβιστεί. Ο αλγόριθμος PR μπορεί να χρησιμοποιηθεί με αρκετούς τρόπους, όπως αυτοί περιγράφηκαν προηγουμένως. Στη συγκεκριμένη διπλωματική εργασία χρησιμοποιήθηκαν τρεις τρόποι, η προς τα εμπρός επανασύνδεση διαδρομών (forward relinking), η προς τα πίσω επανασύνδεση διαδρομών (backward relinking), και η τυχαιοποιημένη επανασύνδεση διαδρομών (randomized relinking).

Το τελευταίο στάδιο του αλγορίθμου αποτελείται από τη στρατηγική διάχυσης των λύσεων. Στόχος αυτής της στρατηγικής είναι να εξερευνηθούν ανεξερεύνητα μονοπάτια. Εδώ ξαναχρησιμοποιήθηκαν οι δύο προηγούμενοι αλγόριθμοι, αλλά με διαφορετικό τρόπο. Πλέον ο πίνακας συχνοτήτων χρησιμοποιείται με την ακριβώς αντίστροφη διαδικασία, όπου προτεραιότητα έχουν τα τόξα που βρίσκονται πιο σπάνια στο σύνολο των λύσεων. Έτσι ο αλγόριθμος περιπλανιέται σε έναν νέο, ανεξερεύνητο χώρο των λύσεων. Στη στρατηγική διάχυσης των λύσεων, προσαρμόστηκε και ο αλγόριθμος PR, ο οποίος δέχεται ως είσοδο δύο λύσεις απ' τις καλύτερες μη εφικτές λύσεις. Έτσι ο αλγόριθμος εξερευνάει τον ανέφικτο χώρο των λύσεων, με σκοπό να ξεπεράσει το τοπικό ελάχιστο στο οποίο έχει εγκλωβιστεί και να βρεθεί στον εφικτό χώρο των λύσεων.

Στον αλγόριθμο (Tabu Search) χρησιμοποιήθηκαν τεχνικές καλύτερης αποδοχής. Άρα σε κάθε γειτονιά δεν ψάχνει για έναν γείτονα που απλώς βελτιώνει τη λύση, αλλά για τον καλύτερο γείτονα. Αυτό αυξάνει εκθετικά τον υπολογιστικό φόρτο και η αύξηση αυτή εξαρτάται από τον αριθμό των κόμβων του προβλήματος. Για να αποφύγουμε αυτήν την αύξηση του υπολογιστικού φόρτου, χρησιμοποιήθηκε μία υβριδική μορφή του αλγορίθμου. Αυτή αποτελείται από τον συνδυασμό του αλγορίθμου Περιορισμένης Αναζήτησης (Tabu Search) και του αλγορίθμου Μεταβλητής Γειτονιάς Αναζήτησης (VNS) (Αλγόριθμος 1). Ο αλγόριθμος μεταβλητής γειτονιάς αναζήτησης χρησιμοποιεί στην αναζήτηση καλύτερης γειτονιάς, τεχνικές πρώτης αποδοχής (Προσαρμογή Αλγορίθμου VNS). Ο συνδυασμός αυτών των δύο μεθόδων επιτρέπει μία μεγάλη εξερεύνηση στον χώρο των λύσεων και ταυτόχρονα εντατική εξερεύνηση της καλύτερης λύσης. Έτσι ο αλγόριθμος είναι πιο αποτελεσματικός στη βελτίωση των λύσεων ακόμα και μεγάλων προβλημάτων σε λογικό χρόνο.

Προσαρμογή Αλγορίθμου VNS

```
X = Αρχική λύση
Κύρια Φάση
Do while το κριτήριο τερματισμού δεν έχει ικανοποιηθεί
   k=1
   Do while k < k_max
      X'= VND(X)
      Αν η λύση ανήκει στη λίστα ταμπού
         Συνέχισε
      Τέλος Αν
      If f(X') ≤ f(x*) τότε
         x* = x'
      Else
         X'= Shake(X)
         k=k+1
      End if
    End do
End do
```

Στον πίνακα (πίνακας 1) που ακολουθεί παρουσιάζονται αναλυτικά οι μέθοδοι τοπικής αναζήτησης που χρησιμοποιήθηκαν σε κάθε έναν από τους αλγορίθμους. Βλέπουμε πως στον αλγόριθμο Tabu Search δεν χρησιμοποιήθηκαν όλες οι μέθοδοι τοπικής αναζήτησης, αυτό γιατί οι μέθοδοι καλύτερης αποδοχής που χρησιμοποιεί αυτός ο αλγόριθμος αυξάνουν εκθετικά τον υπολογιστικό φόρτο. Αντίθετα στον αλγόριθμο VND χρησιμοποιήθηκαν εννέα μέθοδοι τοπικής αναζήτησης.

Πίνακας 1: Μέθοδοι τοπικής αναζήτησης κάθε αλγορίθμου

VND	Tabu Search
1-1 Adjacent-Swap	1-0 Relocation
1-1 Swap	1-1 Swap
2-opt	2-opt
1-0 Relocate	1-1 Exchange
2-0 Relocate	
1-1 Exchange	
2-1 Exchange	
2-2 Exchange	
3-3 Exchange	

Οι μέθοδοι τοπικής αναζήτησης, 1-1 Adjacent-Swap, 1-1 Swap και 2-opt αφορούν αλλαγές στη λύση εντός της ίδιας διαδρομής στην οποία μπορεί να αλλάξει η σειρά με την οποία επισκέπτεται το φορτηγό τους πελάτες. Οι υπόλοιπες μέθοδοι αφορούν αλλαγές στη λύση μεταξύ δύο διαδρομών.

Παρατηρήθηκε ότι σημαντικό ρόλο στην αποτελεσματικότητα του αλγορίθμου παίζουν οι τιμές των παραμέτρων a1 και α2. Γι' αυτόν τον λόγο, με στόχο την καλύτερη εξερεύνηση στον χώρο των λύσεων, εφαρμόστηκε μία τεχνική χαλάρωσης των περιορισμών. Έτσι ξεκινώντας από μικρά a_1, a_2, επιτρέπουμε στον αλγόριθμο να εξερευνήσει μονοπάτια χαμηλού

κόστους τα οποία είναι όμως πιθανών να οδηγούν σε ανέφικτες λύσεις. Στη συνέχεια σταδιακά αυξάνονται οι συντελεστές a_1, a_2 και με αυτόν τον τρόπο ο αλγόριθμος δίνει περισσότερη σημασία στο να γίνουν όλες οι διαδρομές εφικτές. Έτσι γνωρίζοντας τα πολύ καλά μονοπάτια ο αλγόριθμος αναζητεί μικρές αλλαγές πάνω στη λύση που θα κάνουν τις ανέφικτες διαδρομές εφικτές και το κόστος της λύσης θα είναι κοντά στο βέλτιστο. Επίσης χρησιμοποιήθηκε ένα άνω όριο για τις παραμέτρους a1, a2, ώστε να συνεχίζεται ως το τέλος του αλγορίθμου η αναζήτηση σε ανέφικτα μονοπάτια, με μικρότερη όμως συχνότητα. Αν μέχρι το τέλος των συνολικών επαναλήψεων δεν έχει βρεθεί εφικτή λύση ο αλγόριθμος προσθέτει στον στόλο του ένα φορτηγό ώστε να βρεθεί μία καλή εφικτή λύση.

4.4 Υπολογιστικός Φόρτος

Για να βρούμε καλά αποτελέσματα σε λογικό χρόνο, πολύ σημαντικό ρόλο παίζει η μείωση του υπολογιστικού φόρτου του αλγορίθμου. Έτσι για κάθε μία από τις μεθόδους τοπικής αναζήτησης που χρησιμοποιεί ο αλγόριθμος, πρέπει να υπολογισθεί η διαφορά του κόστους των δύο λύσεων, και όχι το συνολικό κόστος της νέας λύσης από την αρχή. Αφού η συνάρτηση τιμωρίας εξαρτάται από τρεις παράγοντες, το κόστος της διαδρομής, τη ζήτηση που έχει ανατεθεί σε κάθε φορτηγό, καθώς και αν το φορτηγό φτάνει έγκαιρα στον πελάτη ή όχι, ο αλγόριθμος πρέπει να προϋπολογίζει το κόστος και των τριών αυτών παραγόντων.

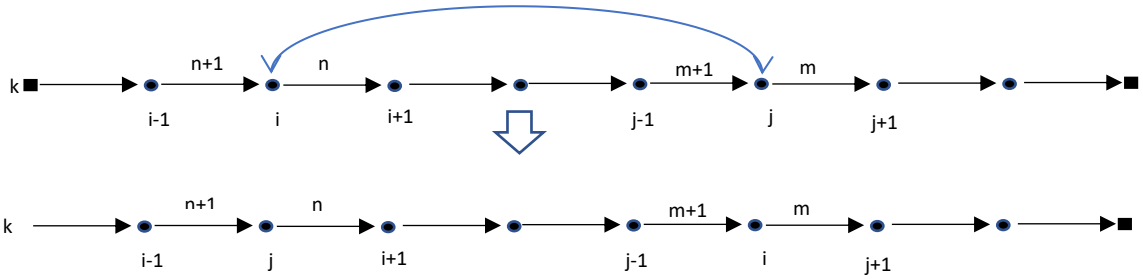
Ας δούμε πρώτα την απλή περίπτωση της ποσότητας που έχει ανατεθεί στο κάθε φορτηγό. Στην περίπτωση που η διαδικασία τοπικής αναζήτησης αναζητεί βελτίωση της λύσης εντός της ίδιας διαδρομής (1-1 swap, 2-opt), δεν αλλάζει κάτι στη ζήτηση που έχει ανατεθεί στο φορτηγό. Στην περίπτωση που η διαδικασία τοπικής αναζήτησης ανταλλάσσει ή επανατοποθετεί πελάτες μεταξύ δύο διαδρομών (1-0 relocate, 1-1 exchange, 2-1 exchange, 2-2 exchange, 2-0 relocate, 3-3 exchange) τότε ανταλλάσσεται ταυτόχρονα και η ζήτηση των πελάτων που συμμετέχουν στην ανταλλαγή αυτή. Έτσι μπορούμε πολύ εύκολα να υπολογίσουμε τη διαφορά στη ζήτηση που θα καλύψει κάθε φορτηγό, σε σταθερό χρόνο O(1).

Στην περίπτωση των χρονικών παραθύρων, η δυσκολία έγκειται στο ότι όταν γίνεται μία αλλαγή στη διαδρομή, επηρεάζονται όλοι οι υπόλοιποι πελάτες που έπονται αυτής της αλλαγής, συνυπολογίζοντας τον ενωρίτερο χρόνο άφιξης καθώς και τον χρόνο λήξης του κάθε χρονικού παραθύρου. Γι' αυτόν τον λόγο δημιουργήθηκαν δύο διανύσματα που προσαρμόζονται στον κάθε κόμβο κάθε διαδρομής. Το πρώτο διάνυσμα (delay time) εκφράζει το περιθώριο που έχει το φορτηγό, να καθυστερήσει την επίσκεψη του κάθε πελάτη, όταν το διάνυσμα αυτό παίρνει αρνητική τιμή τότε η επίσκεψη του συγκεκριμένου πελάτη γίνεται αδύνατη, και το κόστος αυξάνεται αντίστοιχα σύμφωνα με τη συνάρτηση τιμωρίας. Το δεύτερο διάνυσμα (departure time) εκφράζει τον χρόνο αναχώρησης του φορτηγού από τον κάθε πελάτη. Στον χρόνο αναχώρησης από τον κάθε πελάτη συνυπολογίζεται και ο χρόνος εξυπηρέτησης του.

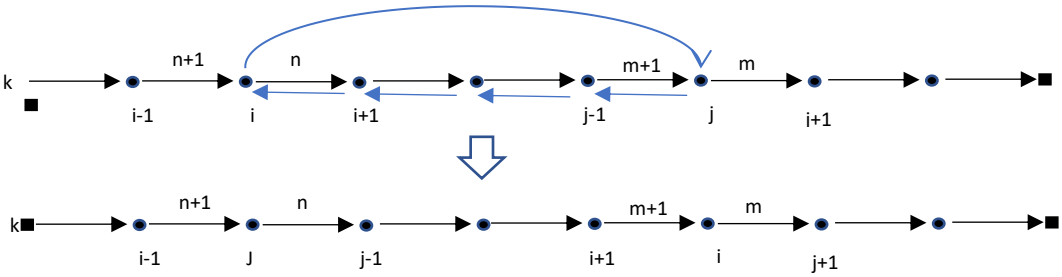
Για ακόμη μεγαλύτερη εξοικονόμηση υπολογιστικού φόρτου, ο αλγόριθμος κάνει σάρωση σε αυτά τα δύο διανύσματα χρησιμοποιώντας μία μόνο μεταβλητή. Όταν η εξεταζόμενη αλλαγή στη λύση προσδίδει κέρδος στη συνάρτηση τιμωρίας, τότε και μόνο τότε επεξεργάζονται αυτά τα δύο διανύσματα, ενημερώνοντάς τα με τις νέες τους τιμές. Έτσι ξέροντας τη χρονική στιγμή αναχώρησης από τον κάθε πελάτη μπορούμε με μία απλή πρόσθεση να υπολογίσουμε τον χρόνο άφιξης στον επόμενο πελάτη, σε μία οποιαδήποτε αλλαγή

γειτονιάς. Αν ο χρόνος άφιξης είναι μικρότερος από τον ενωρίτερο χρόνο εξυπηρέτησης του πελάτη, τότε χρόνος άφιξης θεωρείτε ο ενωρίτερος χρόνος επίσκεψης του πελάτη, αν είναι εντός χρονικών παραθύρων, εξυπηρετείται κανονικά, αλλιώς η επίσκεψη δεν είναι εφικτή.

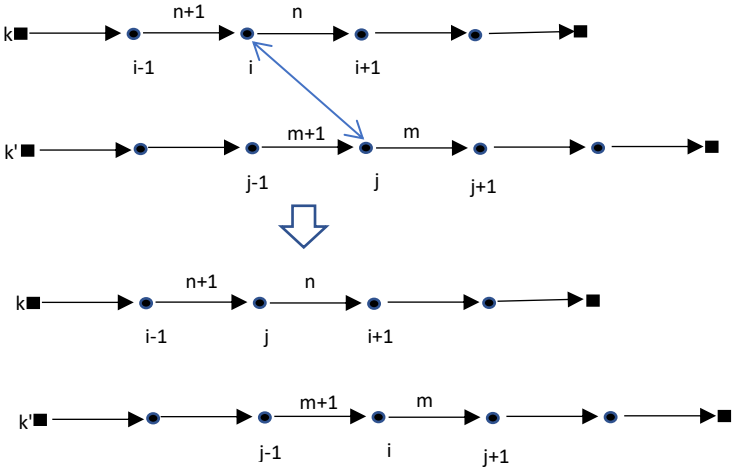
Στη συνέχεια θα εξετάσουμε τον προϋπολογισμό κόστους της διαδρομής, όπως αυτό ορίζεται από τη συνάρτηση κόστους στο συσσωρευτικό πρόβλημα, σε κάθε μία διαδικασία τοπικής αναζήτησης ξεχωριστά. Η δυσκολία στον υπολογισμό της διαφοράς κόστους, είναι ότι το κάθε τόξο πολλαπλασιάζεται με έναν συντελεστή. Άρα όταν υπολογίζεται αυτή η διαφορά κόστους πρέπει να προσαρμοστούν και να συνυπολογιστούν οι συντελεστές αυτοί. Άρα έχουμε δύο παράγοντες που πρέπει να υπολογιστούν. Αρχικά πρέπει να υπολογιστεί ποια θα είναι η διαδρομή μετά την αλλαγή και στη συνέχεια με τι συντελεστή θα πολλαπλασιαστεί το κάθε τόξο της διαδρομής. Στη συνέχεια παρουσιάζεται αναλυτικά το πως μεταβάλλεται το κόστος διαδρομής, και το πως υπολογίζεται η διάφορά κόστους για την κάθε μία γειτονιά.



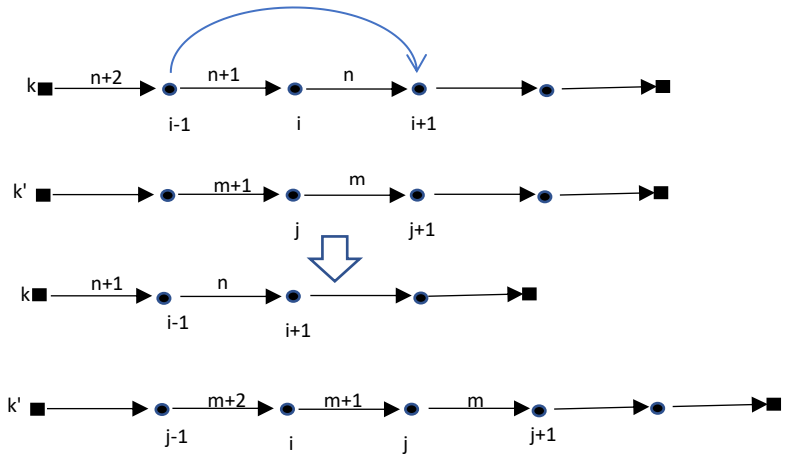
Εικόνα 4.1: Διαφορά κόστους 1-1 Swap



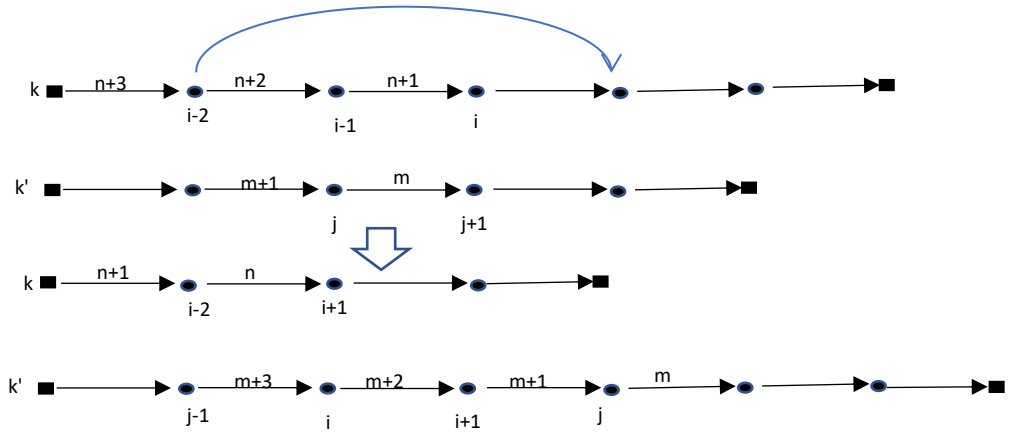
Εικόνα 4.2 : Διαφορά κόστους 2-opt



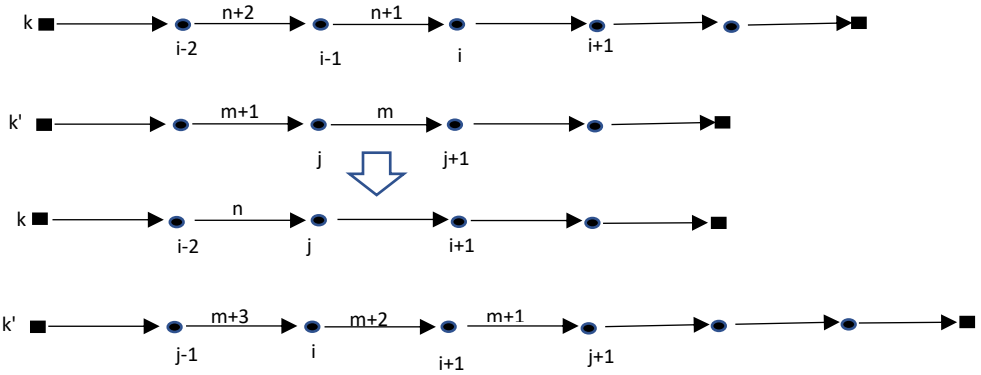
Εικόνα 4.3: Διαφορά κόστους 1-1 exchange



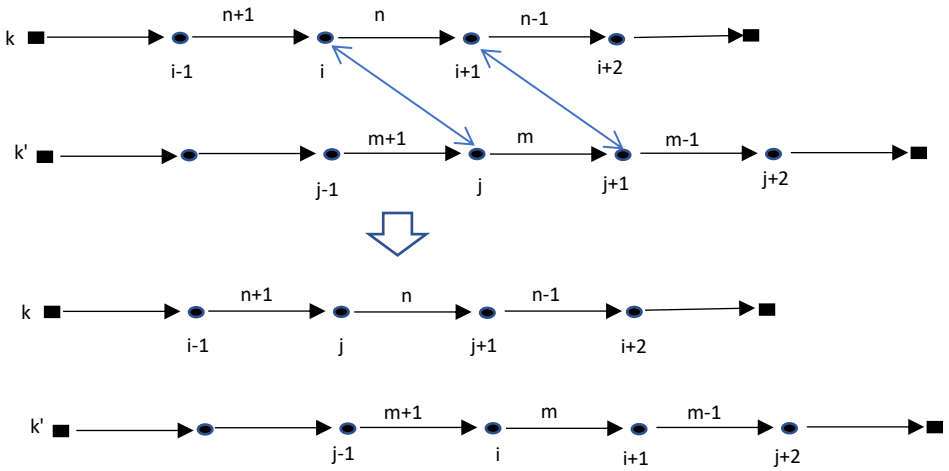
Εικόνα 4.4: Διαφορά κόστους 1-0 Relocate



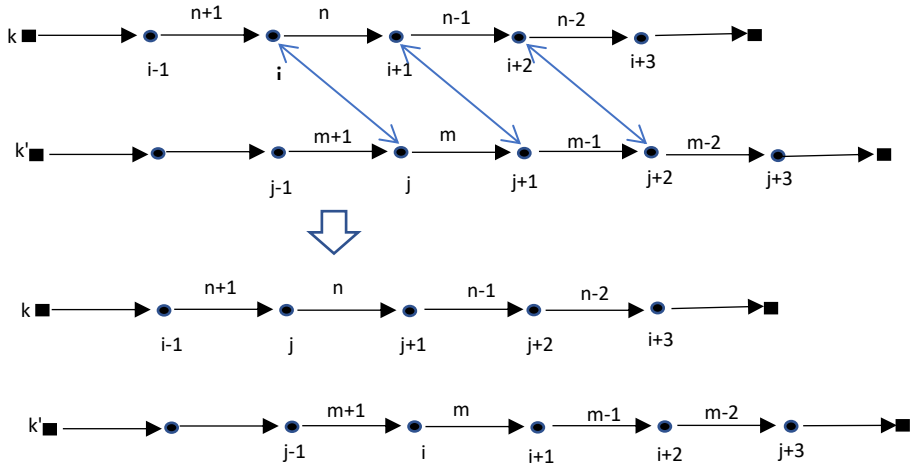
Εικόνα 4.5: Διαφορά κόστους 2-0 Relocate



Εικόνα 4.6: Διαφορά κόστους 2-1 Exchange



Εικόνα 4.7: Διαφορά κόστους 2-2 Exchange



Εικόνα 4.8: Διαφορά κόστους 3-3 Exchange

Σύμφωνα με τα προηγούμενα, για κάθε γειτονιά ξέρουμε ότι από τον πελάτη *i+1* ή *j+1* αντίστοιχα έως το τέλος της διαδρομής δεν επηρεάζεται το κόστος της διαδρομής, άρα δεν χρειάζεται να υπολογίσουμε κάτι και έτσι μειώνεται σημαντικά ο υπολογιστικός φόρτος. Επίσης παρατηρούμε ότι για τις γειτονιές 1-1 swap, 1-1 exchange, 2-2 exchange, 3-3 exchange, η διαφορά κόστους υπολογίζεται σε σταθερό χρόνο O(1), καθώς οι αλλαγές επηρεάζουν μόνο τους κόμβους που εμπλέκονται σε αυτές, οι οποίες είναι συγκεκριμένες για κάθε γειτονιά, άρα σε αυτές τις γειτονιές ο υπολογιστικός φόρτος είναι σταθερός και ανεξάρτητος του μήκους της διαδρομής. Στις υπόλοιπες γειτονιές η διαφορά κόστους δεν υπολογίζεται σε σταθερό χρόνο καθώς αν αυξάνεται το μήκος της διαδρομής, πρέπει να προστεθεί μία φορά ή δύο φορές (στην περίπτωση του 2-0 Relocate) το κόστος των τόξων που προηγούνται της αλλαγής, ενώ αντίστοιχα πρέπει να αφαιρεθεί το μήκος των τόξων που επηρεάζονται όταν μειώνεται το μήκος των διαδρομών.

4.5 Συντονισμός Παραμέτρων

Ο αλγόριθμος ξεκινώντας από μία αρχική λύση, η οποία προέρχεται από τη μέθοδο του πλησιέστερου γείτονα και τη συνάρτηση επιδιόρθωσης της λύσης, επιδιώκει στη συνέχεια μία βελτίωση της λύσης με τις μεθόδους που προαναφέραμε. Έτσι ακριβώς μετά από την αρχική λύση, ο αλγόριθμος διαβάζει όλες τις παραμέτρους σύμφωνα με τις οποίες θα τρέξει τις μεθόδους. Στις διαδικασίες του αλγορίθμου χρησιμοποιήθηκαν οι εξής παράμετροι:

- iterMAX: οι επαναλήψεις του αλγορίθμου.
- tabuIterMAX: ο αριθμός των λύσεων που αποθηκεύονται στη μνήμη μεσαίας και μακράς περιόδου.
- shackingIterMAX: οι επαναλήψεις της διαδικασίας ανακατέματος της λύσης (shacking).
- innerIterMAX: οι επαναλήψεις για κάθε στρατηγική.
- tviMAX: οι επαναλήψεις που εκτελείται ο αλγόριθμος VND.
- tabuSearchIterMax: οι επαναλήψεις που εκτελείται ο αλγόριθμος tabu search.
- Ashaking: το πόσο θα επιτρέψει ο αλγόριθμος στη διαδικασία ανακατέματος της λύσης, να αυξήσει το κόστος της συνάρτησης τιμωρίας.
- a1, a2 : οι παράμετροι της συνάρτησης τιμωρίας.
- vhma: ο ρυθμός με τον οποίο αυξάνονται οι παράμετροι a1, a2.
- aMax1: Η μέγιστη τιμή που μπορεί να πάρει η παράμετρος a1.
- aMax2: Η μέγιστη τιμή που μπορεί να πάρει η παράμετρος a2.
- diarkeiaMnhmhsKostous: Το μέγεθος της μνήμης μικρής περιόδου (short term memory).
- improved: Δείχνει αν η λύση βελτιώνεται στην τρέχουσα επανάληψη.

Αλγόριθμος 1 Tabu_VND

1 Διάβασε τη λύση s και τη λύση s*
2 k=0
3 **Do while** k< k_{max}
4 s' = Tabu Search (s)
5 s' = VND(s)
6 **if** F(s')<F(s*) **then**
7 s*=s'
8 improved = true
9 εισήγαγε το κόστος f(s*) στη λίστα περιορισμένων κινήσεων και διέγραψε από εκεί το τελευταίο
10 εισήγαγε τη λύση s* στη μνήμη των καλύτερων λύσεων και διέγραψε από εκεί την τελευταία
11 **If** η λύση s* είναι ανέφικτη **then**
12 εισήγαγε τη λύση s* στη λίστα των ανέφικτων λύσεων και διέγραψε από εκεί την τελευταία
13 **End if**
14 **End if**
15 k=k+1
16 **End do**

Αλγόριθμος 2 Μνήμη Πρόσφατων Κινήσεων

```
1   Διάβασε τη μνήμη μεσαίας περιόδου
2   Αρχικοποίησε τον πίνακα συχνοτήτων
3   Για κάθε λύση στη μνήμη
4       Για κάθε τόξο μέσα στις λύσεις
5           Προσέθεσε μία μονάδα στο αντίστοιχο κελί του πίνακα συχνοτήτων
6       ΤΕΛΟΣ_ΓΙΑ
7   ΤΕΛΟΣ_ΓΙΑ
8   ΟΣΟ δεν έχουν εξυπηρετηθεί όλοι οι πελάτες
9       Βρες το μεγαλύτερο στοιχείο της σειράς που βρίσκεσαι
10      ΑΝ ικανοποιείται ο περιορισμός χωρητικότητας
11          ΑΝ αυτός ο πελάτης δεν έχει εξυπηρετηθεί
12              Πρόσθεσέ τον στη διαδρομή
13          ΑΛΛΙΩΣ
14              Βρες τον επόμενο μεγαλύτερο στοιχείο
15          ΤΕΛΟΣ_ΑΝ
16      ΑΛΛΙΩΣ
17          Προσέθεσε ένα νέο φορτηγό στον στόλο
18      ΤΕΛΟΣ_ΑΝ
19  ΤΕΛΟΣ_ΟΣΟ
20  ΑΝ ο στόλος είναι μεγαλύτερος απ' τον ελάχιστο επιτρεπτό
21      Χρησιμοποίησε τη συνάρτηση REPAIR
22  ΤΕΛΟΣ_ΑΝ
```

Αλγόριθμος 3 Repair

```
1   ΌΣΟ ο στόλος είναι μεγαλύτερος απ' τον ελάχιστο δυνατό
2       ΌΣΟ η τελευταία διαδρομή έχει πελάτες
3           Βρες την καλύτερη δυνατή επανατοποθέτηση πελάτη από την τελευταία διαδρομή
4           Αν η ζήτησή του πελάτη δεν χωράει σε καμία διαδρομή
5               Τοποθέτησέ τον στην πρώτη θέση της πρώτης διαδρομής
6           ΤΕΛΟΣ_ΑΝ
7       ΤΕΛΟΣ_ΟΣΟ
8   ΤΕΛΟΣ_ΟΣΟ
```

Αλγόριθμος 4 Path Relinking

```
Διάβασε τη λύση s1 και τη λύση s2
ΓΙΑ κάθε διαδρομή της λύσης s1
    ΓΙΑ κάθε πελάτη της διαδρομής
        ΑΝ ο αντίστοιχος πελάτης της λύσης s2 δεν είναι ο ίδιος
            Βρες τον πελάτη στη λύση s2 και κάνε την ανταλλαγή
            ΑΝ το κόστος της λύσης βελτιώνεται
                Αποθήκευσε τη λύση ως καλύτερη
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΓΙΑ
ΤΕΛΟΣ_ΓΙΑ
Επέστρεψε την καλύτερη λύση
```

Αλγόριθμος 5 VND

```
Διάβασε τη λύση s
k=0
ΟΣΟ k<kmax
    Επέλεξε τυχαία δύο διαδρομές
    Όσο η λύση βελτιώνεται
        s'= η λύση από τις γειτονιές του s
        ΑΝ F(s') < F(s) ΚΑΙ F(s') δεν ανήκει στην Tabu List
            s=s'
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΟΣΟ
    k= k+1
ΤΕΛΟΣ_ΟΣΟ
Επέστρεψε τη λύση s
```

Αλγόριθμος 6 Tabu Search

```
Διάβασε τη λύση s
s'= s
F(s')= inf
k=0
ΟΣΟ k<kmax
    Επέλεξε τυχαία δύο διαδρομές
    s'' = ο καλύτερος γείτονας του s
    Αν F(s'') < F(s') ΚΑΙ F(s') δεν ανήκει στην Tabu List
        s'= s''
    ΤΕΛΟΣ_ΑΝ
    k= k+1
ΤΕΛΟΣ_ΟΣΟ
Επέστρεψε τη λύση s'
```

Αλγόριθμος 7 Shaking

```
Διάβασε τη λύση s και το περιθώριο χειροτέρευσης της λύσης A
k=0
ΟΣΟ k<k_max
    Επέλεξε τυχαία δύο διαδρομές
    s'= η λύση από τις γειτονιές του s
    ΑΝ F(s') - A < F(s) ΚΑΙ F(s') δεν ανήκει στην Tabu List
        s=s'
        break
    ΤΕΛΟΣ_ΑΝ
    k= k+1
ΤΕΛΟΣ_ΟΣΟ
Επέστρεψε τη λύση s
```

Αλγόριθμος 8 Κριτήρια Τερματισμού Αλγορίθμου

```
ΑΝ η λύση βελτιώθηκε στις τελευταίες 50 επαναλήψεις
    k1max = k1max + 50
ΤΕΛΟΣ_ΑΝ
ΑΝ η λύση δεν έχει βελτιωθεί για κ1max/2 επαναλήψεις
    ΑΝ η καλύτερη λύση είναι εφικτή
        break
    ΑΛΛΙΩΣ
        πρόσθεσε ένα φορτηγό στην καλύτερη λύση
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
ΑΝ η καλύτερη λύση είναι ανέφικτη
    ΑΝ a1<aMax1 ΤΟΤΕ a1= a1 +vhma ΤΕΛΟΣ_ΑΝ
    ΑΝ a2<aMax2 ΤΟΤΕ a2= a2 +vhma ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ ΑΝ
ΑΝ η καλύτερη λύση στην τελευταία επανάληψη είναι ανέφικτη
    πρόσθεσε ένα φορτηγό στην καλύτερη λύση
ΤΕΛΟΣ ΑΝ
```

Συνολικός Αλγόριθμος Tabu Search

Δημιούργησε αρχική λύση S_0 με τη μέθοδο του πλησιέστερου γείτονα
υπολόγισε $f(S_0)$
$s^* = S_0$
$f(s^*) = f(S_0)$
ΟΣΟ k< kmax
  s = TabuSearch(s)
  s = VND(s)
  Συμπλήρωσε τις τρεις μνήμες με στοιχεία από την s
  k=k+1
ΤΕΛΟΣ_ΟΣΟ
Όσο k1< k1max
  s= s*
  k2=0
  ΟΣΟ k2 < k2max
    s= Tabu_VND (S)
    k2 = k2 + 1
  ΤΕΛΟΣ_ΟΣΟ
  s = path relinking(s, s*)
  s= Tabu_VND (s)
  s = path relinking χρησιμοποιώντας δύο από τις εκλεκτές λύσεις
  s= Tabu_VND (s)
  s= algorithm 2 χρησιμοποιώντας το σύνολο των ανέφικτων λύσεων
  s= path relinking(s, s*)
  s= Tabu_VND (s)
  s= shacking(s)
  s = path relinking (s*,best infeasible solution)
  s= Tabu_VND (s)
  s= shacking(s)
  s= algorithm 2 χρησιμοποιώντας το σύνολο των εκλεκτών λύσεων
  s= Tabu_VND (s)
  s= shacking(s)
  s = path relinking backward χρησιμοποιώντας τις δύο καλύτερες λύσεις
  εξέτασε τα ΚΡΙΤΗΡΙΑ ΤΕΡΜΑΤΙΣΜΟΥ ΑΛΓΟΡΙΘΜΟΥ
ΤΕΛΟΣ_ΟΣΟ

4.6 Αποτελέσματα

Ο αλγόριθμος προγραμματίστηκε στο περιβάλλον της c++ χρησιμοποιώντας το περιβάλλον του Visual Studio 2019. Όλες οι δοκιμές γίνανε σε επεξεργαστή AMD FX-8320 (3.5 GHz) και μνήμη RAM 12GB.

Για την εξασφάλιση της ποιότητας των αποτελεσμάτων του αλγορίθμου, καθώς δεν υπάρχουν στο διαδίκτυο λύσεις για το συγκεκριμένο πρόβλημα (CCVRPTW) με βάση το κόστος, ο αλγόριθμος εξετάστηκε πρώτα στα αντίστοιχα προβλήματα της βιβλιογραφίας, που όμως δεν περιλαμβάνουν τους περιορισμούς των χρονικών παραθύρων. Έτσι μπορούμε να

συγκρίνουμε τις λύσεις του αλγορίθμου με τα καλύτερα ως τώρα αποτελέσματα. Κάθε πρόβλημα λύθηκε 10 φόρες με συγκεκριμένες παραμέτρους, ώστε να υπολογισθεί ο μέσος όρος.

Οι συνολικές επαναλήψεις του αλγορίθμου είναι 250, η διάρκεια μνήμης μεσαίας και μεγάλης περιόδου είναι 30, καθώς επίσης και η μνήμη μικρής περιόδου είναι 30. Οι επαναλήψεις της διαδικασίας ανακατέματος της λύσης (shacking) είναι 10 ενώ το επιτρεπτό όριο αύξησης του κόστους της λύσης σε αυτήν τη διαδικασία είναι 7. Οι επαναλήψεις που εφαρμόζεται κάθε στρατηγική κυμαίνεται από 2 έως 3, ενώ το βήμα αύξησης του a1 είναι μεταβλητό, αρχικοποιείται ως 0,01 και αυξάνεται κατά 0,009 μονάδες. Άρα η αύξηση της τιμής του a1 δεν είναι γραμμική. Η ελάχιστη και η μέγιστη τιμή του a1 εξαρτάται από το πρόβλημα. Το a2 λόγω του ότι δεν έχουμε χρονικά παράθυρα, παίρνει την τιμή 0 και δεν αυξάνεται.

Για να διαπιστωθεί αν ο συνδυασμός των δύο αλγορίθμων που χρησιμοποιήσαμε Tabu Search και VNS βγάζει όντως καλύτερα αποτελέσματα, τρέξαμε αρχικά τον αλγόριθμο σε μερικά ενδεικτικά προβλήματα, χωρίς τη χρήση του αλγορίθμου VNS (πίνακας 2). Άρα θέτουμε τις επαναλήψεις του αλγορίθμου VNS ίσες με το 0 και αυξάνουμε αντίστοιχα τις επαναλήψεις του αλγορίθμου Tabu Search. Οι υπόλοιπες παράμετροι του αλγορίθμου παρέμειναν ίδιες. Η ελάχιστη τιμή της παραμέτρου a1 σε αυτά τα προβλήματα κυμαίνεται από 1-10 ενώ το άνω όριο παίρνει την τιμή είτε 15 είτε 20. Στην περίπτωση του υβριδικού αλγορίθμου Tabu-VNS, ο αλγόριθμος VND επαναλαμβάνεται 35 φορές και ο αλγόριθμος Tabu Search 25 φορές.

Πίνακας 2: Σύγκριση αλγορίθμων Tabu Search χωρίς VNS VS Tabu-VNS.

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ		ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ		Η ΜΕΤΑΞΥ ΤΟΥΣ ΔΙΑΦΟΡΑ
			TABU/VNS	TABU SEARCH	TABU/VNS	TABU SEARCH	
par1	51	5	2243,32	2277,26	2230,35	2245,37	0,67%
par2	76	10	2432,93	2489,36	2391,63	2454,23	2,62%
par3	101	8	4125,82	4183,53	4045,42	4090,84	1,12%
par4	151	12	5119,53	5194,47	4987,52	5158,92	3,44%
par5	200	17	5909,03	5962,19	5826,82	5917,98	1,56%
par11	121	7	7348,10	7435,25	7314,55	7358,39	0,60%
par12	101	10	3564,60	3594,35	3558,92	3564,04	0,14%

Βλέπουμε ότι ο υβριδικός αλγόριθμος Tabu-VNS αποδίδει καλύτερα, με μέγιστη διαφορά μεταξύ τους 3,44% και ελάχιστη 0,14%. Οι παράμετροι που ρυθμίζουν πόσες φορές θα εκτελεστεί η διαδικασία VNS και Tabu Search, καθορίζουν κατά πολύ τον χρόνο περάτωσης του αλγορίθμου. Αυτές εξαρτώνται από το πλήθος των διαδρομών του προβλήματος. Όσο μεγαλύτερο είναι το ελάχιστο μέγεθος στόλου, τόσο περισσότερες δυάδες διαδρομών, άρα τόσο περισσότερος χρόνος για να εξεταστούν όλοι αυτοί οι συνδυασμοί.

Οι επαναλήψεις του αλγορίθμου ακολουθούν την εξής στρατηγική. Αν το πλήθος των διαδρομών είναι < 3 τότε οι επαναλήψεις του αλγορίθμου VND είναι 3 και οι επαναλήψεις του αλγορίθμου Tabu Search είναι 2. Αν το πλήθος των διαδρομών είναι < 10 (και >3) τότε ο αλγόριθμος VND εκτελείται 25 φορές και ο αλγόριθμος Tabu Search εκτελείται 15 φορές, αλλιώς αν το πλήθος των διαδρομών είναι μεγαλύτερο του 10 τότε ο αλγόριθμος VND εκτελείται 35 φορές και ο αλγόριθμος Tabu Search 25. Ας δούμε λοιπόν τις επιδόσεις του αλγορίθμου συγκρινόμενες με τα ως τώρα καλύτερα αποτελέσματα, σύμφωνα με το [19]. Τα προβλήματα που αναγράφονται με κόκκινο χρώμα είναι αυτά που ο αλγόριθμος βρήκε καλύτερο κόστος απ' το ήδη υπάρχον ελάχιστο κόστος.

Στο πρώτο σετ προβλημάτων set A η αρχική τιμή του a1 είναι είτε 10 είτε 15 και η μέγιστη τιμή είναι αντίστοιχα είτε 25 είτε 35, ανάλογα με το πρόβλημα. Στο δεύτερο σετ

προβλημάτων set B η ελάχιστη τιμή του a1 που χρησιμοποιήθηκε ήταν 1 και η μέγιστη τιμή 60. Στο set E η ελάχιστη τιμή του a1 είναι 7 και η μέγιστη 15, ενώ στο set M η ελάχιστη τιμή είναι 4 και η μέγιστη 7. Στο set P χρησιμοποιούμε μεγάλο εύρος τιμών του a1. Έτσι ανάλογα με το πρόβλημα η ελάχιστη τιμή του a1 κυμαίνεται από 0 έως 25 ενώ η μέγιστη από 25 έως 40. Σε αυτά τα προβλήματα η τιμή του κόστους υπολογίζεται με ακέραιες αποστάσεις.

Στην πρώτη στήλη του πίνακα αναφέρεται το όνομα του προβλήματος, στη δεύτερη ο αριθμός των κόμβων συμπεριλαμβανομένου της αποθήκης, στην τρίτη στήλη το ελάχιστο μέγεθος στόλου. Στην τέταρτη στήλη του πίνακα είναι το μέσο κόστος στα 10 τρεξίματα, ενώ στην πέμπτη στήλη το ελάχιστο κόστος που βρήκε ο αλγόριθμός μας. Στην έκτη στήλη αναφέρεται το ως τώρα γνωστό ελάχιστο κόστος (Best Known Value), ενώ στην έβδομη στήλη η ποσοστιαία διαφορά μεταξύ του ελάχιστου κόστους του δικού μας αλγόριθμου και του (BKV).

Πίνακας 3: Επίλυση Set A και σύγκριση με τα BKV

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	BKV	ΠΟΣΟΣΤΙΑΙΑ ΔΙΑΦΟΡΑ
A-n32-k5	32	5	2.192,00	**2.192**	**2.192**	0,00%
A-n33-k5	33	5	1.725,00	**1.725**	**1.725**	0,00%
A-n33-k6	33	6	1.612,00	**1.612**	**1.612**	0,00%
A-n34-k5	34	5	2.104,00	**2.104**	**2.104**	0,00%
A-n36-k5	36	5	2.279,00	**2.279**	**2.279**	0,00%
A-n37-k5	37	5	1.970,00	**1.970**	**1.970**	0,00%
A-n37-k6	37	6	2.241,40	**2.241**	**2.241**	0,00%
A-n38-k5	38	5	2.084,00	**2.084**	**2.084**	0,00%
A-n39-k5	39	5	2.312,00	**2.312**	**2.312**	0,00%
A-n39-k6	39	6	2.216,00	**2.216**	**2.216**	0,00%
A-n44-k6	44	6	2.563,00	**2.563**	**2.563**	0,00%
A-n45-k6	45	6	2.866,90	**2.848**	**2.848**	0,00%
A-n45-k7	45	7	2.837,00	**2.831**	**2.831**	0,00%
A-n46-k7	46	7	2.373,00	**2.373**	**2.373**	0,00%
A-n48-k7	48	7	3.101,00	**3.101**	**3.101**	0,00%
A-n53-k7	53	7	3.148,40	**3.115**	**3.115**	0,00%
A-n54-k7	54	7	3.377,30	**3.357**	**3.357**	0,00%
A-n55-k9	55	9	2.595,70	**2.588**	**2.588**	0,00%
A-n60-k9	60	9	3.452,30	**3.446**	**3.446**	0,00%
A-n61-k9	61	9	2.898,90	**2.868**	**2.868**	0,00%
A-n62-k8	62	8	3.932,20	**3.925**	**3.925**	0,00%
A-n63-k9	63	9	4.645,20	**4.630**	**4.630**	0,00%
A-n63-k10	63	10	3.280,20	**3.256**	**3.256**	0,00%
A-n64-k9	64	9	4.148,80	**4.135**	**4.135**	0,00%
A-n65-k9	65	9	3.496,60	**3.487**	**3.487**	0,00%
A-n69-k9	69	9	3.534,20	**3.528**	**3.528**	0,00%
A-n80-k10	80	10	5.943,80	**5.929**	**5.929**	0,00%

Πίνακας 4: Επίλυση Set B και σύγκριση με τα BKV

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	BKV	ΠΟΣΟΣΤΙΑΙΑ ΔΙΑΦΟΡΑ
B-n31-k5	31	5	1.830,20	**1.830**	**1.830**	0,00%
B-n34-k5	34	5	2.271,00	**2.271**	**2.271**	0,00%
B-n35-k5	35	5	2.846,00	**2.846**	**2.846**	0,00%
B-n38-k6	38	6	2.103,00	**2.103**	**2.103**	0,00%
B-n39-k5	39	5	1.960,10	**1.960**	**1.960**	0,00%
B-n43-k6	43	6	2.123,40	**2.123**	**2.123**	0,00%
B-n44-k7	44	7	2.295,50	**2.295**	**2.295**	0,00%
B-n45-k5	45	5	2.386,00	**2.386**	**2.386**	0,00%
B-n45-k6	45	6	2.062,30	**2.057**	**2.057**	0,00%
B-n50-k7	50	7	2.261,00	**2.261**	**2.261**	0,00%
B-n50-k8	50	8	2.954,60	**2.953**	**2.953**	0,00%
B-n51-k7	51	7	3.134,30	**3.133**	**3.133**	0,00%
B-n52-k7	52	7	2.573,00	**2.573**	**2.573**	0,00%
B-n56-k7	56	7	2.358,00	**2.358**	**2.358**	0,00%
B-n57-k7	57	7	3.893,80	3.883	3.885	-0,05%
B-n57-k9	57	9	4.502,30	**4.500**	**4.500**	0,00%
B-n63-k10	63	10	4.382,80	**4.379**	**4.379**	0,00%
B-n64-k9	64	9	2.611,90	**2.608**	**2.608**	0,00%
B-n66-k9	66	9	4.138,40	4.120	4.132	-0,29%
B-n67-k10	67	10	2.869,00	**2.868**	**2.868**	0,00%
B-n68-k9	68	9	4.058,00	**4.058**	**4.058**	0,00%
B-n78-k10	78	10	3.706,50	**3.701**	**3.701**	0,00%

Πίνακας 5: Επίλυση Set E & Set M και σύγκριση με τα BKV

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	BKV	ΠΟΣΟΣΤΙΑΙΑ ΔΙΑΦΟΡΑ
E-n22-k4	22	4	845,00	**845**	**845**	0,00%
E-n23-k3	23	3	1.908,00	**1.908**	**1.908**	0,00%
E-n30-k3	30	3	1.987,00	1.987	1.984	0,15%
E-n33-k4	33	4	2.852,00	**2.852**	**2.852**	0,00%
E-n51-k5	51	5	2.231,30	**2.213**	**2.213**	0,00%
E-n76-k7	76	7	2.943,30	2.926	2.920	0,21%
E-n76-k8	76	8	2.690,00	2.687	2.686	0,04%
E-n76-k10	76	10	2.417,50	2.392	2.366	1,10%
E-n76-k14	76	14	2.093,90	2.080	2.093	-0,62%
E-n101-k8	101	8	4.019,80	**3.954**	**3.954**	0,00%
E-n101-k14	101	14	2.972,40	2.958	2.955	0,10%
M-n101-k10	101	10	3.569,30	**3.565**	**3.565**	0,00%
M-n121-k7	121	7	7.257,80	7.223	7.230	-0,10%
M-n151-k12	151	12	5.010,10	**4.917**	**4.917**	0,00%

Πίνακας 6: Επίλυση Set P και σύγκριση με τα BKV

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	BKV	ΠΟΣΟΣΤΙΑΙΑ ΔΙΑΦΟΡΑ
P-n16-k8	16	8	396,00	**396**	**396**	0,00%
P-n19-k2	19	2	849,60	**849**	**849**	0,00%
P-n20-k2	20	2	925,30	**924**	**924**	0,00%
P-n21-k2	21	2	929,50	**928**	**928**	0,00%
P-n22-k8	22	8	681,00	**681**	**681**	0,00%
P-n23-k8	23	8	619,50	**616**	**616**	0,00%
P-n40-k5	40	5	1.541,80	**1.541**	**1.541**	0,00%
P-n45-k5	45	5	1.899,00	**1.894**	**1.894**	0,00%
P-n50-k7	50	7	1.563,30	**1.554**	**1.554**	0,00%
P-n50-k8	50	8	1.548,50	**1.533**	**1.533**	0,00%
P-n50-k10	50	10	1.351,70	**1.347**	**1.347**	0,00%
P-n51-k10	51	10	1.495,90	1.487	1.489	-0,13%
P-n55-k7	55	7	1.765,20	**1.764**	**1.764**	0,00%
P-n55-k10	55	10	1.470,70	**1.463**	**1.463**	0,00%
P-n60-k10	60	10	1.714,60	**1.704**	**1.704**	0,00%
P-n60-k15	60	15	1.513,40	**1.509**	**1.509**	0,00%
P-n65-k10	65	10	1.961,60	**1.948**	**1.948**	0,00%
P-n70-k10	70	10	2.137,20	**2.121**	**2.121**	0,00%
P-n76-k4	76	4	4.644,30	**4.610**	**4.610**	0,00%
P-n76-k5	76	5	3.849,50	**3.795**	**3.795**	0,00%
P-n101-k4	101	4	7.031,70	6.946	6.943	0,04%

Στα επόμενα σετ προβλημάτων οι αποστάσεις υπολογίζονται ως δεκαδικοί αριθμοί. Εδώ παρουσιάζονται τα αποτελέσματα του αλγορίθμου συγκρινόμενα με το καλύτερο ως τώρα κόστος με τις ίδιες τιμές όπως προαναφέρθηκαν. Εδώ γίνεται και χρονομέτρηση του αλγορίθμου. Έτσι στους επόμενους πίνακες προστίθεται και η στήλη του μέσου χρόνου σε 10 τρεξίματα μετρημένος σε δευτερόλεπτα.

Πίνακας 7: Επίλυση par προβλημάτων και σύγκριση με τα BKV

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	BKV	ΠΟΣΟΣΤΙΑΙΑ ΔΙΑΦΟΡΑ	ΧΡΟΝΟΣ (s)
par1	51	5	2.243,32	**2.230,35**	**2.230,35**	0,00%	30,45
par2	76	10	2.432,93	**2.391,63**	**2.391,63**	0,00%	37,36
par3	101	8	4.125,82	**4.045,42**	**4.045,42**	0,00%	74,82
par4	151	12	5.119,53	**4.987,52**	**4.987,52**	0,00%	108,55
par5	200	17	5.909,03	5.826,82	5.806,02	0,36%	166,00
par11	121	7	7.348,10	**7.314,55**	**7.314,55**	0,00%	161,40
par12	101	10	3.564,60	**3.558,92**	**3.558,92**	0,00%	48,80

Ο αλγόριθμος δοκιμάστηκε επίσης σε 5 από τα προβλήματα του set Kel. Αυτά τα προβλήματα εκτός από τον μεγάλο αριθμό πελατών που πρέπει να εξυπηρετηθούν, έχουν τη δυσκολία ότι οι πελάτες είναι απόλυτα συμμετρικά κατανεμημένοι στον χώρο γύρο απ' την αποθήκη. Τα προβλήματα αυτά περιέχουν από 240 μέχρι 420 πελάτες Η ελάχιστη τιμή του a1 είναι 0,1 ενώ η μέγιστη είναι 4.

Πίνακας 8: Επίλυση Kel προβλημάτων και σύγκριση με τα BKV

ΠΡΟΒΛΗΜΑ	ΚΟΜΒΟΙ	ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ	ΜΕΣΟ ΚΟΣΤΟΣ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	BKV	ΠΟΣΟΣΤΙΑΙΑ ΔΙΑΦΟΡΑ	ΧΡΟΝΟΣ (s)
kel01	241	9	55.027,99	54.823,40	54.683,36	0,26%	622,67
kel02	321	10	101.678,08	101.303,00	100.560,00	0,74%	1150,33
kel09	256	14	4.746,94	4.730,50	4.723,77	0,14%	301,67
kel10	324	16	6.757,30	6.727,73	6.712,53	0,23%	527,67
kel20	421	38	7.345,51	7.273,31	7.209,31	0,89%	363,33

Βλέπουμε ότι ο αλγόριθμός έχει μικρή απόκλιση από τα βέλτιστα, η οποία κυμαίνεται από 0,14% έως 0,89%.

Στη συνέχεια θα εξετάσουμε τα προβλήματα που εμπεριέχουν τους περιορισμούς των χρονικών παραθύρων. Οι συνολικές επαναλήψεις του αλγορίθμου είναι 250, η διάρκεια μνήμης μεσαίας και μεγάλης περιόδου είναι 30, καθώς επίσης και η μνήμη μικρής περιόδου είναι 30. Οι επαναλήψεις της διαδικασίας ανακατέματος της λύσης (shacking) είναι 10 ενώ το επιτρεπτό όριο αύξησης του κόστους της λύσης σε αυτήν τη διαδικασία είναι 10. Η κάθε στρατηγική εφαρμόζεται για δύο επαναλήψεις, ενώ το βήμα αύξησης των παραμέτρων a1 και a2 είναι μεταβλητό, αρχικοποιείται ως 0,01 και αυξάνεται κατά 0,009 μονάδες. Άρα η αύξηση της τιμής των a1 και a2 δεν είναι γραμμική. Η ελάχιστη και η μέγιστη τιμή του εξαρτάται από το πρόβλημα ενώ η μέγιστη είναι 50. Για τη μέγιστη τιμή του a2 ισχύει a2max=a2+600 ενώ η ελάχιστη εξαρτάται από το πρόβλημα.

Όλα τα προβλήματα που εξετάστηκαν έχουν 100 κόμβους. Τα προβλήματα C1, R1 και RC1 χρησιμοποιούν φορτηγά χωρητικότητας 200 μονάδων, ενώ τα προβλήματα C2, R2 και RC2 χρησιμοποιούν φορτηγά χωρητικότητας 1000 μονάδων. Μπορούμε εύκολα να συμπεράνουμε ότι στη δεύτερη κατηγορία δεν παραβιάζεται εύκολα ο περιορισμός της χωρητικότητας των φορτηγών, ενώ το ελάχιστο δυνατό μέγεθος στόλου είναι αρκετά μικρό, από 2 έως 4 φορτηγά. Το ελάχιστο μέγεθος στόλου σε αυτά τα προβλήματα δίνεται απ' τα δεδομένα, ωστόσο σε κάποια απ' αυτά έχουν βρεθεί λύσεις με λιγότερα οχήματα. Άρα ως βέλτιστο αριθμό οχημάτων ορίζουμε το μικρότερο απ' αυτό των δεδομένων και του [16].

Αρχικά θα ελέγξουμε αν ο υβριδικός αλγόριθμος που σχεδιάσαμε Tabu-VNS βγάζει όντως καλύτερα αποτελέσματα σε σχέση με τον απλό Tabu Search. Έτσι εφαρμόστηκε αρχικά ο αλγόριθμος σε μερικά ενδεικτικά προβλήματα, χωρίς τη χρήση του αλγορίθμου VNS. Άρα θέτουμε τις επαναλήψεις του αλγορίθμου VND ίσες με το 0 και αυξάνουμε τις επαναλήψεις του αλγορίθμου Tabu Search στο 60. Οι υπόλοιπες παράμετροι του αλγορίθμου παρέμειναν ίδιες. Στην περίπτωση του υβριδικού αλγορίθμου Tabu-VNS, ο αλγόριθμος VND επαναλαμβάνεται 35 φορές και ο αλγόριθμος Tabu Search 25 φορές. Ακολουθεί ο πίνακας με τη σύγκριση των δύο αλγορίθμων.

Πίνακας 9: Σύγκριση αλγορίθμων Tabu Search VS Tabu-VNS σε προβλήματα με χρονικά παράθυρα.

ΠΡΟΒΛΗΜΑ	ΕΛΑΧΙΣΤΟ ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ		ΜΕΣΟ ΜΕΓΕΘΟΣ ΣΤΟΛΟΥ		Η ΜΕΤΑΞΥ ΤΟΥΣ ΔΙΑΦΟΡΑ
	TABU/VNS	TABU SEARCH	TABU/VNS	TABU SEARCH	
R101	19	20	19,00	21,4	12,63%
R102	17	18	18,27	18,6	1,82%
R103	14	15	14,47	15,2	5,07%
R104	10	10	10,20	11	7,84%
R105	14	15	14,87	15,3	2,91%

Όπως βλέπουμε ο υβριδικός αλγόριθμος Tabu-VNS αποδίδει καλύτερα σε σχέση με τον απλό αλγόριθμο Tabu Search, καθώς σε τέσσερα απ' τα πέντε προβλήματα που λύθηκαν βρίσκει λύσεις με ένα όχημα λιγότερο. Άρα σύμφωνα με αυτόν θα λυθούν όλα τα υπόλοιπα προβλήματα.

Στο σετ προβλημάτων C1 η ελάχιστη τιμή του a1 είναι 10 ενώ η ελάχιστη τιμή του a2 είναι 30. Στο σετ προβλημάτων R1 η ελάχιστη τιμή του a1 είναι 1 ενώ του a2 είναι 30. Στο σετ προβλημάτων C2 η ελάχιστη τιμή του a1 είναι 10 ενώ η ελάχιστη τιμή του a2 είναι 30, ενώ στο σετ προβλημάτων R2 η ελάχιστη τιμή του a1 είναι 5 ενώ του α2 είναι 1000. Σε όλο το σετ προβλημάτων RC η ελάχιστη τιμή για το a1 είναι 5 ενώ για το α2 είναι 300. Η μέγιστη τιμή του a1 ορίζεται ως a1max= 50 ενώ για το a2 ισχύει a2max= a2 + 600.

Σε όλα τα προβλήματα η μέση τιμή του κόστους αναφέρεται σε 10 επαναλήψεις στις οποίες χρησιμοποιήθηκε ο ελάχιστος αριθμός οχημάτων, ενώ το ελάχιστο κόστος αναφέρεται στον βέλτιστο αριθμό οχημάτων. Η διαφορά αναφέρεται στον ελάχιστο αριθμό οχημάτων του δικού μας αλγορίθμου σε σχέση με τον ελάχιστο ως τώρα γνωστό αριθμό οχημάτων.

Πίνακας 10: Επίλυση Set C1 και Set R1 και σύγκριση με τον βέλτιστο αριθμό οχημάτων

ΠΡΟΒΛΗΜΑ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	ΜΕΣΟ ΚΟΣΤΟΣ	ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΜΕΣΟΣ ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΒΕΛΤΙΣΟΣ ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΔΙΑΦΟΡΑ	ΧΡΟΝΟΣ (s)
C101	3.865,78	3.865,78	**10**	10,0	**10**	0	127,0
C102	3.856,17	3.859,01	10	10,0	9	1	154,7
C103	3.855,03	3.855,09	10	10,0	9	1	222,7
C104	3.772,94	3.775,79	10	10,0	9	1	222,3
C105	3.865,78	3.865,78	**10**	10,0	**10**	0	129,3
C106	3.865,78	3.865,78	**10**	10,0	**10**	0	135,7
C107	3.865,78	3.865,78	**10**	10,0	**10**	0	135,7
C108	3.863,35	3.863,35	**10**	10,0	**10**	0	143,7
C109	3.857,22	3.857,22	10	10,0	9	1	144,7
R101	4.202,69	4.271,84	**19**	19,0	**19**	0	121,0
R102	4.009,32	4.012,60	**17**	18,3	**17**	0	226,0
R103	4.137,95	4.157,88	14	14,5	13	1	424,7
R104	4.247,65	4.326,47	10	10,2	9	1	818,3
R105	4.361,74	4.650,15	**14**	14,9	**14**	0	369,3
R106	4.463,04	4.507,51	13	13,3	12	1	491,0
R107	4.535,91	4.597,75	11	11,2	10	1	826,3
R108	4.031,05	4.095,66	**9**	10,0	**9**	0	668,3
R109	4.387,95	4.501,67	12	12,4	11	1	473,3
R110	4.177,40	4.336,09	11	11,3	10	1	606,0
R111	4.478,57	4.554,15	11	11,5	10	1	661,0
R112	3.991,25	4.219,05	10	10,1	9	1	536,3

Πίνακας 11: Επίλυση Set C2 και Set R2 και σύγκριση με τον βέλτιστο αριθμό οχημάτων

ΠΡΟΒΛΗΜΑ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	ΜΕΣΟ ΚΟΣΤΟΣ	ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΜΕΣΟΣ ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΒΕΛΤΙΣΟΣ ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΔΙΑΦΟΡΑ	ΧΡΟΝΟΣ (s)
C201	7.643,65	7.748,34	**3**	3,0	**3**	0	244,0
C202	7.640,45	7.695,36	**3**	3,0	**3**	0	275,0
C203	7.404,43	7.434,28	**3**	3,0	**3**	0	318,0
C204	7.257,49	7.285,36	**3**	3,0	**3**	0	249,0
C205	7.657,47	7.657,47	**3**	3,0	**3**	0	209,0
C206	7.620,29	7.620,43	**3**	3,0	**3**	0	240,0
C207	7.527,96	7.527,96	**3**	3,0	**3**	0	240,0
C208	7.521,26	7.528,45	**3**	3,0	**3**	0	251,0
R201	15.507,40	15.743,81	**4**	4,3	**4**	0	588,3
R202	18.849,10	19.631,17	**3**	3,8	**3**	0	430,3
R203	15.222,60	15.326,18	**3**	3,7	**3**	0	447,7
R204	20.267,50	20.753,58	**2**	2,8	**2**	0	866,8
R205	16.578,10	18.077,34	**3**	3,7	**3**	0	303,0
R206	14.607,30	15.041,95	**3**	3,6	**3**	0	259,7
R207	12.650,80	13.092,64	3	3,0	2	1	1058,7
R208	16.613,00	17.095,51	**2**	2,7	**2**	0	626,3
R209	14.762,50	15.322,90	**3**	3,5	**3**	0	351,3
R210	15.637,70	15.943,09	**3**	3,5	**3**	0	294,7
R211	12.425,20	12.699,62	3	3,0	2	1	852,3

Πίνακας 12: Επίλυση Set RC1 και Set RC2 και σύγκριση με τον βέλτιστο αριθμό οχημάτων

ΠΡΟΒΛΗΜΑ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ	ΜΕΣΟ ΚΟΣΤΟΣ	ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΜΕΣΟΣ ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΒΕΛΤΙΣΟΣ ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ	ΔΙΑΦΟΡΑ	ΧΡΟΝΟΣ (s)
RC101	5.212,08	5.254,81	15	15,5	14	1	356,0
RC102	5.297,68	5.389,34	13	13,9	12	1	385,3
RC103	5.318,24	5.395,38	**11**	11,3	**11**	0	350,5
RC104	4.830,75	4.925,73	**10**	10,2	**10**	0	271,0
RC105	4.867,66	4.959,39	15	15,5	13	2	588,0
RC106	5.402,45	5.463,76	12	12,4	11	1	519,0
RC107	5.031,48	5.124,28	**11**	11,2	**11**	0	448,0
RC108	5.173,73	5.360,71	**10**	10,5	**10**	0	596,3
RC201	17.286,20	17.655,01	**4**	4,2	**4**	0	121,7
RC202	13.197,70	13.915,22	4	4,0	3	1	548,7
RC203	15.918,00	16.191,30	**3**	3,7	**3**	0	416,7
RC204	11.823,60	11.987,35	**3**	3,5	**3**	0	416,0
RC205	15.071,40	15.552,76	**4**	4,3	**4**	0	161,3
RC206	19.064,00	19.738,97	**3**	3,5	**3**	0	353,3
RC207	15.930,80	16.774,44	**3**	3,6	**3**	0	396,3
RC208	12.952,20	13.649,69	**3**	3,1	**3**	0	342,0

ΚΕΦΑΛΑΙΟ 5: Συμπεράσματα

Στα προβλήματα χωρίς τους περιορισμούς των χρονικών παραθύρων είδαμε πως ο υβριδικός αλγόριθμος Tabu-VNS είναι αρκετά αποτελεσματικός στην εύρεση μίας καλής λύσης. Στα περισσότερα εξ αυτών ο αλγόριθμός μας βρήκε τις ως τώρα γνωστές βέλτιστες λύσεις, ενώ σε κάποια την βελτίωσε. Πιο αναλυτικά, από τα 96 προβλήματα που λύθηκαν συνολικά, ο υβριδικός αλγόριθμος Tabu-VNS βρήκε βέλτιστη λύση σε 84 προβλήματα ενώ σε 5 απ' αυτά βελτίωσε την ήδη υπάρχουσα βέλτιστη λύση. Αναλυτικά οι βέλτιστες λύσεις έχουν καταγραφεί στο Παράρτημα Α. Σε 12 από τα 84 προβλήματα ο αλγόριθμος δεν βρήκε βέλτιστη λύση. Η μεγαλύτερη διαφορά από το βέλτιστο είναι στο πρόβλημα kel20 στο οποίο πρέπει να εξυπηρετηθούν 420 πελάτες, και η ποσοστιαία διαφορά είναι 0,86%. Αντίστοιχα η μεγαλύτερη βελτίωση αποτελέσματος είναι στο πρόβλημα E-n76-k14 στο οποίο πρέπει να εξυπηρετηθούν 75 πελάτες και η βελτίωση είναι 0,62%.

Στα προβλήματα με τους περιορισμούς των χρονικών παραθύρων είδαμε πως ο αλγόριθμός μας είναι και εδώ αρκετά αποδοτικός, ενώ στα περισσότερα προβλήματα βρήκε λύσεις στον βέλτιστο αριθμό οχημάτων. Όπως είδαμε σε 1 από τα 56 προβλήματα που λύθηκαν, οι λύσεις που βρήκε ο αλγόριθμος ήταν με δύο φορτηγά περισσότερο από το βέλτιστο, σε 18 από τα 56 βρήκε λύσεις χρησιμοποιώντας ένα φορτηγό λιγότερο από το βέλτιστο, ενώ στα υπόλοιπα 37 προβλήματα ο αλγόριθμός μας βρήκε λύσεις στον βέλτιστο αριθμό οχημάτων. Ο αλγόριθμος έβγαλε πολύ καλά αποτελέσματα στο συσσωρευτικό πρόβλημα δρομολόγησης οχημάτων, τόσο στα προβλήματα με χρονικά παράθυρα (CCVRPTW), όσο και σε αυτά χωρίς χρονικά παράθυρα (CCVRP), στα οποία σε κάποια απ' αυτά έχουν επαληθευθεί οι βέλτιστες λύσεις τους με αλγορίθμους Δυναμικού Προγραμματισμού.

Βιβλιογραφία

Ξένη

[1] Clarke, G., Wright, J.W. (1964). *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points Operations Research*, 12, 568-581.

[2] Dantzig, G.B., Ramser, J.H. (1959). The Truck Dispatcing Problem, Management Science, 6(1), 80-91.

[3] Glover, F. (1989). Tabu Search I, *ORSA Journal on Computing*, 1 (3), 190-206.

[4] Glover, F. (1990). Tabu Search II, *ORSA Journal on Computing*, 2 (1), 4-32.

[5] Glover, F., Laguna, M. (1997). *Tabu Search,* New York, Springer Science+Business Media, LLC.

[6] Glover, F., Laguna, M., Marti, R. (2003). *Scatter Search and Path Relinkng, Handbooks of Metaheurestics*, Glover, F., Kochenberger, G. A., (Editors), Kluewer Academic Publishers, Dordrecht, 1-36.

[7] Goldeberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison – Wesley Publishing Company, INC, Massachussets.

[8] Golden, B.L., Assad, A.A. (1988). *Vehicle Routing: Methods and studies*, North Holland, Amsterdam.

[9] Golden, B.L., Raghavanm S., Wasil, E.A. (Eds.) (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges.* Operations Research/Computer Science Interfaces Series, 43, Springer LLC.

[10] Hansen, P., Mladenovic, N. (2001). *Variable Neighborhood Search: Principle and Applications, European Journal of Operational Research*, 130, 449-467.

[11] Hansen, P., Mladenovic, N. (2002). *Variable Neighborhood Search, Handbook of Metaheuristics*, Operations Research and Management Science, Kluwer Academic Publishers, 57, 145-184.

[12] Harland, C.M. (1996). *Supply Chain Management: Relationships, Chains and Networks, British Journal Management,* British Academy of Management, 7, 63-80.

[13] Haupt, R. L., Haupt, S. E. (2004). *Practical Genetic Algorithms*, Second Edition, John Wiley and Sons, Inc., Hoboken, New Jersey.

[14] Holland, J.H. (1975). *Adaption in Natural Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[15] Lin, s. (1965). *Computer Solutions of the Traveling Salesman Problem. Bell Systems, Technical Journal*, 44, 2245-2269.

[16] Liu, R., Jiang, Z. (2019). *A hybrid large-neighborhood search algorithm for the cumulative capacitated vehicle routing problem with time-window constraints, Applied Soft Computing Journal,* 80, 18-30.

[17] Lysgaard, J., Wohlk, s. (2014). A Branch-and-Cut-and-Price Algorithm for the Cumulative Capacited Vehcle Routing Problem, *European Journal of Operational Research*, 220, 349-360.

[18] Ngueveua, S.U., Prins, C., Calvo, R.W. (2010). An Effective Memetic Algorithm for the cumulative Capacitated Vehicle Routing Problem, *Computers an Operations Research*, 37, 877 – 1885.

[19] Nucamendi-Guillen, S., Angel Bello, F., Martinez-Salazar, I., Cordero-Franco, A. (2018). *The cumulative capacitated vehicle routing problem, New formulations and iterated greedy algorithms, Expert Systems with Applications*, 113, 315-327.

[20] Or, I. (1976). *Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking.* Ph. D. Thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanstton, IL.

[21] Pereira, G., Tavares, J. (2008). *Bio-inspired Algorithms for the Vehicle Routing Problem. Studies in Computational Intelligence*, 161, Springer, Berlin, Heidemberg.

[22] Rego, C.R., Glover, f. (2002). *Local Search and Metaheuristics. The Traveling Salesman Problem and its Variations,* Gutin, G., Punnen, A. (editors), Kluwer Academic Publishers, Dordrecht, 309-367.

[23] Ribeiro, G.M., Laporte, G. (2012). An adaptive Large Neighborhood Search Heuristic for the Cumulative Capacitated Vehicle Routing Problem*, Computers an Operations Research,* 39, 728-735.

[24] Rosenkratz, D.J., Stearns, R.E., Lewis P.M. (1977). *An Analysis of Several Heuristics for the Travelling Salesman Problem, SIAM Journal Computing,* 6, 563-581.

[25] Solomon, M.M (1987). *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research*, 35(2), 254-265.

[26] Solomon, M.M., Desrosiersm J. (1988). *Time Window Constrained Routing and Scheduling Problems, Trasporation Scince*, 22(1), 1-13

[27] Toth, P., Vigo, D. (2002). *The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications*, Siam, Philadelphia.

[28] Watters, C.D.J. (1987). A Solution Procedure for the Vehicle Scheduling Problem Based on Iterative Route Improvement, *Journal of Operative Research Society*, 38, 833-839.

Ελληνική

[29] Μαρινάκης, Ι., & Μαρινάκη, Μ., & Ματσατσίνης, Ν., & Ζοπουνίδης Κ., (2011). *Μεθευρετικοί και Εξελικτικοί Αλγόριθμοι σε Προβλήματα διοικητικής επιστήμης* , Αθήνα, Κλειδάριθμος.

[30] Μαρινάκης, Ι., & Μαρινάκη, Μ., & Μυγδαλάς, Α., (2019). *Προβλήματα Δρομολόγησης Οχημάτων στη Διαχείριση της εφοδιαστικής Αλυσίδας* , Αθήνα, Εκδόσεις Νέων τεχνολογιών.

Παράρτημα Α Νέες Καλύτερες Λύσεις

M-n121-k7 – capacity =200

0 87 92 91 90 18 118 108 21 20 23 25 22 24 27 30 31 28 32 35 36 34 33 0 Load=197 Route Time=194 Route Cost = 1443

0 95 96 93 94 97 115 110 52 54 57 59 65 61 62 64 66 0 Load=192 Route Time= 212 Route Cost = 988

0 102 101 99 100 116 98 73 76 77 78 80 79 53 55 58 56 60 63 0 Load=194 Route Time= 210 Route Cost = 1087

0 105 106 107 104 103 67 69 70 71 74 72 75 68 40 43 0 Load=197 Route Time= 203 Route Cost = 779

0 88 82 111 86 85 112 84 117 113 83 6 7 9 10 11 3 2 1 120 0 Load=198 Route Time= 125 Route Cost = 683

0 89 114 109 37 38 39 42 41 44 46 49 47 50 51 48 45 0 Load=199 Route Time= 199 Route Cost = 1291

0 119 81 5 4 15 14 13 8 12 17 16 19 26 29 0 Load=198 Route Time= 204 Route Cost = 952

Total Cost = 7223

E-n76-k14 – capacity = 100

0 6 33 73 1 22 61 0 Load = 97 Route Time=84 Route Cost = 166

0 34 52 27 57 15 20 0 Load = 99 Route Time=82 Route Cost =151

0 46 8 14 59 0 Load = 98 Route Time = 79 Route Cost = 98

0 30 48 47 36 71 60 70 0 Load = 100 Route Time=90 Route Cost = 229

0 68 2 62 43 64 0 Load = 100 Route Time=93 Route Cost = 129

0 17 32 50 18 55 0 Load = 90 Route Time=92 Route Cost = 146

0 67 7 35 19 54 13 0 Load = 98 Route Time= 66 Route Cost = 142

0 51 16 63 23 56 41 42 0 Load = 100 Route Time= 91 Route Cost = 239

0 40 44 3 49 24 0 Load = 93 Route Time= 77 Route Cost= 144

0 12 39 9 25 31 0 Load = 100 Route Time= 91 Route Cost = 150

0 75 74 28 21 69 0 Load = 95 Route Time= 83 Route Cost = 130

0 26 72 58 10 38 65 0 Load = 99 Route Time= 76 Route Cost =168

0 4 45 29 5 37 0 Load = 99 Route Time= 64 Route Cost = 96

0 53 11 66 0 Load = 96 Route Time= 74 Route Cost = 92

Total Cost = 2.080

B-n57-k7 – capacity= 100

0 54 35 40 29 4 41 17 0	Load= 100 Route Time= 187 Route Cost =312
0 46 23 27 24 49 44 21 10 3 0	Load= 100 Route Time= 139 Route Cost =499
0 48 45 11 31 6 15 0	Load= 99 Route Time= 193 Route Cost = 494
0 2 22 42 14 39 7 32 47 20 0	Load= 99 Route Time= 200 Route Cost =751
0 16 13 33 5 56 26 19 34 0	Load= 100 Route Time= 159 Route Cost = 510
0 1 55 30 52 50 37 43 0	Load= 99 Route Time= 202 Route Cost = 522
0 18 38 36 12 53 25 28 8 9 51 0	Load= 100 Route Time= 171 Route Cost = 795

Total Cost = 3.883

B-n66-k9 – capacity= 100

0 52 54 18 23 7 28 0	Load= 99 Route Time= 150 Route Cost= 220
0 41 20 53 17 9 21 0`	Load= 100 Route Time= 198 Route Cost= 256
0 32 58 33 65 8 40 0	Load=75 Route Time= 176 Route Cost= 464
0 42 27 5 14 39 29 3 15 60 4 0	Load=88 Route Time= 196 Route Cost= 773
0 1 55 11 57 38 19 35 59 0	Load=99 Route Time= 161 Route Cost= 379
0 45 31 13 56 22 43 34 51 0	Load=100 Route Time= 115 Route Cost= 440
0 49 12 24 6 16 2 0	Load=100 Route Time= 166 Route Cost= 438
0 30 37 48 25 64 44 46 26 0	Load=100 Route Time= 160 Route Cost= 602
0 50 47 61 62 63 10 36 0	Load=100 Route Time= 211 Route Cost= 548

Total Cost = 4.120

P-n51-k10 – capacity= 80

0 11 16 2 3 0	Load= 80 Route Time= 79 Route Cost= 111
0 27 8 26 31 28 0	Load= 70 Route Time= 75Route Cost=143
0 6 48 23 7 43 0	Load= 78 Route Time= 82 Route Cost= 142
0 46 12 37 44 42 40 0	Load= 79 Route Time= 94 Route Cost= 144
0 18 25 24 0	Load= 79 Route Time= 65 Route Cost= 81
0 5 49 10 30 39 0	Load= 77 Route Time= 89 Route Cost=156
0 47 4 17 15 45 33 0	Load= 80 Route Time= 84 Route Cost = 180
0 32 1 22 20 35 36 0	Load= 78 Route Time= 95 Route Cost 7 = 183
0 14 13 41 19 0	Load= 80 Route Time= 82 Route Cost = 149
0 38 9 50 34 21 29 0	Load= 76 Route Time= 80 Route Cost = 198

Total Cost = 1.487