ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΓΟΡΙΘΜΟΣ ΠΡΟΣΟΜΟΙΩΜΕΝΗΣ ΑΝΟΠΤΗΣΗΣ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΠΕΡΙΟΡΙΣΜΕΝΗ ΧΩΡΗΤΙΚΟΤΗΤΑ

ΕΜΜΑΝΟΥΗΛ ΒΟΥΤΣΑΚΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Δρ. ΙΩΑΝΝΗΣ ΜΑΡΙΝΑΚΗΣ

ΧΑΝΙΑ 2021

Περιεχόμενα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω, αρχικά ,τον καθηγητή μου κ. Ιωάννη Μαρινάκη για την καθοδήγηση και την υποστήριξη του για την υλοποίηση της παρούσας διπλωματικής εργασίας. Επίσης, ευχαριστώ θερμά τον κ. Λευτέρη Τσακιράκη, υποψήφιο διδάκτορα του Πολυτεχνείου Κρήτης για τις πολύτιμες συμβουλές, τις υποδείξεις του καθώς και για τον χρόνο που αφιέρωσε για την επίλυση των αποριών μου. Ακόμα θα ήθελα να ευχαριστήσω τους φίλους μου για τις υπέροχες στιγμές που ζήσαμε μαζί όλα αυτά τα χρόνια. Τέλος οφείλω τις μεγαλύτερες ευχαριστίες στην οικογένεια μου που ήταν πραγματικά δίπλα μου και με στήριξε καθόλη τη διάρκεια των σπουδών μου.

Περίληψη

Στην παρούσα διπλωματική εργασία ασχολούμαστε με το περιορισμένης χωρητικότητας πρόβλημα δρομολόγησης οχημάτων (Capacitated Vehicle Routing Problem). Στόχος μας είναι η ελαχιστοποίηση του κόστους κυκλικών διαδρομών που εξυπηρετούν τους πελάτες από την αφετηρία-αποθήκη. Το όχημα που εκτελεί το κάθε δρομολόγιο επιστρέφει στην αποθήκη μετά το πέρας κάθε διαδρομής. Η διαδρομή καθορίζεται από τους παράγοντες της χωρητικότητας και του μέγιστου επιτρεπτού χρονικού ορίου που μπορεί να διανύσει κάθε όχημα. Για την εξυπηρέτηση όλων των πελατών χρησιμοποιήσαμε τον αλγόριθμο του Πλησιέστερου Γείτονα, με τον οποίο βρίσκουμε μια αρχική αποδεκτή λύση και στη συνέχεια τη βελτιώνουμε μέσω τριών αλγορίθμων τοπικής αναζήτησης (1-0 επανατοποθέτηση, 1-1 ανταλλαγή, εσωτερική ανταλλαγή). Τέλος για να βελτιώσουμε ακόμα περισσότερο τη λύση μας χρησιμοποιήσαμε τον ευρετικό αλγόριθμο της Προσομοιωμένης Ανόπτησης (Simulated Annealing).

Κεφάλαιο 1 – Εισαγωγή

1.1 Ιστορική εξέλιξη της Εφοδιαστικής (Logistics)

Ο τομέας της εφοδιαστικής υπάρχει από τα αρχαία χρόνια και η ανάπτυξη του συνδέεται με διάφορους λαούς και πολιτισμούς. Η ετυμολογία του όρου ‘’Logistics’’ έχει προέλευση από την ελληνική λέξη ‘’λόγος’’, με την έννοια της λογικής για την επίτευξη συγκεκριμένων στόχων. Με βάση ιστορικά τεκμήρια η εφοδιαστική που πλησιάζει την σημερινή της έννοια, τοποθετείται στην Ρωμαϊκή εποχή. Εκείνη τη περίοδο οι λεγεώνες σε διάφορα σημεία των συνόρων της αυτοκρατορίας είχαν ανάγκη την έγκαιρη και την επιτυχημένη μεταφορά υλικών, τροφίμων, διαταγών και ανδρών για την αντιμετώπιση εχθρικών επιδρομών. Κατα πολλούς ιστορικούς, από τους πρώτους που εφάρμοσαν Logistics για τον καλύτερο εφοδιασμό των στρατευμάτων τους ήταν ο Μέγας Αλέξανδρος και ο Μέγας Ναπολέων. Σίγουρα όμως σε γενικό πλαίσιο για την ιστορική ανάπτυξη της εφοδιαστικής πρέπει να αναφερθούν οι πολιτισμοί των αρχαίων Ελλήνων, των Αιγυπτίων, των Φοινίκων και του Βυζαντίου, οι οποίοι είχαν δημιουργήσει μεταφορικά δίκτυα τα οποία χρησιμοποιούσαν για την εξέλιξη του εμπορίου τους καθώς και συστήματα καλής διαχείρισης αποθηκών και αποθεμάτων [1].

Επιχειρηματικά και επιστημονικά η συστηματική ενασχόληση με την εφοδιαστική ξεκινάει από τη δεκαετία του ’60 καθώς αναπτύχθηκαν οι τομείς της διοίκησης και της διανομής σε επίπεδο επιχειρήσεων. Στις επόμενες δεκαετίες, το φαινόμενο της παγκοσμιοποίησης με το ανταγωνιστικό εμπόριο, την ανάπτυξη των διεθνών αγορών και οικονομιών, αλλά και τη δημιουργία οργανισμών και συνεργασιών καθόρισαν σημαντικά την εξέλιξη των Logistics. Σήμερα η πρόοδος της τεχνολογίας και η ιλιγγιώδης ταχύτητα με την οποία εξελίσσεται η ηλεκτρονική αγορά μέσω του διαδικτύου έχει συμβάλει στην αναβάθμιση των εφαρμογών των Logistics στις επιχειρήσεις.

1.2 Εφοδιαστική (Logistics)

Σήμερα ο όρος Logistics χρησιμοποιείται αρκετά στη καθημερινότητα και αφορά την διαχείριση της ροής υλικών, προϊόντων και πληροφοριών από την πηγή στον καταναλωτή και αντίστροφα. Αυτή η διαχείριση της εφοδιαστικής έχει ως στόχο την δημιουργία ενός σχεδίου που ικανοποιεί αποτελεσματικά και οικονομικά τους εφοδιαστικούς στόχους μιας επιχείρησης [2]. Σκοπός λοιπόν των Logistics σε μια επιχείρηση είναι η παράδοση της κατάλληλης ποσότητας προϊόντος με το χαμηλότερο κόστος στην καλύτερη δυνατή ποιότητα.

Διάφορες περιοχές εφαρμογών των Logistics [1], όπως:

- Επιχειρηματική Εφοδιαστική (Business Logistics)
- Φυσική Διανομή (Physical Distribution)
- Διαχείριση Υλικού (Material Management)
- Στρατιωτική Εφοδιαστική (Military Logistics)
- Συστήματα ταχείας απόκρισης (Quick-Response Systems)
- Διαχείριση Διανομών (Distribution Management)
- Πληροφοριακά Συστήματα (Information Systems)
- Διαχείριση Εφοδιαστικής Αλυσίδας (Supply Chain Management)

1.3 Εφοδιαστική Αλυσίδα (Supply Chain)

Η Εφοδιαστική Αλυσίδα (Supply Chain) ή το Δίκτυο Εφοδιαστικής (Logistics Network) συγκροτείται από όλα τα στάδια που εμπλέκονται άμεσα ή έμμεσα στην ικανοποίηση του καταναλωτή. Η εφοδιαστική αλυσίδα είναι δυναμική, δηλαδή περιλαμβάνει τη ροή αγαθών, πληροφοριών και κεφαλαίων μεταξύ των διαφόρων σταδίων. Η αλυσίδα αυτή απαρτίζεται από τους προμηθευτές, τους κατασκευαστές, τις αποθήκες, τις εγκαταστάσεις διανομής, τους μεταφορείς, τους πωλητές λιανικής και τους πελάτες, αλλά και από τις πρώτες ύλες, τα διάφορα προϊόντα που βρίσκονται σε απόθεμα καθώς και από έτοιμα προϊόντα [1]. Τα στάδια σε κάθε εφοδιαστική αλυσίδα μπορεί να διαφέρουν,αυτό εξαρτάται από τον διαφορετικό σχεδιασμό των αλυσίδων, ο

οποίος διαμορφώνεται από τις απαιτήσεις των πελατών. Ο πελάτης είναι η μοναδική πηγή εσόδων σε κάθε εφοδιαστική αλυσίδα ενώ οι διάφορες ροές (υλικών, πληροφοριών, κεφαλαίων) σε κάθε στάδιο δημιουργούν έξοδα. Η απόδοση μιας εφοδιαστικής αλυσίδας εξαρτάται από τρία κύρια κριτήρια:

- Το ολικό κόστος (overall cost), το οποίο καλείται να ελαχιστοποιηθεί. Το κόστος αυτό προκύπτει από τη διανομή, τις μεταφορές κ.λ.π. Για να επιτευχθεί η μείωση του δε πρέπει να ελαχιστοποιηθεί το κόστος κάθε δραστηριότητας μεμονωμένα αλλά το κόστος κατά μήκος όλου του συστήματος. Για αυτό το λόγο είναι σημαντική η κατανόηση των δραστηριοτήτων και των αλληλεπιδράσεων μεταξύ του κάθε σταδίου της εφοδιαστικής αλυσίδας.

- Το ολικό κέρδος (overall profitability), το οποίο καλείται να μεγιστοποιήθει. Ομοίως με το ολικό κόστος θα πρέπει να μετρηθεί με βάση το κέρδος σε όλο το μήκος του συστήματος και όχι ξεχωριστά για κάθε στάδιο.

- Ο χρονικός κύκλος (cycle time), ο οποίος είναι ο συνολικός χρόνος μέχρι το πέρας της όλης διαδικασίας από τις πρώτες ύλες στο τελικό προϊόν στον καταναλωτή. Για την υλοποίηση της πραγματικής διαδικασίας, εκτιμάται ότι γίνεται χρήση μόνο του 5% του κύκλου. Όσο περισσότερο αυξάνεται η απόδοση του κύκλου μιας αλυσίδας τόσο πιο αποδοτική είναι. Είναι γεγονός όμως πως αυτή η βελτίωση δεν είναι δυνατόν να επιτευχθεί εύκολα καθώς μεταξύ των διαφορετικών στοιχείων της αλυσίδας εμφανίζονται διαφορετικοί στόχοι ενώ καθοριστικός είναι και ο χρονικός παράγοντας.

Ο βασικός στόχος της εφοδιαστικής αλυσίδας είναι η μεγιστοποίηση της ολικής αξίας (overall value), η οποία προκύπτει από την διαφορά μεταξύ της αξίας του έτοιμου προϊόντος που φτάνει στον καταναλωτή και της συνολικής διαδικασίας της εφοδιαστικής αλυσίδας για την υλοποίηση του [5]. Για να επιτευχθεί κάτι τέτοιο, σκοπός της διαχείρισης της εφοδιαστικής αλυσίδας πρέπει να είναι να καταστήσει δυνατή και αποτελεσματική ως προς το κόστος όλου του συστήματος.

Εικόνα 1.1 Ροή υλικών/προϊόντων και πληροφοριών στην εφοδιαστική αλυσίδα. [2]

1.4 Διαχείριση Εφοδιαστικής Αλυσίδας (ΔΕΑ)

Η Διαχείριση Εφοδιαστικής Αλυσίδας (ΔΕΑ) αφορά τη λήψη αποφάσεων για όλες τις δραστηριότητες μιας εφοδιαστικής αλυσίδας. Έτσι προκειμένου να επιτευχθεί ο κατάλληλος σχεδιασμός σε μια αλυσίδα ώστε να εξασφαλιστούν η αποδοτικότητα και η αποτελεσματικότητα της, είναι αναγκαίο να ληφθούν αποφάσεις για τη καλύτερη ροή υλικών, προϊόντων, πληροφοριών και κεφαλαίων [6]. Η διαχείριση περιλαμβάνει δύο επίπεδα:

- Τον προγραμματισμό, όπου γίνονται οι προβλέψεις με βάση τα δεδομένα ζήτησης και οργανώνονται τα πλάνα υλοποίησης.
- Την εκτέλεση, που τίθεται σε εφαρμογή το πλάνο που καθορίστηκε στο επίπεδο του προγραμματισμού και παρακολουθείται η εξέλιξη του, καθώς συλλέγονται πληροφορίες από όλο το εύρος της εφοδιαστικής αλυσίδας.

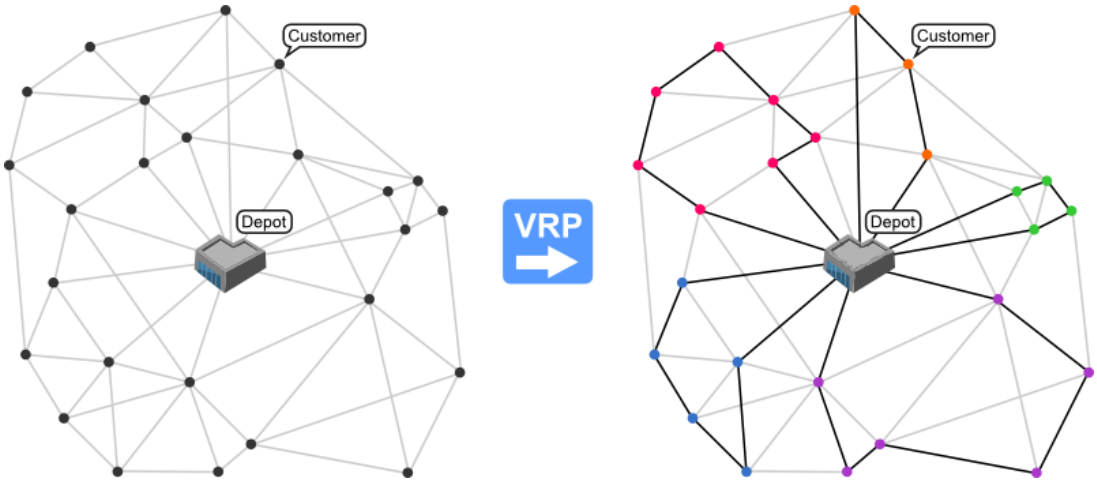
Η σωστή διαχείριση μιας εφοδιαστικης αλυσίδας είναι πολύ σημαντική για τη διαμόρφωση μια αποτελεσματικής επιχειρηματικής στρατηγικής.

Εικόνα 1.2 Στοιχεία της εφοδιαστικής διαχείρησης. [3]

Κεφάλαιο 2 – Προβλήματα Δρομολόγησης Οχημάτων

2.1 Το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem-VRP)

Η πρώτη μορφή του προβλήματος δρομολόγησης οχημάτων (Vehicle Routing Problem) αναφέρθηκε ως πρόβλημα αποστολής φορτηγών (truck dispatching problem) για πρώτη φορά από τους Dantzig και Ramser (1959). Νέες έρευνες, όπως αυτή των Clarke και Wright (1964) έδωσαν καλύτερες προσεγγίσεις και συνέβαλαν έτσι στη διαμόρφωση της μορφής του προβλήματος που χρησιμοποιείται μέχρι σήμερα. Το VRP χρησιμοποιείται για την ελαχιστοποίηση του κόστους σε ένα σύνολο κυκλικών διαδρομών που πραγματοποιεί ένα ή περισσότερα οχήματα, τα οποία εξυπηρετούν τους πελάτες από την αποθήκη-αφετηρία. Το όχημα κάθε φορά επισκέπτεται ένα υποσύνολο πελατών, περνώντας από τον καθένα ακριβώς μια φορά, έτσι ώστε όταν έχουν ολοκληρωθεί όλες οι κυκλικές διαδρομές να έχουν καλυφθεί ολοι οι πελάτες. Σκοπός του είναι η εξυπηρέτηση όλων των πελατών με το ελάχιστο εφικτό κόστος.



Εικόνα 2.1 Το γράφημα κυκλικων διαδρομών στο Πρόβλημα Δρομολόγησης Οχημάτων (VRP) πολλαπλών οχημάτων.

Το δίκτυο που χρησιμοποιείται για την μεταφορά των προϊόντων, περιγράφεται γενικά από ένα γράφημα, του οποίου τα τόξα αντιπροσωπεύουν τους δρόμους του δικτύου και κάθε κόμβος αντιπροσωπεύει έναν πελάτη (customer). Ο κόμβος από τον οποίο ξεκινούν όλα τα τόξα και το κάθε όχημα (vehicle) είναι η αποθήκη (depot) όπου γίνεται η φόρτωση και η εκφόρτωση των οχημάτων. Κάθε τόξο αντιστοιχεί σε ένα κόστος το οποίο σχετίζεται με την απόσταση που χρειάζεται για να το διασχίσει κάποιο όχημα.

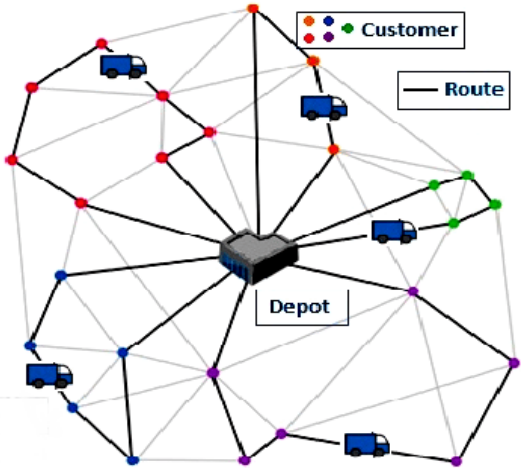
Αναφορικά κάποιες από τις παραλλαγές-επεκτάσεις του Προβλήματος Δρομολόγησης Οχημάτων που έχουν δημιουργηθεί από τους εκάστοτε περιορισμούς και τις ανάγκες των επιχειρήσεων:

- Το Πρόβλημα του Πλανόδιου Πωλητή (Travelling Salesman Problem-TSP)
- Το Περιορισμένης Χωρητικότητας Πρόβλημα Δρομολόγησης Οχημάτων (Capacitated VRP - CVRP)
- Το Πρόβλημα Δρομολόγησης Οχημάτων με Πολλαπλές Αποθήκες (Multidepot VRP - MDVRP)
- Το Ανοιχτό Πρόβλημα Δρομολόγησης Οχημάτων (Open VRP - OVRP)

- Το Ανοιχτό-Κλειστό Πρόβλημα Δρομολόγησης Οχημάτων (Close Open VRP - COVRP)
- Το Πρόβλημα Δρομολόγησης Οχημάτων μέσα σε Δεδομένα Χρονικά Περιθώρια (Χρονικά Παράθυρα) (VRP with Time Windows - VRPTW)

2.2 Το Περιορισμένης Χωρητικότητας Πρόβλημα Δρομολόγησης Οχημάτων (CVRP)

Μια από τις πιο συνήθεις παραλλαγές του απλού VRP είναι το Πρόβλημα Δρομολόγησης Οχημάτων, το οποίο περιέχει τους περιορισμούς της χωρητικότητας και του μέγιστου μήκους διαδρομής ή του μέγιστου επιτρεπτού χρονικού ορίου. Συγκεκριμένα η συνολική ζήτηση είναι ανέφικτο να καλυφθεί από μόνο ένα όχημα, οπότε το κάθε όχημα συνεχίζει τη διαδρομή του εως ότου να ξεπεραστεί η μέγιστη επιτρεπτή χωρητικότητα του. Όταν συμβεί αυτό, επιστρέφει στην αποθήκη για να ολοκληρώσει τη διαδρομή του και να ξεκινήσει ένα νέο άδειο όχημα μια νέα διαδρομή. Το ίδιο φυσικά συμβαίνει αν παραβιαστεί και ο άλλος περιορισμός, που αφορά την μέγιστη επιτρεπτή απόσταση ή το χρονικό όριο που μπορεί να υπάρχει σύμφωνα με τα δεδομένα. Τα δεδομένα αυτά, είναι το πλήθος των κόμβων (πελάτες προς εξυπηρέτηση) και η ζήτηση που αντιστοιχεί στον καθένα. Ακόμα οι συντεταγμένες τους, ώστε να είναι γνωστή η τοποθεσία τους για τον υπολογισμό των αποστάσεων. Επίσης, πρέπει να δίνεται το πλήθος των οχημάτων που θα χρησιμοποιηθούν με τη χωρητικότητα που αντιστοιχεί στο καθένα, αλλά και ο μέγιστος χρόνος ή η μέγιστη δυνατή απόσταση που μπορεί να διατεθεί. Στο CVRP ο σκοπός είναι ο ίδιος με το απλό VRP, δηλαδή να εξυπηρετηθούν όλοι οι πελάτες με το ελάχιστο δυνατό κόστος διαδρομής, δηλαδή βελτιστοποιώντας τον αριθμό των δρομολογίων που πρέπει να εκτελεστούν ή το μήκος ή τον χρόνο που χρειάζεται για να ολοκληρωθεί η διαδικασία.



Εικόνα 2.2 Γράφημα του Περιορισμένης Χωρητικότητας Προβλήματος Δρομολόγησης Οχημάτων (CVRP) .

Η μοντελοποίηση του προβλήματος όπως παρουσιάστηκε από τους Fisher και Jaikumar:

Έστω:

$$x_{ijk} = \begin{cases} 1, & \text{εάν το οχημα } k \text{ επισκέπτεται τον πελάτη } j \text{ αμέσως μετά τον } i \\ 0, & \text{αλλιώς} \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{εάν ο πελάτης } i \text{ επισκέπτεται απο το όχημα } k \\ 0, & \text{αλλιώς} \end{cases}$$

Αντικειμενική συνάρτηση :

$\min \sum_{i,j} c_{i,j} \sum_k x_{ijk}$

υπο περιορισμούς:

$$\sum_k y_{ik} = \begin{cases} 1, i = 1,2, \dots, n \\ m, \quad i = 1 \end{cases} \qquad (1)$$

$$\sum_i q_i y_{ij} \leq Q_k \qquad k = 1, \dots, m \qquad (2)$$

$$\sum_j x_{ijk} = \sum_j x_{jik} = y_{ik} \quad i = 1, \dots, n \quad (3)$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \quad \gamma\iota\alpha\ \acute{o}\lambda\alpha\ \tau\alpha\ S \subseteq \{2, \dots, n\}, \quad k = 1, \dots, m \quad (4)$$

$$y_{ik} \in \{0,1\}, \quad \begin{matrix} k = 1, \dots, m \\ i = 1, \dots, n \end{matrix} \quad (5)$$

$$x_{ijk} \in \{0,1\}, \quad \begin{matrix} i, j = 1, \dots, n \\ k = 1, \dots, m \end{matrix} \quad (6)$$

Ο περιορισμός (1) εξασφαλίζει ότι κάθε πελάτης εξυπηρετείται από ένα μόνο όχημα με εξαίρεση την αποθήκη την οποία επισκέπτονται όλα τα οχήματα. Ο περιορισμός (2) αποτελεί τον περιορισμό της χωρητικότητας. Ο περιορισμός (3) δηλώνει ότι το όχημα που πάει σε έναν πελάτη φεύγει από τον ίδιο πελάτη. Ο περιορισμός (4) φανερώνει ότι το όχημα έχει επισκευτεί όλους τους κόμβους εκτός της αποθήκης. Οι περιορισμοι (5) και (6) δείχνουν ότι οι μεταβλητές x, y είναι ακέραιες και μη αρνητικές [1].

Κεφάλαιο 3 – Μέθοδοι-Αλγόριθμοι Βελτιστοποίησης

3.1 Απλοί Ευρετικοί Αλγόριθμοι (Heuristics)

Τα προβλήματα συνδυαστικής βελτιστοποίησης πολλές φορές είναι ιδιαίτερα δύσκολο έως σχεδόν αδύνατο να επιλυθούν σε βαθμό βέλτιστου αποτελέσματος. Αυτό συνήθως οφείλεται στο μέγεθος του προβλήματος, το οποίο όσο αυξάνεται, τόσο δυσκολότερη είναι η βελτιστοποίηση της λύσης. Για αυτό τον λόγο χρησιμοποιούνται διαφορετικές τεχνικές που διευκολύνουν την εύρεση μιας σχεδόν βέλτιστης και ταυτόχρονα επαρκή λύση. Προκειμένου να θεωρηθεί μια λύση ενός ευρετικού αλγορίθμου αποδεκτή πρέπει να ικανοποιούνται κάποια κριτήρια όπως η απόκλιση της από τη βέλτιστη (ποιότητα λύσης), η ευχέρεια εύρεσης λύσης, η ύπαρξη λογικής για τους κανόνες που ορίζουν τον ευρετικό αλγόριθμο που οδήγησε στη λύση [1].
Γενικά για τα προβλήματα που στοχεύουν στην εύρεση βέλτιστης λύσης έχουν δημιουργηθεί διάφοροι αλγόριθμοι, οι οποίοι οδηγούν στη βελτίωση της λύσης.

Οι κατηγορίες τους είναι:

- Αλγόριθμοι απληστίας (greedy algorithms), οι οποίοι παρόλο που θέλουν να οδηγήσουν σε μια καλύτερη λύση του προβλήματος, χρειάζονται πάρα πολύ χρόνο.

- Προσεγγιστικοί αλγόριθμοι (approximation algorithms), κάνουν χρήση περισσότερων πληροφοριών για την επίλυση του προβλήματος.

- Αλγόριθμοι τοπικής αναζήτησης (local search algorithms), που ψάχνουν να βελτιώσουν μια αρχική λύση αναζητώντας στη γειτονιά της λύσης.

Στην παρούσα διπλωματική έγινε χρήση των αλγορίθμων τοπικής αναζήτησης για τη βελτίωση της αρχικής λύσης.

3.2 Αλγόριθμοι Τοπικής Αναζήτησης (Local Search)

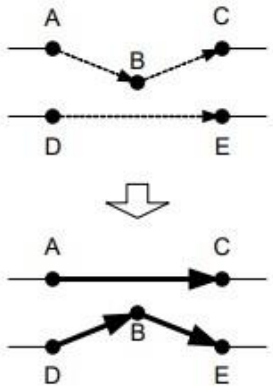
Η βάση της τοπικής αναζήτησης θεωρείται μια από τις παλαιότερες μεθόδους βελτιστοποίησης, πρόκειται για τη μέθοδο δοκιμής και σφάλματος. Αξίζει να επισημανθεί πως παρά την απλότητα της κύριας ιδέας στην οποία στηρίζεται η τοπική αναζήτηση είναι αρκετά επιτυχημένη για την βελτίωση των αρχικών λύσεων. Οι αλγόριθμοι της συγκεκριμένης κατηγορίας ξεκινούν από την αρχική λυση και επαναληπτικά αναζητούν στη γειτονιά της λύσης μέχρι να βρεθεί καλύτερο εφικτό αποτέλεσμα. Για αυτό τον λόγο είναι πολύ σημαντική η επιλογή της κατάλληλης γειτονιάς που θα γίνει η αναζήτηση. Όμως το πιο σημαντικό στοιχείο για τη διαδικασία είναι η επιλογή της μεθόδου που θα χρησιμοποιηθεί για την αναζήτηση. Μερικές απο τις πιο διαδεδομένες μέθοδοι είναι η ‘’2-opt’’, η ‘’3-opt’’, η ‘’1-0 relocate’’ κ.α. Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκαν τρεις μέθοδοι τοπικής αναζήτησης: η ‘’1-0 relocate’’, η ‘’1-1 exchange’’ και η ‘’swap’’ οι οποίες θα αναλυθούν παρακάτω.

Ένας γενικός αλγόριθμος τοπικής αναζήτησης προγραμματιστικά:

διαδικασία *local_search*
begin
  t μια αρχική λύση του προβλήματος
  do while *βρίσκεται μια βελτιωμένη λύση (improve(t))*
    t=improve(t)
    return t
end

3.2.1 Μέθοδος 1-0 relocate (1-0 επανατοποθέτηση)

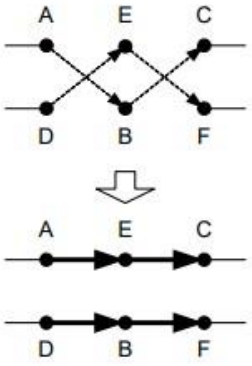
Η μέθοδος 1-0 επανατοποθέτησης (1-0 relocate) προτάθηκε από τον Waters και στόχος της είναι η απαλοιφή περιττών διαδρομών της αρχικής λύσης. Η διαδικασία επανατοποθέτησης γίνεται μεταξύ δύο διαδρομών, όπου διαγράφεται ένας κόμβος από μια διαδρομή και τοποθετείται σε μια άλλη, κάτι που επιφέρει μείωση στο κόστος των διαδρομών και στο ολικό κόστος.



Εικόνα 3.1 Παράδειγμα 1-0 relocate.[10]

3.2.2 Μέθοδος 1-1 exchange (1-1 ανταλλαγή)

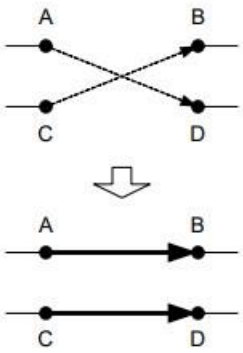
Η μέθοδος 1-1 ανταλλαγής (1-1 exchange) προτάθηκε επίσης από τον Waters και γίνεται μεταξύ δύο κόμβων δύο διαφορετικών τόξων, οι οποίοι ανταλλάζουν θέση. Συνεπώς διαγράφεται ο ένας και στη θέση του τοποθετείται ο άλλος και για τις δύο διαδρομές.



Εικόνα 3.2 Παράδειγμα 1-1 exchange.[10]

3.2.3 Μέθοδος opt

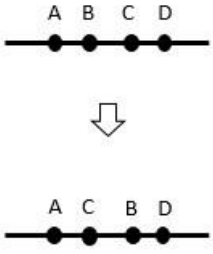
Η μέθοδος opt προτάθηκε από τον Croes το 1958. Για την εφαρμογή της χρειάζεται μόνο μία διαδρομή, στην οποία καταργούνται ένα ή περισσότερα τόξα που μικραίνουν το μέγεθος της και δημιουργούν νέες ακμές που συνδέουν τους κόμβους της με αποτέλεσμα τη μείωση του συνολικού κόστους της. Αν δημιουργηθεί μόνο ένα νέο τόξο έχουμε την μέθοδο 1-opt. Αν σχηματιστούν δύο νέα τόξα έχουμε την μέθοδο την 2-opt. Αν δημιουργηθούν τρία νέα τόξα έχουμε την 3-opt μέθοδο.



Εικόνα 3.3 Παράδειγμα 2-opt.[10]

3.2.4 Μέθοδος swap (εσωτερική ανταλλαγή)

Η μέθοδος swap μοιάζει σε μεγάλο βαθμό με εκείνη της opt. Ειδικότερα εφαρμόζεται με την ανταλλαγή δύο κόμβων μέσα στην ίδια διαδρομή και με αυτό τον τρόπο κατασκευάζονται νέα τόξα, δημιουργώντας ένα νέο κόστος της διαδρομής. Χρησιμοποιώντας την επαναλαμβανόμενα γίνεται προσπάθεια για να ελαχιστοποιήθει το ολικό κόστος.



Εικόνα 3.4 Παράδειγμα swap.

3.3 Μεθευρετικοί Αλγόριθμοι (Metaheuristics)

Οι ευρετικοί αλγόριθμοι αντιμετωπίζουν πρόβλημα στην αποφυγή τοπικών ελαχίστων ή μεγίστων (αναλόγως το είδος του προβλήματος), δηλαδή μετά από ένα σημείο δε μπορούν να βελτιώσουν παραπάνω τη λύση. Αυτός είναι ο κύριος λόγος δημιουργίας των μεθευρετικών αλγορίθμων που καταφέρνουν να συνδυάσουν διάφορες στρατηγικές τοπικής αναζήτησης ώστε ο αλγόριθμος να ξεπερνάει και να ξεφεύγει από τοπικά ελάχιστα [1].

Στη σημερινή εποχή το μεγαλύτερο ποσοστό προβλημάτων βελτιστοποίησης λύνεται με τη βοήθεια μεθευρετικών αλγορίθμων. Η διαδικασία που υλοποιείται για την εύρεση της καλύτερης λύσης γίνεται αναζήτηση στο πεδίο της λύσης. Γεγονός είναι, ότι στις διαδικασίες τους χρησιμοποιούνται συχνά κλασικοί ευρετικοί αλγόριθμοι. Υπάρχουν πολλοί τρόποι για να κατηγοριοποιηθούν οι μεθευρετικοί αλγόριθμοι. ’Ενας τρόπος είναι με βάση το πλήθος λύσεων που χρησιμοποιούν. Κάποιοι από τους μεθευρετικούς αναζητούν τη νέα λύση με τη χρήση μιας αρχικής λύσης ενώ άλλοι χρησιμοποιώντας μια αρχική λύση. Βέβαια υπάρχουν και υβριδικές μορφές που δεν εστιάζουν μόνο σε μια κατηγορία. Ακόμα σε αυτό το είδος αλγορίθμων αρκετές φορές κατά την εξέλιξη της διαδικασίας μπορεί να παρουσιάζονται κάποιες μη εφικτές λύσεις ώστε να αποφευχθούν τυχόν τοπικά ελάχιστα που θα σταματήσουν την εύρεση της καλύτερης τελικής λύσης.

Οι μεθευρετικοί αλγόριθμοι έχουν το χαρακτηριστικό να μοντελοποιούν εφαρμογές και φαινόμενα της φύσης. Ακόμα μπορούν να μεταφερθούν σε παράλληλη μορφή και να προσαρμοστούν με ιδιαίτερη ευκολία.

Διάφοροι μεθευρετικοί αλγόριθμοι που χρησιμοποιούνται συχνά είναι:

- Η Προσομοιωμένη Ανόπτηση (Simulated Annealing - SA)
- Η Περιορισμένη Αναζήτηση (Tabu Search - TS)
- Οι Γενετικοί και Εξελικτικοί Αλγόριθμοι (Genetic and Evolutionary Algorithms)
- Ο Αλγόριθμος της Διασκορπισμένης Αναζήτησης (Scatter Search)
- Ο Αλγόριθμος Βελτιστοποίησης Αποικίας Μυρμιγκιών (Ant Colony Optimization)

- Η Διαδικασία Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure - GRASP)

Στη παρούσα διπλωματική εργασία εφαρμόστηκε ο μεθευρετικός αλγόριθμος της Προσομοιωμένης Ανόπτησης (SA) που αναλύεται ειδικότερα στο επόμενο κεφάλαιο.

3.4 Εξελικτικοί Αλγόριθμοι

Οι πρώτος γενετικός αλγόριθμος που αποτελεί πρόγονο των σημερινών εξελικτικών αλγορίθμων δημιουργήθηκε από τον John Holland τη δεκαετία του 60'. Οι εξελικτικοί αλγόριθμοι χρησιμοποιούν μοντέλα τα οποία βασίζονται στη φυσική εξέλιξη. Ειδικότερα, μιμούνται την έννοια της φυσικής επιλογής, της φυσικής γενετικής και της βιολογικής εξέλιξης δίνοντας καλύτερες τελικές λύσεις σε προβλήματα βελτιστοποίησης [1].

Έτσι, η επιβίωση του ικανότερου (δηλαδή της καλύτερης λύσης μέσα σε ένα πληθυσμό) συνδυάζεται με την οργανωμένη ανταλλαγή πληροφοριών (γονιδίων ή μετάλλαξη) με σκοπό την εύρεση της βέλτιστης λύσης.

Οι κυριότεροι από τους εξελικτικούς αλγόριθμους είναι:

- Οι Γενετικοί Αλγόριθμοι (Genetic Algorithms)
- Οι Γενετικοί Αλγόριθμοι Πολλαπλών Πληθυσμών – Νησιών (Island Genetic Algorithms)
- Οι Υβριδικοί Γενετικοί Αλγόριθμοι ή Μιμητικοί (Hybrid Genetic Algorithms or Memetic Algorithms)
- Η Διαφορική Εξέλιξη (Differential Evolution)

Κεφάλαιο 4 – Προσομοιωμένη Ανόπτηση (Simulated Annealing)

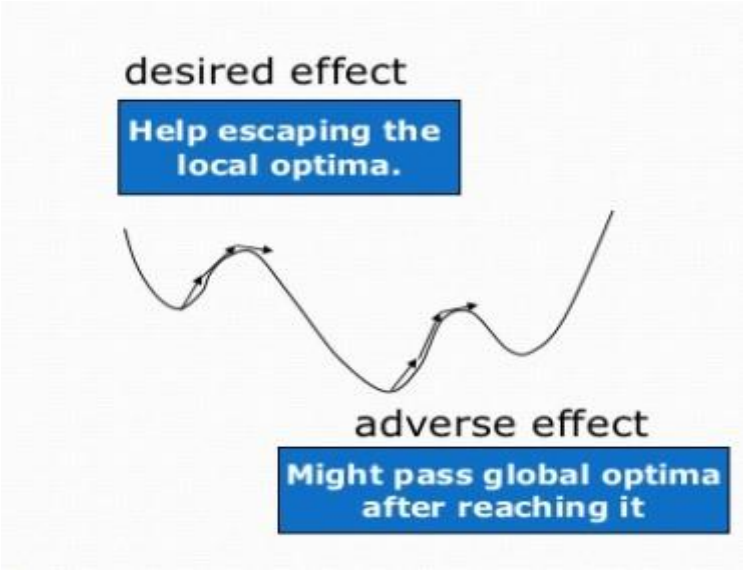
4.1 Γενική Περιγραφή

Για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης μια ιδιαίτερα δημοφιλής μέθοδος που χρησιμοποιείται είναι η προσομοιωμένη ανόπτηση (simulated annealing). Η μέθοδος αυτή προτάθηκε από τον Kirkpatrick το 1983 και ονομάστηκε έτσι από τη φυσική διεργασία της ανόπτησης των μετάλλων στην οποία βασίζεται. Ο όρος ‘’ανόπτηση’’ εντοπίζεται στη θερμοδυναμική και συγκεκριμένα πρόκειται για τη διαδικασία κατά την οποία ένα στερεό σώμα θερμαίνεται ξεπερνώντας το σημείο τήξεως του και έπειτα ψύχεται ξανά με αργό ρυθμό (ο ρυθμός ψύξης αφορά τη διαμόρφωση της δομής του υλικού). Με αυτόν τον τρόπο όταν σταματήσει η ψύξη τότε το στάδιο της ενέργειας του υλικού είναι το πιο χαμηλό.Ακολουθώντας το συγκεκριμένο μοντέλο διαμορφώνεται ο αλγόριθμος της προσομοιωμένης ανόπτησης, δηλαδή όταν εκτελείται ο αλγόριθμος, αυτό αναπαρίσταται με την κίνηση στον εφικτό χώρο αναζήτησης, ενώ όταν σταματήσει σημαίνει πως έχει βρεθεί μια εφικτή λύση.

Ο συγκεκριμένος αλγόριθμος περιλαμβάνει μία γειτονιά N(s) με νέες καταστάσεις, που μπορεί να οδηγηθεί κάποιος από την τρέχουσα κατάσταση s. Η προσομοιωμένη ανόπτηση χρησιμοποιεί μια αρχική κατάσταση την λεγόμενη s_0 και με μια τυχαία διαταραχή του συστήματος δημιουργείται μια νέα λύση s'. Το αποτέλεσμα της μεταβολής της τιμής αυτής δίνεται ως $\delta=f(s')-f(s_0)$. Αν το δ είναι αρνητικό τότε η νέα κατάσταση είναι πάντα αποδεκτή, εάν όμως $\delta \geq 0$, τότε η νέα κατάσταση είναι αποδεκτή με κάποια πιθανότητα, η οποία υπολογίζεται από τον νόμο της στατιστικής θερμοδυναμικής. Με βάση την θερμοκρασία t, η πιθανότητα αύξησης της ενέργειας της ποσότητας δ, αποτυπώνεται ως $p(\delta)=e^{-k\delta/t}$ (οπου k, η σταθερά Boltzmann αλλά στη συνδυαστική βελτιστοποίηση δεν χρειάζεται και για αυτό τον σκοπό μπορεί να παραληφθεί). Έτσι η πιθανότητα είναι $p(\delta)=e^{-\delta/kt}$.

Η προσομοιωμένη ανόπτηση κάνει χρήση ενός μηχανισμού ωστέ να αποδέχεται μικρές αυξήσεις στη τιμή του δ, προκειμένου να ελέγχει την πιθανότητα αποδοχης. Με αυτό

τον τρόπο καταφέρνει να μην εγκλωβίζεται σε τοπικά ακρότατα και να αναζητά λύσεις σε μεγαλύτερο πεδίο της γειτονιάς [9].



Εικόνα 4.1 Αποφυγή εγκλωβισμού σε τοπικό ακροτάτο μέσω της SA και πιθανότητα να χαθεί το βέλτιστο όταν γίνει χειρότερη μια λύση.

Αξίζει να σημειωθεί πως σε μεγαλύτερες θερμοκρασίες είναι πιο πιθανό μια λύση με χειρότερη τιμή στη συνάρτηση κόστους να είναι αποδεκτή ως η νέα κατάσταση του προβλήματος. Για να μειώνεται η τιμή της θερμοκρασίας (ψύχρανση), χρησιμοποιείται πάντοτε μια συνάρτηση $\alpha(t)$ και αναπαριστάται με τέσσερις στρατηγικές, οι οποίες είναι:

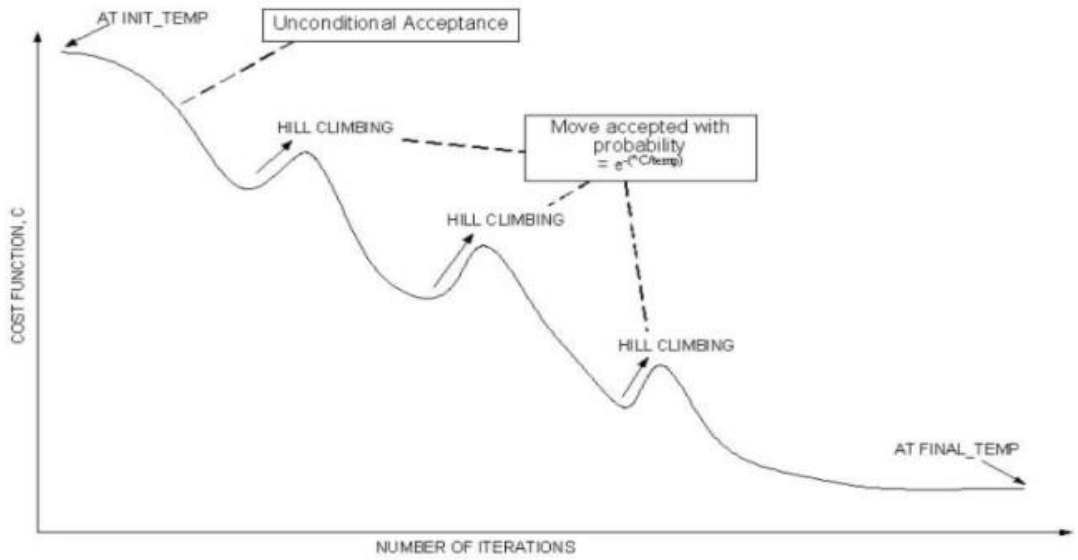
- η εκθετική μείωση με μορφή $\alpha(t)=t\times e^{-a\times i}$ ή $\alpha(t)=t\times a^{i}$ ή $\alpha(t)=t\times a$, όπου $\alpha=0.95$ ή $\alpha=0.98$.
- το προσαρμοστικό πρόγραμμα ψύχρανσης, το οποίο με βάση την ολοένα και αυξανόμενη ανακάλυψη νέων χαρακτηριστικών στο πεδίο αναζήτησης κάνει δυναμική επιλογή του t.
- μια καθορισμένου μήκους με μορφή $\alpha(t)=a\times t$, όπου $0{,}8\leq\alpha\leq0{,}98$.
- η λογαριθμική προσέγγιση, η οποία όμως δεν χρησιμοποιείται συχνά λόγω των πολλών υπολογισμών που περιλαμβάνει.

Η προσομοιωμένη ανόπτηση έχει φτάσει σε σημείο σύγκλισης όταν έχει φτάσει σε ένα από τα παρακάτω κριτήρια:

- όταν η θερμοκρασία φτάνει σε ένα συγκεκριμένο χαμηλό επίπεδο.
- όταν δεν έχει βρεθεί αποδεκτή λύση μετά από ένα πλήθος επαναλήψεων κατά τις οποίες υπάρχει η προοδευτική μείωση της θερμοκρασίας.
- όταν έχει ξεπεραστεί ο αριθμός των αποδεκτών κινήσεων.
- όταν έχουν ολοκληρωθεί οι επαναλήψεις του αλγορίθμου.

Τέλος, κάποιοι παράμετροι που μπορούν να καθοριστούν κατά τη διάρκεια της διαδικασίας είναι το πεδίο αναζήτησης της λύσης, η συνάρτηση ψύχρανσης ή το πλήθος των εσωτερικών επαναλήψεων. Ρυθμίζοντας τους με κατάλληλο τρόπο επιτυγχάνεται αλλαγή τόσο στα αποτελέσματα όσο και στην ταχύτητα της διαδικασίας του προγράμματος.

Ένα καλό παράδειγμα κατανόησης για τη σύγκλιση της μεθόδου προσομοιωμένης ανόπτησης είναι η αναρρίχηση.



Εικόνα 4.2 Παράδειγμα του αλγορίθμου προσομοιωμένης ανόπτησης κατά την αναρρίχηση. [9]

4.2 Ο Ψευδοκώδικας του Αλγορίθμου

Το πρόγραμμα του αλγορίθμου της προσομοιωμένης ανόπτησης (simulated annealing) σε ψευδοκώδικα είναι [1]:

select μία αρχική λύση s_0

select μία αρχική θερμοκρασία t_0

select μία συνάρτηση μείωσης της θερμοκρασίας $\alpha(t)$

repeat

repeat

Τυχαία επιλογή μιας γειτονιάς $s \in N(s_0)$

$\delta = f(s)-f(s_0)$

if $\delta<0$ **then**

$s_0=s$

else

δημιουργούμε τυχαία x, ομοιόμορφα κατανεμημένα στην ακτίνα (0,1)

if $x<e^{-\delta/t}$ **then**

$s_0=s$

endif

endif

until ο μέγιστος αριθμός επαναλήψεων ολοκληρωθεί

$t=a(t)$

until κάποιο κριτήριο τερματισμού ικανοποιηθεί

Κεφάλαιο 5 – Περιγραφή και Ανάλυση του Προβλήματος

5.1 Εισαγωγή στο Πρόβλημα

Στη παρούσα διπλωματική εργασία επιλύεται το περιορισμένης χωρητικότητας πρόβλημα δρομολόγησης οχημάτων (CVRP) μέσω αλγορίθμων για τη προσέγγιση των βέλτιστων λύσεων ορισμένων επιλεγμένων σεναρίων. Το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε για την κατασκευή, την επεξεργασία και την υλοποίηση του κώδικα του συγκεκριμένου προβλήματος είναι MATLAB R2019a. Στόχος είναι η ελαχιστοποίηση του κόστους των κυκλικών διαδρομών από την αποθήκη (αφετηρία) για την εξυπηρέτηση όλων των πελατών (κόμβοι) με επιστροφή του οχήματος σε αυτή μετά το πέρας κάθε δρομολογιού (διαδρομής). Το όχημα αναγκάζεται να ολοκληρώσει τη διαδρομή του όταν παραβιαστεί ένας από τους περιορισμούς του προβλήματος ή αν έχει περάσει από όλους τους κόμβους από μία φορά , δηλαδή έχουν εξυπηρετηθεί όλοι οι πελάτες. Οι περιορισμοί του συγκεκριμένου προβλήματος είναι η μέγιστη χωρητικότητας του οχήματος και το μέγιστο επιτρεπτό χρονικό όριο που μπορεί να διανύσει κάθε όχημα. Η κατασκευή του κώδικα για την επίτευξη του στόχου ,περιλαμβάνει τρία στάδια. Στο πρώτο στάδιο εφαρμόζεται ο αλγόριθμος του Πλησιέστερου γείτονα για την εύρεση μιας αρχικής λύσης. Στη συνέχεια, στο δεύτερο, χρησιμοποιούνται τρείς αλγόριθμοι τοπικής αναζήτησης (1-0 relocate, 1-1 exchange, swap) για τη βελτίωση της αρχικής λύσης. Τέλος στο τρίτο στάδιο, στο οποίο γίνεται εφαρμογή του μεθευρετικού αλγορίθμου της Προσομοιωμένης Ανόπτησης (SA) για τη περαιτέρω βελτίωση της λύσης, ώστε το τελικό αποτέλεσμα να προσεγγίζει ικανοποιητικά τις ήδη υπάρχουσες βέλτιστες λύσεις των σεναρίων. Έτσι λοιπόν στο παρόν κεφάλαιο γίνεται μια αναλυτική περιγραφή του κώδικα που κατασκευάστηκε για την επίλυση του προτεινόμενου προβλήματος.

5.2 Εισαγωγή και Ανάλυση Δεδομένων

Πρίν ξεκινήσει η υλοποίηση του κυρίως κώδικα, το πρόγραμμα ‘’διαβάζει’’ κάποια απαραίτητα δεδομένα τα οποία εισάγονται μέσω αρχείου .xls. Εκεί περιλαμβάνονται σε ξεχωριστά φύλλα, οι συντεταγμένες που δείχνουν τη θέση των πελατών (κόμβων), η ζήτηση του κάθε πελάτη (ο κόμβος που αντιπροσωπεύει την αποθήκη-αφετηρία έχει μηδενική ζήτηση), ο χρόνος εξυπηρέτησης του κάθε πελάτη. Επίσης σε ένα άλλο φύλλο αναφέρεται η μέγιστη επιτρεπτή χωρητικότητα και ο μέγιστος επιτρεπτός χρόνος του κάθε οχήματος. Για την προτεινόμενη εφαρμογή του αλγορίθμου και την επίτευξη του στόχου της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκαν 14 παραδείγματα (σενάρια) των Christofides, Mingozzi και Toth (1979) [CMT], των οποίων η βέλτιστη μέχρι στιγμής λύση είναι γνωστή. Αυτά αποθηκεύτηκαν σε ξεχωριστά αρχεία .xlsx καθώς τα δεδομένα είναι διαφορετικά για κάθε παράδειγμα.

Αρχικά μέσω της εντολής xlsread το πρόγραμμα διαβάζει τα παραπάνω δεδομένα από το excel και τα τοποθετεί στις μεταβλητές:

- ‘’Synt’’ οι συντεταγμένες τοποθεσίας των πελατών (κόμβων)
- ‘’Zitisi’’ η ζήτηση του κάθε πελάτη (ο πρώτος έχει μηδενική ζήτηση άρα είναι η αποθήκη από την οποία ξεκινούν τα οχήματα)
- ‘’servicetime’’ ο χρόνος εξυπηρέτησης
- ‘’Komvoi’’ το πλήθος των κόμβων (πλήθος πελατών)
- ‘’Qmax’’ η μέγιστη επιτρεπτή χωρητικότητα
- ‘’Tmax’’ ο μέγιστος επιτρεπτός χρόνος του κάθε οχήματος

Στη συνέχεια υπολογίζονται οι αποστάσεις μεταξύ των πελατών μέσω των συντεταγμένων. Η απόσταση μεταξύ δύο κόμβων στο συγκεκριμένο πρόβλημα αντιπροσωπεύει και το κόστος μετάβασης από τον ένα στον άλλον. Έτσι δημιουργείται το διάνυσμα Cost(i,j), που αναπαριστά το κόστος από τον κόμβο i στον κόμβο j. Ο υπολογισμός του για όλους τους κόμβους γίνεται με τη βοήθεια του τύπου της Ευκλείδειας απόστασης:

$$Cost(i,j) = \sqrt{(Synt(i,1) - Synt(j,1))^2 - (Synt(i,2) - Synt(j,2))^2}$$

Τα κόστη αυτά καταχωρούνται σε έναν πίνακα αποστάσεων Cost, ο οποίος αρχικά είναι μηδενικός με διαστάσεις (‘’Komvoi’’ * ‘’Komvoi’’). Έπειτα ο πίνακας αυτός τετραγωνοποιείται, δηλαδή γίνεται συμμετρικός, (Cost(i,j)=Cost(j,i)), ενώ η κύρια διαγώνιος του αποτελείται μόνο από μηδενικά στοιχεία, λόγω της μη δυνατής επιλογής να παραμείνει ένα όχημα σε κάποιον άλλο κόμβο πέρα της αποθήκης.

5.3 Εύρεση Αρχικής Λύσης – Αλγόριθμος Πλησιέστερου Γείτονα

Σε πρώτο στάδιο στόχος είναι να βρεθεί μια αρχική λύση, ώστε να βελτιστοποιηθεί στη συνέχεια και να πλησιάσει τις μέχρι τώρα βέλτιστες υπάρχουσες λύσεις του κάθε παραδείγματος CMT για το CVRP. Η αρχική λύση που αναζητείται είναι ένα πλήθος εφικτών διαδρομών που θα ικανοποιεί τους περιορισμούς του προβλήματος.

Προκειμένου να επιτευχθεί κάτι τέτοιο, χρησιμοποιείται ο αλγόριθμος του Πλησιέστερου Γείτονα (Nearest Neighborhood Algorithm). Ο συγκεκριμένος ανήκει στην κατηγορία των αλγορίθμων απληστίας (greedy algorithms) δηλαδή πρόκειται για έναν απλό ευρετικό αλγόριθμο. Θεωρείται ένας από τους πιο χρήσιμους για τη δημιουργία αρχικής αποδεκτής λύσης. Η διαδικασία που ακολουθείται είναι πως ένας προμηθευτής (όχημα) ξεκινάει από έναν κόμβο και δημιουργεί μια διαδρομή εξυπηρετώντας κάθε φορά τον πλησιέστερο από τον τελευταίο κόμβο. Έτσι δημιουργείται ένα μονοπάτι το οποίο πρέπει στο τέλος να περιλαμβάνει όλους τους πελάτες. Πολλές φορές, όπως στη περίπτωση του προτεινόμενου προβλήματος της συγκεκριμένης εργασίας, υπάρχουν περιορισμοί που αναγκάζουν το όχημα να επιστρέψει στην αποθήκη και να ξεκινήσει ξανά μια νέα διαδρομή, δημιουργώντας έτσι μια σειρά μονοπατιών, η οποία αποτελεί το τελικό δρομολόγιο.

Σε πρώτη φάση αρχικοποιούνται οι μεταβλητές που θα χρησιμοποιηθούν:

- elegxmenoi: ένας μηδενικός βοηθητικός πίνακας, για την ακρίβεια διάνυσμα γραμμής με τους πελάτες (κόμβους) που έχουν ελεγχθεί. Κάθε κόμβος που ελέγχεται παίρνει τη τιμή ‘’1’’ στην αντίστοιχη θέση του συγκεκριμένου

διανύσματος. Η αφετηρία, δηλαδή στα συγκεκριμένα προβλήματα η αποθήκη είναι ο πρώτος κόμβος και έχει εξ' αρχής τη τιμή ''1''.

- checked: πρόκειται για έναν μετρητή, ο οποίος συγκρατεί το πλήθος των κόμβων που έχουν ελεγχθεί.

- diadromes: μετρητής για το πλήθος των δρομολογίων-διαδρομών.

- thesi: δείκτης της θέσης ενός κόμβου σε μία διαδρομή.

- seira: πίνακας που εξαρτάται από τις ''diadromes'' και τη ''thesi'' και παρουσιάζει τη σειρά με την οποία εξυπηρετούνται όλοι οι πελάτες.

- TotalDemand: η ζήτηση μια διαδρομής. Αυτή αποθηκεύεται σε ένα αρχικά μηδενικό διάνυσμα στήλης με όνομα ''RDemand'' το οποίο περιλαμβάνει τη συνολική ζήτηση κάθε διαδρομής.

- Rroutetime: η χρονική διάρκεια μιας διαδρομής. Αυτή αποθηκεύεται σε ένα αρχικά μηδενικό διάνυσμα στήλης με όνομα ''EachRroutetime'' το οποίο εμπεριέχει το συνολικό χρόνο κάθε διαδρομής.

- routecost: το κόστος της κάθε διαδρομής. Αυτό αποθηκεύεται σε ένα αρχικά μηδενικό διάνυσμα στήλης μεγέθους όσο το πλήθος των κόμβων του εκάστοτε παραδείγματος, το ''Routecost''.

- routetime: η χρονική διάρκεια της κάθε διαδρομής. Αυτή αποθηκεύεται σε ένα αρχικά μηδενικό διάνυσμα, το ''Routetime''.

Κατόπιν, εφαρμόζεται η διαδικασία του Πλησιέστερου Γείτονα σε έναν κλειστό βρόχο επαναλήψεων μέχρι να εξυπηρετηθούν ολοι οι κόμβοι και τα οχήματα να έχουν επιστρέψει στην αποθήκη. Αν η ‘’thesi’’ έχει την τιμή ‘’1’’, τότε εξετάζεται τον πλησιέστερο κόμβο από την αποθήκη ενώ σε διαφορετική περίπτωση εξετάζεται η εύρεση του πλησιέστερου κόμβου από τον τελευταίο κόμβο που επισκέφτηκε το όχημα. Όταν παραβιαστεί τουλάχιστον ένας από τους περιορισμούς της χωρητικότητας ή του μέγιστου χρονικού ορίου, το όχημα αναγκάζεται να επιστρέψει στην αποθήκη, να τερματίσει τη διαδρομή και να ξεκινήσει μια νέα, επαναλαμβάνοντας την ίδια διεργασία έως ότου να έχουν ελεγχθεί όλοι οι κόμβοι. Έτσι έχοντας καλύψει τα παραπάνω προαπαιτούμενα υπολογίζουμε τα στοιχεία της αρχικής εφικτής λύσης:

- Totalcost: το συνολικό κόστος που υπολογίζεται αθροίζοντας τα επιμέρους κόστη, δηλαδή το routecost κάθε διαδρομής (διάνυσμα Routecost).

- ORoutetime: ο ολικός χρόνος που είναι το άθροισμα των χρόνων, δηλαδή των routetime (διάνυσμα Routetime).

- old_seira: σε αυτή εκχωρείται ο πίνακας της σειράς εξυπηρέτησης των πελατών ώστε να κρατηθεί η αρχική σειρά των διαδρομών.

5.4 Βελτίωση Αρχικής Λύσης – Αλγόριθμοι Τοπικής Αναζήτησης

Στο δεύτερο στάδιο του κώδικα, χρησιμοποιούνται τρεις μέθοδοι Τοπικής Αναζήτησης, οι οποίες παρουσιάστηκαν στο 3ο Κεφάλαιο και στόχο έχουν να βελτιώσουν την αρχική λύση. Αυτές είναι η 1-0 επανατοποθέτηση (relocate), η 1-1 ανταλλαγή (exchange) και η εσωτερική ανταλλαγή (swap).

Για την υλοποίηση των μεθόδων ,αρχικοποιήθηκαν οι εξής μεταβλητές:

- Q: η νέα ζήτηση.
  NRoutecost: νέο κόστος διαδρομής, δηλαδή νέο διάνυσμα Routecost.
  NRoutetime: νέος χρόνος διαδρομής, δηλαδή νέο διάνυσμα Routetime.

- roll: μηδενικό αρχικά διάνυσμα 2 στοιχείων που θα αποθηκεύονται οι τυχαίες επιλογές διαδρομών σε κάθε επανάληψη.
  x και y: οι μεταβλητές που χρησιμοποιούνται ως δείκτες για τη θέση των στοιχείων που πρόκειται να υποβληθούν σε κάποια αλλαγή ή μεταξύ τους ανταλλαγη.

Εν συνεχεία, υλοποιούνται οι μέθοδοι τοπικής αναζήτησης:

1-0 επανατοποθέτηση (1-0 relocate)

Όπως προαναφέρθηκε, η μέθοδος αυτή εφαρμόζεται με τη διαγραφή ενός κόμβου από μια διαδρομή και την επανατοποθέτηση του σε μία άλλη μειώνοντας έτσι το κόστος. Στο συγκεκριμένο κώδικα επιλέγονται τυχαία δύο διαδρομές από τον πίνακα, που δείχνει τη σειρά εξυπηρέτησης των πελατών της αρχικής λύσης. Ελέγχεται αν είναι εφικτή η εφαρμογή της μεθόδου σε αυτές τις διαδρομές. Αν είναι, αυτές τοποθετούνται σε δύο διανύσματα ‘’Avector’’ και ‘’Bvector’’ ώστε να γίνει εκεί η διαδικασία της επανατοποθέτησης των κόμβων χωρίς να επηρεαστεί ο αρχικός πίνακας. Επιλέγεται τυχαία η θέση ‘’x’’ του βοηθητικού διανύσματος ‘’Avector’’ στην οποία θα τοποθετηθεί ο διαγραμμένος κόμβος από το διάνυσμα ‘’Bvector’’, ο οποίος βρίσκεται επίσης σε μια τυχαία θέση ‘’y’’. Κανένας από τους κόμβους δε θα πρέπει να είναι η αποθήκη. Στη συνέχεια γίνεται υπολογισμός της νέας ζήτησης και νέου χρόνου για κάθε νέα διαδρομή οι οποίες θα πρέπει να συμβαδίζουν με τους αντίστοιχους περιορισμούς του CVRP. Εφόσον το νέο κόστος είναι μικρότερο από το προηγούμενο, τότε τοποθετούμε τα διανύσματα στον πίνακα της σειράς εξυπηρέτησης πελατών. Η διαδικασία επαναλαμβάνεται μέχρι το πέρας των επαναλήψεων που έχουμε ορίσει. Έτσι δημιουργείται ένα νέο σύνολο διαδρομών στον πίνακα ‘’seira’’, δηλαδή τον πίνακα της βελτιωμένης λύσης.

Παράδειγμα: *1-0 επανατοποθέτηση (1-0 relocate)*

roll= [3,2], δηλαδή τυχαία επιλέγεται η διαδρομή ‘’3’’ ως Avector και η διαδρομή ‘’2 ‘’ως Bvector.

x=5, επιλέγεται τυχαία η 5η θέση (στήλη) του Avector.

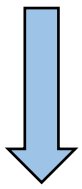
y=5, επιλέγεται τυχαία το στοιχείο που βρίσκεται στη 5η θέση (στήλη) του Bvector.

Avector:

1	2	3	4	5	6	7	8
1	7	49	9	27	1	0	0

Bvector:

1	2	3	4	5	6	7	8
1	28	2	33	12	39	6	1



Avector:

1	2	3	4	5	6	7	8
1	7	49	9	27	12	1	0

Bvector:

1	2	3	4	5	6	7	8
1	28	2	33	39	6	1	0

1-1 ανταλλαγή (1-1 exchange)

Κατόπιν ο κώδικας εφαρμόζει επαναληπτικά μια άλλη μέθοδο τοπικής αναζήτησης για να βελτιώσει ακόμα περισσότερο τη λύση, πρόκειται για την 1-1 exchange. Ο αλγόριθμος της κάνει ανταλλαγή μεταξύ 2 κόμβων που βρίσκονται σε ξεχωριστές διαδρομές ώστε να ελαχιστοποιήσει το κόστος. Η διαδικασία της είναι παρόμοια με της 1-0 επανατοποθέτησης όμως στη συγκεκριμένη περίπτωση επιλέγονται τυχαία δύο θέσεις κόμβων ‘’Acell’’ και ‘’Bcell’’ που αντιστοιχούν στις τυχαίες διαδρομές που έχουν εισαχθεί στα βοηθητικά διανύσματα ‘’Avector’’ και ‘’Bvector’’. Ακολούθως αν πληρούνται οι περιορισμοί γινεται ανταλλαγή του κόμβου της θέσης ‘’Acell’’ με τον κόμβο της ‘’Bcell’’. Υπολογίζεται η καινούργια ζήτηση και ο νέος χρόνος και όταν το κόστος μειώνεται σε σχέση με το προηγούμενο υπάρχον, τότε γίνεται η αντικατάσταση των αντίστοιχων διαδρομών του πίνακα της σειράς των εξυπηρετούμενων πελατών με τα διανύσματα ‘’Avector’’ και ‘’Bvector’’.

Παράδειγμα: *1-1 ανταλλαγή (1-1 exchange)*

roll= [1,2], δηλαδή τυχαία διαλέγεται η διαδρομή ‘’1’’ ως Avector και η διαδρομή ‘’2 ‘’ ως Bvector.

Acell=2, επιλέγεται τυχαία το στοιχείο που βρίσκεται στη $2^{η}$ θέση (στήλη) του Avector.

Bcell=2, επιλέγεται τυχαία το στοιχείο που βρίσκεται στη $2^{η}$ θέση (στήλη) του Bvector.

temp1=47, το στοιχείο που βρίσκεται στη $2^{η}$ στήλη του Avector.

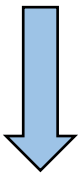
temp2=28, το στοιχείο που βρίσκεται στη $2^{η}$ στήλη του Bvector.

Avector:

1	2	3	4	5	6	7	8
1	47	13	48	5	18	38	1

Bvector:

1	2	3	4	5	6	7	8
1	28	2	33	12	39	6	1



Avector:

1	2	3	4	5	6	7	8
1	28	13	48	5	18	38	1

Bvector:

1	2	3	4	5	6	7	8
1	47	2	33	12	39	6	1

ανταλλαγή (swap)

Τέλος για περαιτέρω βελτίωση της λύσης χρησιμοποιείται επαναληπτικά η μέθοδος swap. Η λογική της είναι αντίστοιχη των παραπάνω μεθόδων, μόνο που η συγκεκριμένη εφαρμόζεται σε μία διαδρομή. ’Ετσι επιλέγεται τυχαία μια διαδρομή της σειράς εξυπηρετούμενων πελατών και εισάγεται σε ένα βοηθητικό διάνυσμα ‘’Vector’’. Έπειτα διαλέγονται επίσης τυχαία δύο θέσεις ‘’ACell’’ και ‘’BCell’’του

συγκεκριμένου διανύσματος, οι οποίες δεν πρέπει να αντιστοιχουν στην αποθήκη. Αν η ζήτηση και το μέγιστο χρονικό όριο των στοιχείων που βρίσκονται σε αυτές τις θέσεις πληρούν τους περιορισμούς, τότε γίνεται ανταλλαγή μεταξύ τους. ’Ετσι προκύπτει μια νέα διαδρομή, η οποία αν έχει μικρότερο κόστος σε σχέση με τη προηγούμενη θεωρείται αποδεκτή.

Παράδειγμα: *εσωτερική ανταλλαγή (swap)*

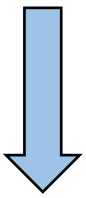
roll= [2], δηλαδή τυχαία διαλέγεται η διαδρομή ‘’2’’ ως Vector.

ACell=3, επιλέγεται τυχαία το στοιχείο που βρίσκεται στη 3^{η} θέση (στήλη) του Avector.

BCell=5, επιλέγεται τυχαία το στοιχείο που βρίσκεται στη 5^{η} θέση (στήλη) του Bvector.

Vector:

1	2	3	4	5	6	7	8
1	28	2	33	12	39	6	1



Vector:

1	2	3	4	5	6	7	8
1	28	12	33	2	39	6	1

5.5 Εύρεση Τελικής Λύσης – Αλγόριθμος Προσομοιωμένης Ανόπτησης (SA)

Στο τρίτο και τελευταίο στάδιο του κώδικα, εφαρμόζεται ο αλγόριθμος της προσομοιωμένης ανόπτησης (simulated annealing), η εφαρμογή του οποίου, θα συμβάλλει στη τελική διαμόρφωση και βελτίωση της λύσης. Η μέθοδος αυτή χρησιμοποιείται επαναληπτικά και περιλαμβάνεται στον ολικό βρόγχο που εμπεριέχει τις τοπικές αναζητήσεις του προηγούμενου σταδίου.

Βήμα 1

Σε πρώτη φάση επιλέγεται μια αρχική λύση ‘’S_0’’ η οποία έχει υπολογιστεί στο πρώτο στάδιο του κώδικα, δηλαδή μέσω του αλγορίθμου του Πλησιέστερου Γείτονα. Πρόκειται για το ολικό κόστος ‘’Totalcost’’ των διαδρομών που βρίσκονται στον πίνακα της σειράς εξυπηρέτησης πελατών με όνομα ‘’old_seira’’.

Βήμα 2

Εκτός του βρόγχου ορίζονται η αρχική θερμοκρασία ‘’T_0’’ καθώς και ένας μετρητής με την ονομασία ‘’fores’’ ο οποίος δείχνει τον αριθμό των φορών που η λύση γίνεται αποδεκτή μετά την εφαρμογή της προσομοιωμένης ανόπτησης. Συχνά όταν ξεκινάει ο αλγόριθμος έχει μεγάλη τιμή θερμοκρασίας και μειώνεται με την πάροδο των επαναλήψεων. Επίσης αρχικοποιείται η μεταβλητή ‘’OVERALLcost ‘’ η οποία αναφέρεται στην τελική λύση του αλγορίθμου.Αρχικά παίρνει την τιμή του ολικού κόστους της μεθόδου του Πλησιέστερου Γείτονα , δηλαδή του ‘’Totalcost’’.

Βήμα 3

Μετά το πέρας των τοπικών αναζητήσεων έχει υπολογιστεί μια νέα βελτιωμένη γειτονική λύση ‘’S’’. Στη συγκεκριμένη περίπτωση αυτή είναι το νέο συνολικό κόστος διαδρομών ‘’NORroutecost’’ του πίνακα εξυπηρετησης πελατών ‘’seira’’.Αρχικά γίνεται σύγκριση μεταξύ του ‘’NORroutecost’’ και του ‘’OVERALLcost’’, έτσι ώστε σε περίπτωση που ο αλγόριθμος χειροτερέψει τη λύση, να διατηρηθεί η βέλτιστη. Με αυτόν τον τρόπο αποθηκεύεται ως ‘’OVERALLcost’’ η καλύτερη λύση.

Βήμα 4

Υπολογίζεται το αποτέλεσμα της συγκεκριμένης αλλαγής της λύσης, δηλαδή της διαταραχής ‘’delta’’, όπου delta=S-S_0 .Αν γίνει αποδεκτή, δηλαδή delta < 0 , τότε έχει επιτευχθεί ελαχιστοποίηση του κόστους μέσω των τοπικών αναζητήσεων και το ‘’S_0‘’παίρνει την τιμή του ‘’S’’. Αν όμως delta ≥ 0 τότε σημαίνει πως η λύση ίσως έχει χειροτερέψει. Στη διαδικασία της τοπικής αναζήτησης αυτή η λύση θα απορριπτόταν όμως σύμφωνα με τον αλγόριθμο της προσομοιωμένης ανόπτησης η νέα κατάσταση γίνεται αποδεκτή με κάποια πιθανότητα ‘’P’’. Για τον υπολογισμό της συγκεκριμένης πιθανότητας χρειάζεται το delta και η θερμοκρασία ‘’T’’ κατά την επανάληψη u που βρίσκεται το τρέχον πρόγραμμα. Η θερμοκρασία ‘’T’’ υπολογίζεται από τον τύπο $T=T_0-u\times A$. Το ‘’A’’ είναι τυχαίος αριθμός με τιμές μεγαλύτερες του 0,8 και μικρότερες του 1. ’Ετσι η πιθανότητα δίνεται ως $P=e^{(-delta/T)}$, η οποία για να γίνει αποδεκτή καθορίζεται από μια τιμή ‘’X’’, η οποία προκύπτει μέσω μιας γεννήτριας τυχαίων αριθμών μεταξύ του 0 και του 1. Αν η τιμή της είναι μικρότερη από την τιμή της πιθανότητας τότε η πιθανότητα ‘’P’’ είναι αποδεκτή. Σε αυτή τη περίπτωση εκχωρείται η τιμή του ‘’S’’ στο ‘’S_0’’και αυξάνεται κατά μία μονάδα η τιμή του μετρητή ‘’fores’’. Η διαδικασία συνεχίζεται μέχρι το τέλος των επαναλήψεων ‘’u’’ που έχουν οριστεί.

Παράδειγμα: *προσομοιωμένη ανόπτηση (simulated annealing)*

‘Εστω σενάριο :

n=51 (κόμβοι)

Qmax=160 (μέγιστη χωρητικότητα)

Tmax=200 (μέγιστος χρόνος διαδρομής)

ST=10 (χρόνος εξυπηρέτησης)

Από τον χρήστη έχουν οριστεί:

u=500 (συνολικός αριθμός επαναλήψεων)

T_0=500 (αρχική τιμή θερμοκρασίας)

‘Εστω ότι το κόστος που έχει βρεθεί ως αρχική λύση μέσω της διαδικασίας του αλγορίθμου του Πλησιέστερου Γείτονα είναι ‘’Totalcost=1271’’.

Για u=1, μετά το πέρας των τοπικών αναζητήσεων το κόστος έχει διαμορφωθεί ως ‘’NORroutecost=1171’’.

Σε πρώτο στάδιο ο αλγόριθμος που κατασκευάστηκε ελέγχει αν η βελτιώση που έχει προκύψει αποτελεί τη βέλτιστη λύση ‘’OVERALLcost’’.

Όσον αναφορά τη 1η επανάληψη ,ως ‘’OVERALLcost’’ έχει δωθεί αρχικά η τιμή της λύσης του Πλησιέστερου Γείτονα ,δηλαδή ‘’OVERALLcost=Totalcost=1271’’.

‘Ετσι ξεκινώντας γίνεται ένας έλεγχος αν το ‘’OVERALLcost>NORroutecost’’ ,εάν αυτό είναι αποδεκτό τότε εκχωρείται η τιμή που προέκυψε μέσω των τοπικών αναζητήσεων ως η πλέον καλύτερη λύση που έχει βρεθεί. Έτσι σε αυτή τη περίπτωση ‘’OVERALLcost=NORroutecost’’ ,δηλαδή ‘’OVERALLcost=1281’’.

Μετά από τον παραπάνω έλεγχο ξεκινάει η διαδικασία εφαρμογής του αλγορίθμου της προσομοιωμένης ανόπτησης.

Καθώς ο αλγόριθμος βρίσκεται στη 1η επανάληψη ως ‘’S_0’’ αρχικά εχει οριστεί η αρχική λύση ‘’Totalcost’’ ,οπότε ‘’S_0=1271’’ .Σε πρώτη φάση εκχωρείται ως ‘’S’’ η νέα λύση, δηλαδή η τιμή του ‘’NORroutecost’’ ,οπότε ‘’S=1171’’ .

Στη συνέχεια υπολογίζεται η τιμή της διαταραχής ‘’delta’’ που προκύπτει από τον τύπο ‘’delta=S-S_0’’ . ‘Ετσι για το συγκεκριμένο παράδειγμα ‘’delta=1171-1271= -100’’ , άρα ‘’delta <0 ‘’, οπότε η νέα λύση είναι 100% αποδεκτή , έτσι εκχωρείται η τιμή του ‘’S’’ στο ‘’S_0’’. Συνεπώς τώρα ‘’S_0=1171’’.

‘Εστω ότι στην επόμενη επανάληψη u=2 , μετά το πέρας των τοπικών αναζητήσεων το NORroutecost που έχει προκύψει είναι 1181.

Γινεται έλεγχος αν ‘’ OVERALLcost>NORroutecost ‘’ ,δηλαδή αν η νέα λύση είναι καλύτερη από την υπάρχουσα βέλτιστη . Φαίνεται πως κάτι τέτοιο δεν ισχύει σε αυτή τη περίπτωση, καθώς 1171<1181 .

‘Ετσι το κόστος χειροτερεύει , όμως ο αλγόριθμος της προσομοιωμένης ανόπτησης ενδέχεται να κάνει αποδεκτή τη λύση με κάποια πιθανότητα αποδοχής .

Συνεχίζοντας στη διαδικασία της προσομοιωμένης ανόπτησης , εκχωρείται στο ‘’S’’ η νέα τιμή του ‘’NORroutecost’’ ,οπότε ‘’S=1181’’ .

Παρατηρείται ότι ‘’delta>0’’ καθώς ‘’delta=1181-1171=10 ‘’, άρα υπολογίζεται η πιθανότητα αποδοχής της συγκεκριμένης διαταραχής από τον τύπο ‘’p=e^(-delta/Ti)’’

Το ‘’Ti’’ είναι η θερμοκρασία εκείνη τη στιγμή και προκύπτει ως ‘’ Ti=T0-u*A’’ ,όπου υποθέτωντας ότι η τυχαία τιμή του A είναι ‘’0,81’’ το οποίο παίρνει τυχαίες τιμές από 0,80 έως και 0,99 , τότε ‘’Ti=500-2*0,81=498,38’’ .

‘Αρα η πιθανότητα στο συγκεκριμένο παράδειγμα είναι : ‘’p=e^(-10/498,38)=0,98’’

Κατόπιν με μία γεννήτρια τυχαίων αριθμών προκύπτει μια τιμή ‘’X’’ μεταξύ του 0 και του 1. ‘Εστω ότι η τιμή που προέκυψε τυχαία είναι η 0,90 . Αφού η τιμή είναι μικρότερη

από την τιμή της πιθανότητας (X<p) ,δηλαδή 0,90<0,98 , η λύση γίνεται αποδεκτή με πιθανότητα ‘’p=0,98’’.

‘Επειτα εκχωρείται η τιμή του ‘’S’’ στο ‘’S_0’’. Συνεπώς τώρα ‘’S_0=1181’’.

Ταυτόχρονα αυξάνεται η τιμή του μετρητή ‘’fores’’ ,που απεικονίζει πόσες φορές γίνεται αποδεκτή μια τέτοια λύση, κατά μία μονάδα , δηλαδή ‘’fores=1’’ .

Η διαδικασία συνεχίζεται μέχρι να ολοκληρωθούν όλες οι επαναλήψεις ‘’u’’ που έχουν οριστεί από τον χρήστη εξαρχής.

Αν η διαδικασία σταματούσε στη $2^{η}$ επανάληψη , δε θα είχαμε ως καλύτερη λύση την τελευταία με τιμή 1181, αλλά τη λύση της $1^{ης}$ επανάληψης με τιμή 1171 την οποία την έχει κρατήσει το πρόγραμμα ως ‘’OVERALLcost’’ κατά τον έλεγχο βέλτιστης λύσης που γίνεται. Εκτός φυσικά από το κόστος διατηρείται και οι αντιστοιχες καλύτερες τιμές για τον ολικό χρόνο (OVERALLtime), τη συνολική ζήτηση (OVQ) και τη τελική σειρά εξυπηρέτησης πελατών (OVseira). Έτσι δεν υπάρχει πιθανότητα να χαθεί η καλύτερη λύση.

Παρακάτω παρουσιάζονται τα αποσπάσματα του κώδικα που έχει δημιουργηθεί για τη παρούσα διπλωματική εργασία στο περιβάλλον της MATLAB:

‘Ελεγχος βέλτιστης λύσης

```
if (OVERALLcost>NORroutecost)
   OVERALLcost=NORroutecost;
   OVERALLtime=NORroutetime;
   OVQ=Q;
   OVseira=seira;
   OVERALLroutecost=NRoutecost;
   OVERALLroutetime=NRoutetime;
end
```

Διαδικασία Προσομοιωμένης Ανόπτησης

```
S=NORroutecost;
delta=S-S0;
        if delta<0
         S0=S;
        else
         A=Amin+rand (1,1) *(Amax-Amin);
         Ti=T0-u*A;
         p=e^(-delta/Ti);
         X=Xmin+rand (1,1) *(Xmax-Xmin);
                if X<p
                  S0=S;
                  fores=fores+1;
                end
       end
```

Κεφάλαιο 6 – Αποτελέσματα και Συμπεράσματα

6.1 Παρουσίαση Αποτελεσμάτων

Στον Πίνακα 6.1 αποτυπώνονται τα αποτελέσματα που προέκυψαν μετά από ένα μεγάλο αριθμό εκτέλεσης του προγράμματος. Συγκρίνοντας τις καλύτερες λύσεις που βρέθηκαν μέσω της διαδικασίας με τις μέχρι στιγμής παγκοσμίως βέλτιστες, οι οποίες είναι γνωστές σύμφωνα με τη βιβλιογραφία [11], παρατηρείται ότι κατά μέσο όρο υπάρχει απόκλιση 16%. Με βάση τα δεδομένα των 14 παραδειγμάτων (σεναρίων) CMT που μελετήθηκαν είναι εύλογο να χωριστούν σε δύο κατηγορίες. Η 1^{η} κατηγορία περιλαμβάνει τα 7 σενάρια (CMT 1 ,CMT 2 , CMT 3, CMT 4, CMT 5, CMT 11, CMT 12) τα οποία δεν εξαρτώνται από το χρονικό περιορισμό του μέγιστου ορίου διαδρομής (Tmax ≈ ∞) καθώς και ο χρόνος εξυπηρέτησης του κάθε πελάτη είναι μηδενικός. Στη 2^{η} κατηγορία κατατάσσονται τα υπόλοιπα 7 παραδείγματα (CMT 6 ,CMT 7 , CMT 8, CMT 9, CMT 10, CMT 13, CMT 14) στα οποία υπάρχει τόσο μέγιστο επιτρεπτό χρονικό όριο όσο και χρόνος εξυπηρέτησης πελατών.

Όσον αναφορά τη διαδικασία εκτέλεσης του προγράμματος για κάθε παράδειγμα:

Σε πρώτη φάση δημιουργείται μια αρχική λύση μέσω του αλγορίθμου του Πλησιέστερου Γείτονα, αποδεκτή σύμφωνα με τους περιορισμούς του προβλήματος. Όπως φαίνεται από τα αποτελέσματα στον παρακάτω πίνακα (Πίνακας 6.1), η αρχική λύση βρίσκεται αρκετά μακριά από τη τελική, καθώς για όλα τα παραδείγματα η μέση απόκλιση από τις βέλτιστες λύσεις υπολογίστηκε στο 37%. Αυτό δημιουργεί εξαρχής ένα μεγάλο εμπόδιο στη βελτίωση της λύσης καθώς επιβραδύνει σημαντικά τη σύγκλιση του αλγορίθμου. Σε δεύτερη φάση γίνεται η διαδικασία βελτίωσης της αρχικής λύσης μέσω των μεθόδων τοπικής αναζήτησης. Η μέθοδος που επιλέχθηκε ως πρώτη σε σειρά ήταν η 1-0 επανατοποθέτηση (1-0 relocate), η οποία τυχαιοποιημένα εξέταζε αν ένας πελάτης μιας διαδρομής μπορεί να εισαχθεί σε κάποια άλλη με στόχο την ελαχιστοποίηση του συνολικού κόστους αλλά και την εξάλειψη κάποιων πολύ μικρών διαδρομών. Δεύτερη σε σειρά επιλέχθηκε η 1-1 ανταλλαγή (1-1 exchange) η οποία τυχαιοποιημένα αντάλλαζε πελάτες απο διαφορετικές διαδρομές χωρίς να επηρεάζει το μέγεθος τους αλλά μόνο τη μείωση του ολικού κόστους. Τελευταία μέθοδος που διαλέχθηκε ήταν η εσωτερική ανταλλαγή (swap), η οποία επίσης τυχαία

επελέγε δύο πελάτες εντός της ίδιας διαδρομης και τους αντάλλαζε μεταξύ τους στοχεύοντας στη μείωση του κόστους. Με τη συγκεκριμένη σειρά εκτέλεσης των παραπάνω μεθόδων επιτυγχάνεται πρώτα η διαμόρφωση του τελικού πλήθους διαδρομών δίνοντας στον αλγόριθμο της 1-0 επανατοποθέτησης μεγαλύτερη ευελιξία κινήσεων προκειμένου να εξαλείψει μικρές διαδρομές με μεγάλο κόστος, μείωνοντας έτσι σε έναν βαθμό το ολικό κόστος. Εν συνεχεία γίνεται πιο στοχευμένη ανταλλαγή πελατών σε διαφορετικές διαδρομές με αποτέλεσμα να μειωθεί ακόμα περισσότερο το ολικό κόστος μέσω της 1-1 ανταλλαγής. Ενώ η χρήση της εσωτερικής ανταλλαγής προσδίδει μια περαιτέρω μείωση του ολικού κόστους μέσα στις ήδη διαμορφωμένες τελικές διαδρομές. Τέλος σε τρίτη φάση εφαρμόστηκε ο μεθευρετικός αλγόριθμος της προσομοιωμένης ανόπτησης με στόχο τη διαμόρφωση της καλύτερης τελικής λύσης δημιουργώντας συνθήκες για την αποφυγή εγκλωβισμού μιας λύσης σε τυχών τοπικά βέλτιστα.

Αναλύωντας τα στοιχεία του Πίνακα 6.1, στη πρώτη στήλη του παρουσιάζονται τα ονόματα των παραδειγμάτων (σεναρίων) των Christofides, Mingozzi και Toth (1979) [CMT]. Στη δεύτερη στήλη βρίσκεται ο αριθμός των κόμβων,δηλαδή όσοι είναι οι πελάτες προς εξυπηρέτηση και ένας ο κόμβος που αντιπροσωπεύει την αποθήκη (π.χ 51=50 πελάτες + 1 η αποθήκη). Στη τρίτη και την τέταρτη στήλη αποτυπώνονται τα όρια των περιορισμών της χωρητικότητας και του μέγιστου επιτρεπτού χρόνου αντίστοιχα. Στην πέμπτη στήλη εμφανίζεται ο χρόνος εξυπηρέτησης σε κάθε σενάριο. Ακολούθως στην επόμενη στήλη βρίσκεται η τιμή της αρχικής λύσης που βρέθηκε σε κάθε παράδειγμα μετά την ολοκλήρωση του αλγορίθμου του Πλησιέστερου Γείτονα (Totalcost). Στην έβδομη στήλη εμφανίζεται η καλύτερη τελική λύση για το καθένα μετά από την επαναληπτική εκτέλεση του προγράμματος (OVERALLcost). Στην όγδοη με βάση τη βιβλιογραφία [11] αναγράφονται οι μέχρι στιγμής παγκοσμίως βέλτιστες λύσεις των παραδειγμάτων που λήφθηκαν προς επεξεργασία, η επόμενη στήλη περιλαμβάνει το ποσοστό που δείχνει τη βελτίωση της λύσης από την αρχική στη τελική , με αυτόν το τρόπο απεικονείζεται το πόσο καθοριστικός ηταν ο ρόλος των τοπικών αναζητήσεων και της προσομοιωμένης ανόπτησης στη βελτίωση της λύσης για τον αλγόριθμο. Η δέκατη αποτυπώνει σε μορφή ποσοστού την απόκλιση μεταξύ της τελικής καλύτερης λύσης που βρέθηκε μέσω της διαδικασίας εκτέλεσης του προγράμματος και της ήδη γνωστής βέλτιστης λύσης του κάθε σεναριού. Αυτή φανερώνει το πόσο καλά λειτούργησε ο αλγόριθμος για τα δεδομένα του εκάστοτε

προβλήματος. Στη προτελευταία στήλη του πίνακα φαίνεται το τελικό πλήθος των διαδρομών που χρειάστηκαν για την παραγωγή της καλύτερης λύσης που βρέθηκε. Στη τελευταία στήλη βρίσκεται ο βέλτιστος μέχρι στιγμής αριθμός διαδρομών για κάθε παράδειγμα.

Name	Nodes	Qmax	Tmax	ST	Initial Solution	Final Solution	Best Known Solution	Deviation Initial-Final (%)	Deviation Final-Best (%)	Final Routes	Best Known Routes
CMT1	51	160	999999	0	710,34	573,00	524,61	24%	9%	6	5
CMT2	76	140	999999	0	962,31	899,57	835,26	7%	8%	11	10
CMT3	101	200	999999	0	1053,65	919,51	826,14	15%	11%	8	8
CMT4	151	200	999999	0	1309,41	1155,71	1028,42	13%	12%	12	12
CMT5	200	200	999999	0	1580,70	1457,84	1291,29	8%	13%	17	16
CMT6	51	160	200	10	1054,43	659,41	555,43	60%	19%	7	6
CMT7	76	140	160	10	1839,79	1063,71	909,68	73%	17%	13	11
CMT8	101	200	230	10	1764,13	1064,62	865,95	66%	23%	12	9
CMT9	151	200	200	10	2621,58	1550,52	1162,55	69%	33%	22	14
CMT10	200	200	200	10	3265,59	1910,44	1395,85	71%	37%	27	18
CMT11	121	200	999999	0	1213,88	1132,44	1042,12	7%	9%	8	7
CMT12	101	200	999999	0	1087,86	831,06	819,56	31%	1%	10	10
CMT13	121	200	720	50	2937,61	1770,36	1541,14	66%	15%	12	11
CMT14	101	200	1040	90	1146,54	1004,92	866,37	14%	16%	13	11

Πίνακας 6.1 :Τελικά Αποτελέσματα Προβλημάτων CMT

Η πρώτη φάση, δηλαδή ο αλγόριθμος του Πλησιέστερου Γείτονα ολοκληρώνεται όταν έχουν εξυπηρετηθεί όλοι οι πελάτες του κάθε παραδείγματος. Αντίθετα το κριτήριο τερματισμού για τις άλλες δύο φάσεις του προγράμματος, δηλαδή τις τοπικές αναζητήσεις και της προσομοιωμένης ανόπτησης είναι η ολοκλήρωση ενός συγκεκριμένου αριθμού επαναλήψεων. Χρησιμοποιείται ένας ξεχωριστός βρόχος επαναλήψεων για κάθε μία από αυτές, αλλά και ένας ολικός που εμπεριέχει αυτές καθώς και την μέθοδο της προσομοιωμένης ανόπτησης. Ο μετρητής που αυξάνεται όταν έχουν ολοκληρωθεί από μία φορά όλες οι μέθοδοι είναι ο ‘’u’’, ενώ οι μετρητές που δείχνουν το πλήθος των επαναλήψεων που έχουν ολοκληρωθεί σε κάθε μέθοδο

ξεχωριστά είναι οι ‘’t’’ (για τη 1-0 relocate), ‘’e’’ (για 1-1 τη exchange) και ‘’h’’ (για τη swap). Στο συγκεκριμένο πρόβλημα δίνεται η δυνατότητα στον χρήστη να καθορίσει εξαρχής αυτόν τον αριθμό δίνοντας τις κατάλληλες τιμές στις μεταβλητές ,δηλαδή στους μετρητές των επαναλήψεων. Η παραμετροποίηση μιας μεταβλητής που είναι καθοριστική για την εξέλιξη της εκτέλεσης της προσομοιωμένης ανόπτησης είναι η αρχική θερμοκρασία ‘’T_0’’ . Αυτή ορίζεται επίσης από τον χρήστη πριν την εφαρμογή του αλγορίθμου . Έτσι έπειτα από ένα μεγαλο πλήθος δοκιμών κατά την εκτέλεση του προγράμματος, με βάση τα δεδομένα και τους περιορισμούς του κάθε παραδείγματος CMT ορίστηκαν οι κατάλληλες τιμές των μεταβλητών. Αναλυτικά αυτές εμφανίζονται στον παρακάτω πίνακα (Πίνακας 6.2). Παράλληλα στο συγκεκριμένο πίνακα παρουσιάζονται οι αριθμοί των κινήσεων της κάθε μεθόδου αναζήτησης μέσω των αντίστοιχων μετρητών:

- moves: μετρητής των κινήσεων της μεθόδου 1-0 επανατοποθέτησης, δηλαδή συγκρατεί το πλήθος των επιτυχημένων αλλαγών που γίνονται.
- antallages: μετρητής των κινήσεων της μεθόδου 1-1 ανταλλαγής, δηλαδή διατηρεί το πλήθος των επιτυχημένων ανταλλαγών.
- changes: μετρητής για τη μεθόδο εσωτερικής ανταλλαγής, δηλαδή κρατάει το πλήθος εσωτερικών ανταλλαγών που επιτυγχάνονται.
- fores: μετρητής για τη μέθοδο της προσομοιωμένης ανόπτησης, δηλαδή απεικονίζει τον αριθμό των φορών που η λύση είναι αποδεκτή.

	CMT1	CMT2	CMT3	CMT4	CMT5	CMT6	CMT7	CMT8	CMT9	CMT10	CMT11	CMT12	CMT13	CMT14
fores	360	384	365	350	357	428	1884	1918	7656	9765	368	375	624	662
t	500	500	500	800	1000	1000	2000	2000	2000	3000	1000	800	800	800
e	800	800	800	1000	1500	2000	2000	2000	2000	3000	1500	2000	1000	1000
h	1000	1000	1000	1000	1000	2000	1000	1000	1000	1500	2000	1000	1000	1000
u	400	400	400	400	400	500	2000	2000	8000	10000	400	400	700	700
T0	400	400	400	400	400	500	2000	2000	8000	10000	400	400	700	700
Ti	47,03	40,96	38,37	63,03	62,83	23,04	373,22	165,27	375,98	702,87	10,84	17,88	91,33	83,19
moves	31	10	21	29	32	52	88	124	152	252	24	16	153	31
antallages	10	7	10	20	10	28	33	48	85	122	9	9	113	13
changes	9	1	16	17	20	14	19	48	90	106	17	13	108	9

Πίνακας 6.2 Τιμές μεταβλητών για κάθε παράδειγμα CMT.

Για κάθε παράδειγμα CMT ο αλγόριθμος εκτελέστηκε διαφορετικά καθώς τα δεδομένα αλλα και τα αποτελέσματα που λήφθηκαν υπόψιν μετά από ένα μεγάλο πλήθος δοκιμών είχαν ως απόρρια συγκεκριμένη παραμετροποίηση των μεταβλητών όπως αυτή αποτυπώνεται στον παραπάνω πίνακα (Πίνακας 6.2). Για τα προβλήματα της $1^{ης}$ κατηγορίας ο αριθμός των συνολικών επαναλήψεων ‘’u’’ ορίστηκε στις 400 , καθώς παρατηρήθηκε πως αρκούν για τη βελτίωση της αρχικής λύσης. Δοκιμάζοντας περισσότερες επαναλήψεις για τη συγκεκριμένη ομάδα προβλημάτων δε παρατηρήθηκε κάποια περαιτέρω βελτίωση της λύσης αντίθετα ο αλγόριθμος καθυστερούσε μένοντας στάσιμος στα ίδια αποτελέσματα. Για τα παραδείγματα της $2^{ης}$ κατηγορίας επιλέχθηκε μεγαλύτερος αριθμός συνολικών επαναλήψεων κατά τη διαδικασία βελτίωσης της αρχικής λύσης για κάθε ένα από αυτά σύμφωνα με το πλήθος των πελατών και τα όρια των περιορισμών. Ορίστηκαν 500 επαναλήψεις για το σενάριο "CMT 6" που περιλαμβάνει 51 κόμβους, 2000 επαναλήψεις για τα παραδείγματα "CMT 7" και "CMT 8" με 76 και 101 κομβους αντίστοιχα. Στη περίπτωση των παραδειγμάτων "CMT 9" και "CMT 10" χρησιμοποιήθηκαν πάνω από 8000 επαναλήψεις για τη βελτίωση της αρχικής λύσης καθώς τοσο το πλήθος των κόμβων, που ηταν μεγαλύτερο από 150 αλλά και οι περιορισμοί της χωρητικότητας και του χρόνου δυσκόλευαν την εξέλιξη της διαδικασίας του αλγορίθμου με αποτέλεσμα να δημιουργούνται πολλές μικρές διαδρομές με μεγάλο κόστος ,οι οποίες ηταν δύσκολο να εξαλειφθούν παρα τις πολλες αλλαγές που γίνονται σε ολα τα στάδια βελτίωσης της λύσης. Τα δυο τελευταία σενάρια της συγκεκριμένης κατηγορίας ,δηλαδη "CMT 13" και "CMT 14" χρειάστηκαν 700 επαναλήψεις για να φέρουν τα επιθυμητά αποτελέσματα με πλήθος κομβων κατω των 150 παρά τις μεγάλες τιμές που εμφανίζουν στον περιορισμό του μέγιστου χρονικού ορίου. Οι τιμές που δώθηκαν ως αρχική θερμοκρασία "’T_0" για κάθε παράδειγμα είναι ίσες με τον αντίστοιχο αριθμό των επαναλήψεων "u", διότι η μείωση της θερμοκρασίας γίνεται γραμμικά. Αυτό επιλέχθηκε καθώς δεν ηταν επιθυμητό να μηδενίζεται κάποια στιγμή κατα την εκτέλεση του προγράμματος η τιμή " T_i" που δείχνει την θερμοκρασία στην επανάληψη "u_i". Παρατηρήθηκε πως μετά απο έναν μεγάλο αριθμό επαναλήψεων για όλα τα παραπάνω παραδείγματα δεν υπήρχε βελτίωση της λύσης, ενώ αντιθέτως υπήρχε προσκόλληση του αλγορίθμου σε μια τιμή για πολυ μεγάλο χρονικό διάστημα.

6.2 Γραφική Απεικόνιση Αποτελεσμάτων

Στην συγκεκριμένη ενότητα του κεφαλαίου γίνεται γραφική αναπαράσταση κάποιων εκ των σεναρίων που εκτελέστηκαν. Παράλληλα παρουσιάζεται η μετάβαση από την αρχική στην τελική σειρά εξυπηρέτησης των πελατών δηλαδή οι αλλαγές στις διαδρομές, στο κόστος, τον χρόνο και την ζήτηση μετά το πέρας των τοπικών αναζητήσεων και του αλγορίθμου της προσομοιωμένης ανόπτησης.

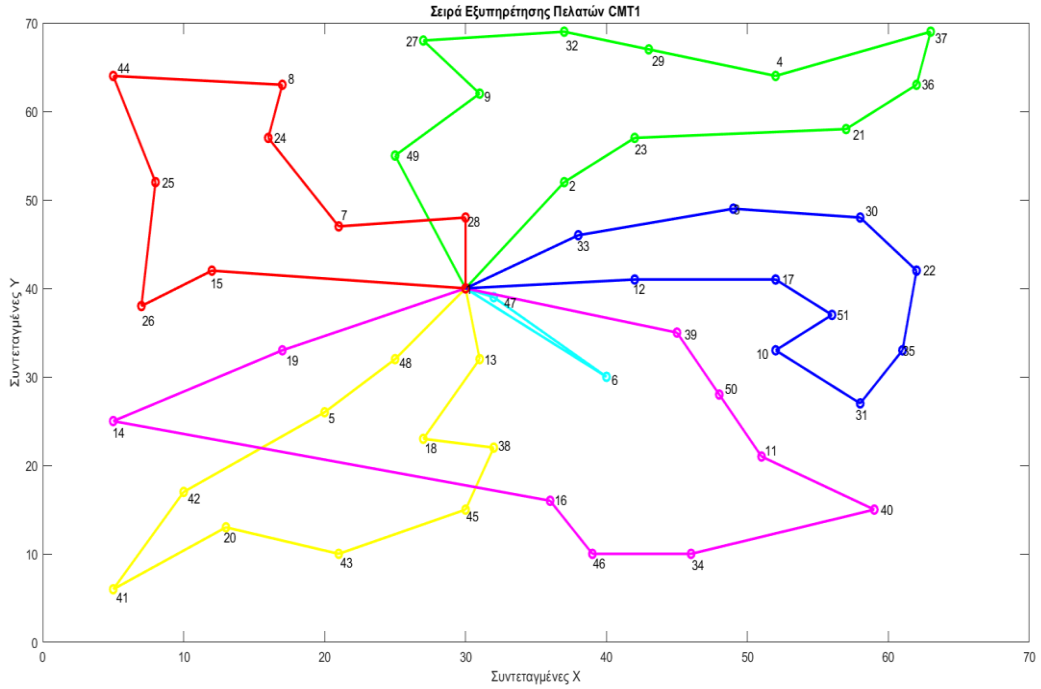
Αποτελέσματα CMT 1

n=51

Qmax=160

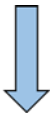
Tmax=999999

servicetime=0



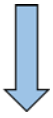
Συνολικό Κόστος : 573,00

old_seira	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	47	13	48	5	18	38	16	45	43	20	42	1
2	1	28	2	33	12	39	6	50	10	51	17	1	0
3	1	7	49	9	27	32	29	4	21	36	37	30	1
4	1	19	15	26	14	41	46	34	11	1	0	0	0
5	1	23	3	22	35	31	40	24	8	44	1	0	0
6	1	25	1	0	0	0	0	0	0	0	0	0	0



seira	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	13	18	38	45	43	20	41	42	5	48	1	0
2	1	6	47	1	0	0	0	0	0	0	0	0	0
3	1	49	9	27	32	29	4	37	36	21	23	2	1
4	1	19	14	16	46	34	40	11	50	39	1	0	0
5	1	33	3	30	22	35	31	10	51	17	12	1	0
6	1	15	26	25	44	8	24	7	28	1	0	0	0

Routecost		Routetime		Demand
103,45		103,45		155
90,31		90,31		143
134,31		134,31		160
145,25		145,25		158
186,90		186,90		151
50,12		50,12		10



NRoutecost		NRoutetime		Q
102,54		102,54		147
28,42		28,42		26
119,34		119,34		154
130,15		130,15		159
95,27		95,27		156
97,28		97,28		135

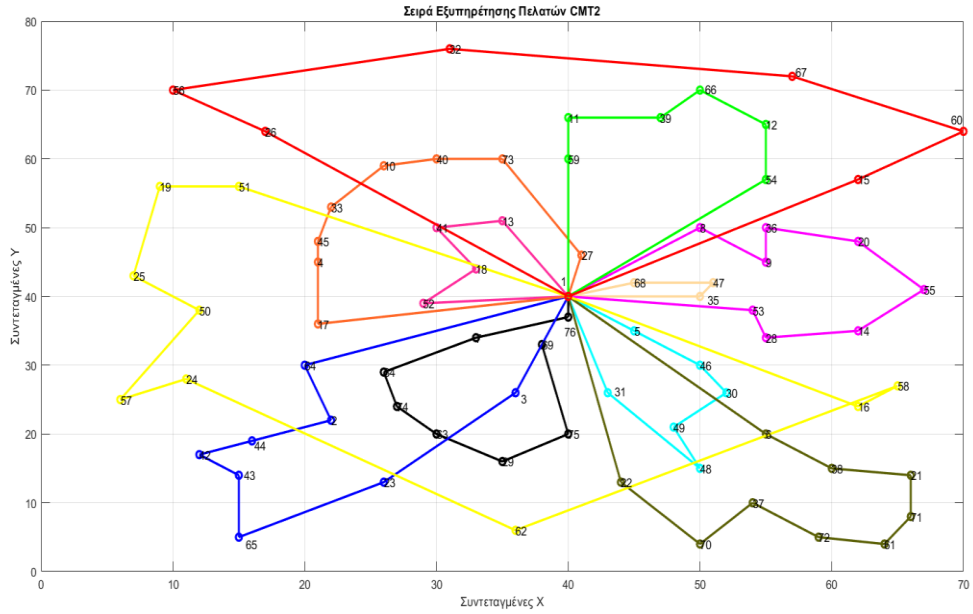
Αποτελέσματα CMT 2

n=76

Qmax=140

Tmax=999999

servicetime=0

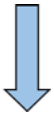


Συνολικό Κόστος : 899,57

old_seira	1	2	3	4	5	6	7	8	9	10	11
1	1	76	69	7	52	18	41	13	1	0	0
2	1	68	35	47	9	36	8	54	1	0	0
3	1	27	5	46	30	49	48	1	0	0	0
4	1	53	28	14	55	20	15	1	0	0	0
5	1	31	75	29	63	74	34	64	1	0	0
6	1	3	2	44	42	43	65	23	1	0	0
7	1	17	4	45	33	10	40	73	1	0	0
8	1	59	11	39	66	12	1	0	0	0	0
9	1	6	38	21	71	61	72	37	70	22	1
10	1	16	58	62	24	57	50	25	19	51	1
11	1	26	56	32	67	60	1	0	0	0	0

seira	1	2	3	4	5	6	7	8	9	10	11
1	1	52	18	41	13	1	0	0	0	0	0
2	1	35	47	68	1	0	0	0	0	0	0
3	1	5	46	30	49	48	31	1	0	0	0
4	1	53	28	14	55	20	36	9	8	1	0
5	1	69	75	29	63	74	34	7	76	1	0
6	1	64	2	44	42	43	65	23	3	1	0
7	1	17	4	45	33	10	40	73	27	1	0
8	1	59	11	39	66	12	54	1	0	0	0
9	1	6	38	21	71	61	72	37	70	22	1
10	1	16	58	62	24	57	50	25	19	51	1
11	1	26	56	32	67	60	15	1	0	0	0

Routecost		Routetime		Demand
49,27		49,27		130
59,28		59,28		139
68,98		68,98		121
78,55		78,55		110
72,37		72,37		123
97,56		97,56		128
73,47		73,47		121
74,23		74,23		117
106,12		106,12		132
176,61		176,61		136
105,87		105,87		107



NRoutecost		NRoutetime		Q
41,34		41,34		81
23,62		23,62		76
58,70		58,70		125
75,24		75,24		120
62,56		62,56		139
99,59		99,59		139
74,16		74,16		139
75,74		75,74		139
106,12		106,12		132
176,61		176,61		136
105,89		105,89		138

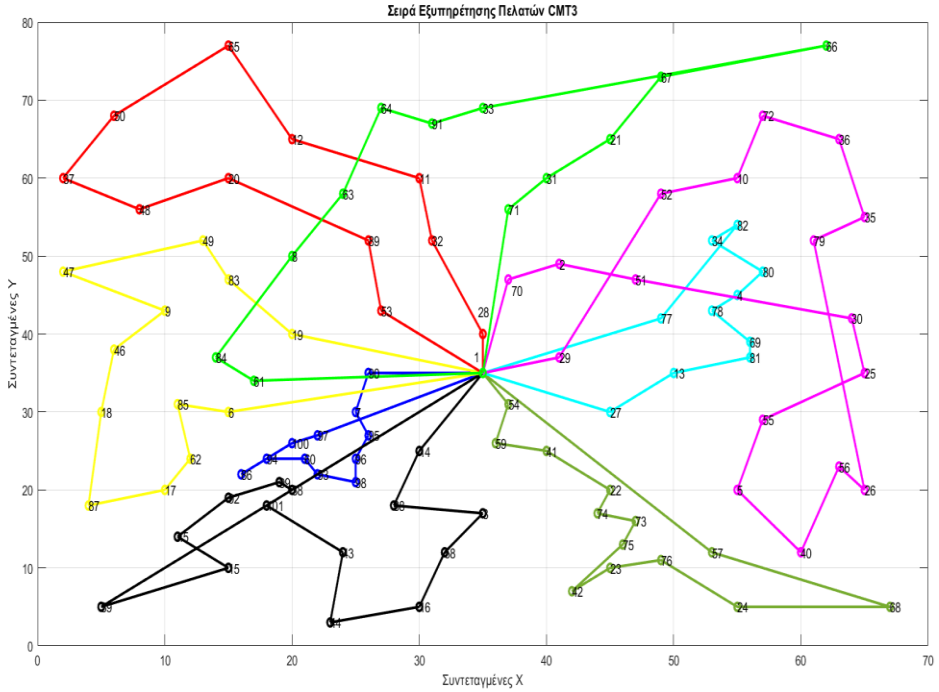
Αποτελέσματα CMT 3

n=76

Qmax=140

Tmax=999999

servicetime=0



Συνολικό Κόστος : 919,51

old_seira	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	54	59	41	22	74	73	75	23	76	57	40	24	1	0	0	0	0	0	0
2	1	28	29	27	13	81	69	78	4	80	34	82	1	0	0	0	0	0	0	0
3	1	90	7	95	96	98	93	60	100	97	94	86	92	1	0	0	0	0	0	0
4	1	14	88	3	58	16	44	43	101	38	99	62	17	45	15	1	0	0	0	0
5	1	53	19	61	84	85	6	18	46	9	83	49	48	37	1	0	0	0	0	0
6	1	70	2	51	77	30	25	55	56	26	5	42	68	79	35	36	72	10	52	1
7	1	32	89	8	63	11	91	33	64	12	20	50	65	1	0	0	0	0	0	0
8	1	71	31	21	67	66	47	87	39	1	0	0	0	0	0	0	0	0	0	0

seira	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	54	59	41	22	74	73	75	42	23	76	24	68	57	1	0	0	0	0
2	1	27	13	81	69	78	4	80	34	82	77	1	0	0	0	0	0	0	0
3	1	90	7	95	96	98	93	60	94	86	100	97	1	0	0	0	0	0	0
4	1	14	88	3	58	16	44	43	101	38	99	92	45	15	39	1	0	0	0
5	1	6	85	62	17	87	18	46	9	47	49	83	19	1	0	0	0	0	0
6	1	70	2	51	30	25	55	5	40	56	26	79	35	36	72	10	52	29	1
7	1	28	32	11	12	65	50	37	48	20	89	53	1	0	0	0	0	0	0
8	1	71	31	21	67	66	33	91	64	63	8	84	61	1	0	0	0	0	0

Routecost		Routetime		Demand
93,32		93,32		196
82,67		82,67		197
69,90		69,90		191
123,55		123,55		188
131,98		131,98		179
236,76		236,76		195
150,02		150,02		180
165,46		165,46		132

NRoutecost		NRoutetime		Q
111,17		111,17		195
75,56		75,56		178
57,81		57,81		190
137,84		137,84		173
120,05		120,05		192
188,94		188,94		199
120,81		120,81		177
107,32		107,32		154

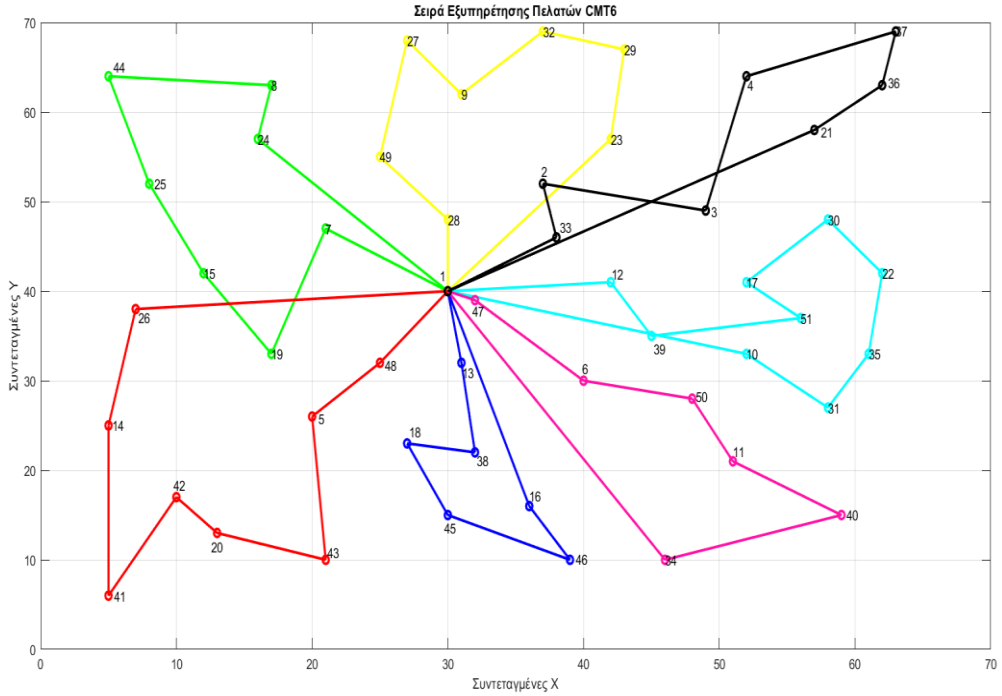
Αποτελέσματα CMT 6

n=51

Qmax=160

Tmax=200

servicetime=10



Συνολικό Κόστος : 659,41

old_seira	1	2	3	4	5	6	7	8
1	1	47	13	48	5	18	38	1
2	1	28	2	33	12	39	6	1
3	1	7	49	9	27	1	0	0
4	1	19	15	26	14	1	0	0
5	1	23	29	32	1	0	0	0
6	1	3	17	51	10	1	0	0
7	1	50	11	31	35	1	0	0
8	1	24	8	44	1	0	0	0
9	1	16	45	43	1	0	0	0
10	1	25	42	20	1	0	0	0
11	1	30	22	21	1	0	0	0
12	1	46	34	40	1	0	0	0
13	1	4	36	37	1	0	0	0
14	1	41	1	0	0	0	0	0

seira	1	2	3	4	5	6	7	8	9	10	11
1	1	23	29	32	9	27	49	28	1	0	0
2	1	10	31	35	22	30	17	51	39	12	1
3	1	24	8	44	25	15	19	7	1	0	0
4	1	16	46	45	18	38	13	1	0	0	0
5	1	48	5	43	20	42	41	14	26	1	0
6	1	34	40	11	50	6	47	1	0	0	0
7	1	21	36	37	4	3	2	33	1	0	0

Routecost		Routetime		Demand
53,9429		123,9429		90
56,4696		126,4696		107
64,9369		114,9369		73
73,7713		123,7713		140
67,0160		107,0160		49
63,9683		113,9683		84
76,9573		126,9573		76
74,8025		114,8025		56
72,4379		112,4379		48
97,0231		137,0231		53
85,5446		125,5446		59
90,5377		130,5377		64
92,6220		132,6220		46
84,4038		104,4038		7

NRoutecost		NRoutetime		Q
83		153,3690		111
99		189,3533		137
100		169,5443		143
73		133,4976		86
114		194,1422		148
88		148,0680		103
101		171,4360		123

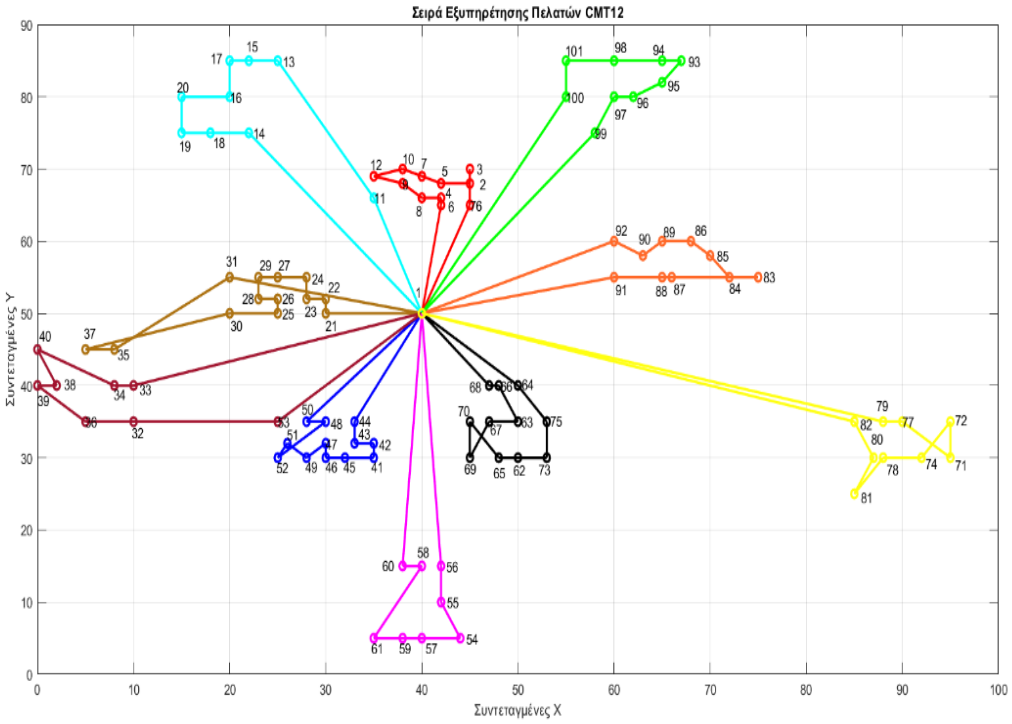
Αποτελέσματα CMT 12

n=101

Qmax=200

Tmax=999999

servicetime=0



Συνολικό Κόστος: 831,06

old_seira	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	21	22	23	24	27	29	28	26	25	30	31	35	37	1	0
2	1	68	66	64	63	67	70	69	65	62	73	75	1	0	0	0
3	1	6	4	5	7	9	10	12	11	8	76	2	3	99	1	0
4	1	44	43	42	41	45	46	47	49	51	52	53	50	48	32	1
5	1	91	90	89	86	85	84	83	87	88	92	97	1	0	0	0
6	1	14	18	19	20	16	17	15	13	1	0	0	0	0	0	0
7	1	33	34	36	38	39	40	60	58	1	0	0	0	0	0	0
8	1	100	101	98	94	93	95	96	82	1	0	0	0	0	0	0
9	1	56	55	54	57	59	61	81	80	78	74	1	0	0	0	0
10	1	79	77	72	71	1	0	0	0	0	0	0	0	0	0	0

seira	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	21	22	23	24	27	29	28	26	25	30	37	35	31	1
2	1	68	66	63	67	69	70	65	62	73	75	64	1	0	0
3	1	6	4	8	9	12	10	7	5	2	3	76	1	0	0
4	1	44	43	42	41	45	46	47	49	51	52	48	50	1	0
5	1	92	90	89	86	85	84	83	87	88	91	1	0	0	0
6	1	14	18	19	20	16	17	15	13	11	1	0	0	0	0
7	1	33	34	40	38	39	36	32	53	1	0	0	0	0	0
8	1	100	101	98	94	93	95	96	97	99	1	0	0	0	0
9	1	56	55	54	57	59	61	58	60	1	0	0	0	0	0
10	1	79	77	71	72	74	78	81	80	82	1	0	0	0	0

Routecost	Routetime		Demand
92,98	92,98		200
63,06	63,06		200
90,60	90,60		200
101,99	101,99		180
113,25	113,25		180
95,88	95,88		190
137,70	137,70		200
155,72	155,72		190
174,39	174,39		190
62,29	62,29		80

NRoutecost	NRoutetime		Q
89,05	89,05		200
67,78	67,78		200
57,57	57,57		170
66,73	66,73		150
76,84	76,84		170
96,04	96,04		200
98,73	98,73		180
95,94	95,94		190
102,68	102,68		200
79,70	79,70		150

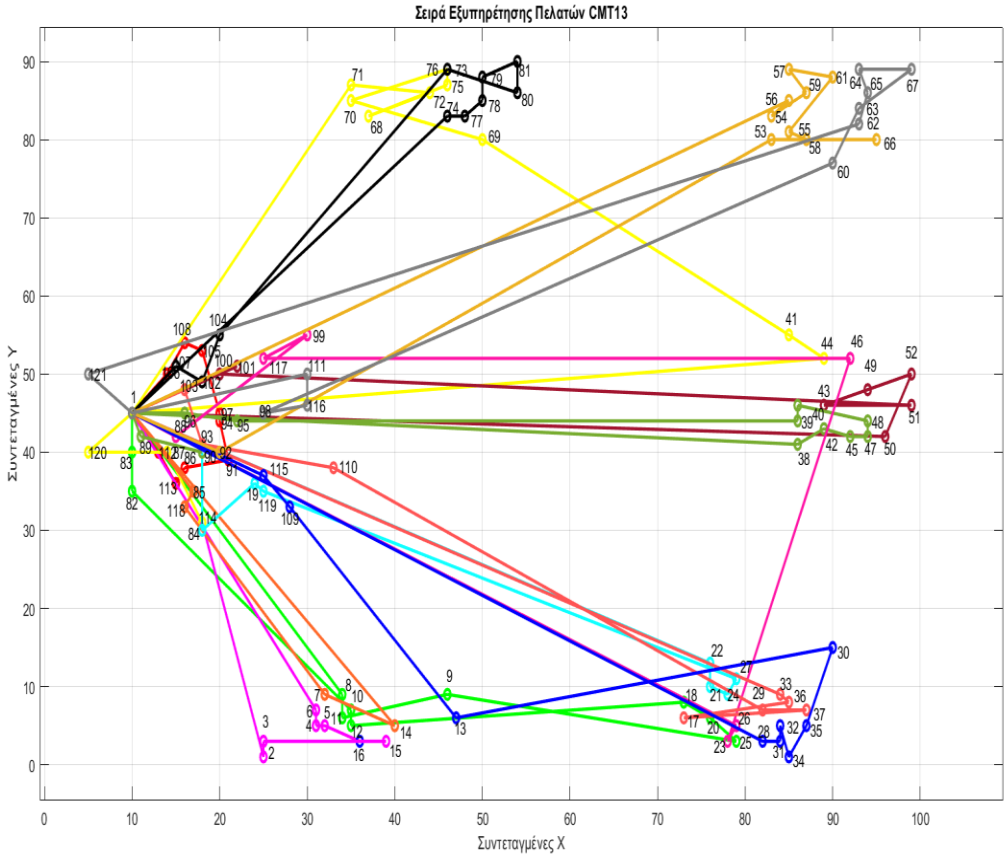
Αποτελέσματα CMT 13

n=121

Qmax=200

Tmax=720

servicetime=50



Συνολικό Κόστος: 1770,36

old_seira	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	89	83	112	87	88	93	90	92	91	19	115	1
2	1	96	103	102	100	101	117	111	116	98	95	1	0
3	1	106	107	108	105	104	97	94	86	113	85	118	1
4	1	120	82	114	84	119	109	110	99	121	1	0	0
5	1	7	8	10	11	12	16	1	0	0	0	0	0
6	1	6	4	5	3	2	15	1	0	0	0	0	0
7	1	68	70	71	72	75	73	1	0	0	0	0	0
8	1	14	13	9	18	17	1	0	0	0	0	0	0
9	1	74	77	78	79	76	81	1	0	0	0	0	0
10	1	69	80	54	55	53	1	0	0	0	0	0	0
11	1	22	21	24	27	29	1	0	0	0	0	0	0
12	1	41	44	46	49	1	0	0	0	0	0	0	0
13	1	39	40	43	42	1	0	0	0	0	0	0	0
14	1	38	45	47	48	1	0	0	0	0	0	0	0
15	1	20	26	25	23	1	0	0	0	0	0	0	0
16	1	33	36	37	35	1	0	0	0	0	0	0	0
17	1	28	31	32	34	1	0	0	0	0	0	0	0
18	1	58	60	62	63	1	0	0	0	0	0	0	0
19	1	56	59	57	61	1	0	0	0	0	0	0	0
20	1	30	50	51	52	1	0	0	0	0	0	0	0
21	1	66	65	64	67	1	0	0	0	0	0	0	0

seira	1	2	3	4	5	6	7	8	9	10	11	12
1	1	106	108	105	97	94	91	86	113	112	1	0
2	1	8	11	9	25	20	18	12	10	82	83	1
3	1	6	4	5	16	15	3	2	114	87	120	1
4	1	71	72	68	75	76	70	69	41	44	1	0
5	1	14	7	118	85	1	0	0	0	0	0	0
6	1	74	77	78	79	81	80	73	104	102	107	1
7	1	50	52	49	43	51	100	101	1	0	0	0
8	1	22	21	24	27	119	19	84	90	89	1	0
9	1	38	42	45	47	48	40	39	95	96	1	0
10	1	26	23	46	117	99	88	1	0	0	0	0
11	1	33	36	17	37	29	110	93	103	1	0	0
12	1	28	31	32	34	35	30	13	109	115	1	0
13	1	56	54	59	57	61	55	58	66	53	92	1
14	1	121	62	65	64	67	63	60	98	116	111	1

Routecost		Routetime		Demand
418,6776		641,8678		135
485,5854		598,5585		109
543,7982		654,3798		100
931,9419		593,1942		120
100,8870		450,8870		81
120,8780		470,8780		90
121,5860		471,5860		113
163,3464		463,3464		97
135,2953		485,2953		97
175,5379		475,5379		60
167,2331		467,2331		54
172,1894		422,1894		62
163,0319		413,0319		70
170,1939		420,1939		51
162,7488		412,7488		59
174,7123		424,7123		23
178,4318		428,4318		43
188,3610		438,3610		65
186,7646		436,7646		61
211,2390		461,2390		56
107,1689		307,1689		60

NRoutecost		NRoutetime		Q
45,0947		495,0947		88
187,6800		687,6800		139
126,6145		626,6145		150
232,2346		682,2346		181
109,4424		309,4424		52
137,7318		637,7318		105
210,1203		560,1203		83
170,2556		620,2556		95
173,2020		623,2020		116
231,4641		531,4641		86
202,1757		602,1757		79
205,3371		655,3371		96
218,1775		718,1775		126
132,3022		632,3022		134

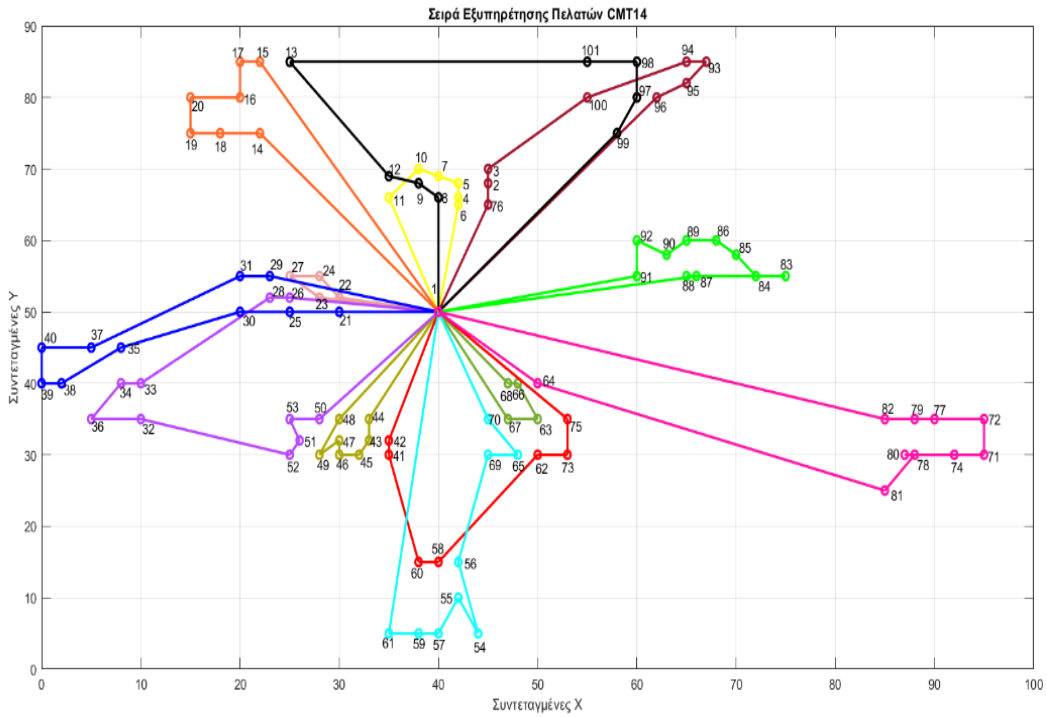
Αποτελέσματα CMT 14

n=101

Qmax=200

Tmax=1040

servicetime=90



Συνολικό Κόστος: 1004,92

old_seira	1	2	3	4	5	6	7	8	9	10	11
1	1	21	22	23	24	27	29	28	26	25	1
2	1	68	66	64	63	67	70	69	65	62	1
3	1	6	4	5	7	9	10	12	11	8	1
4	1	76	2	3	99	97	96	95	94	1	0
5	1	44	43	42	41	45	46	47	49	51	1
6	1	48	50	53	52	32	33	34	35	1	0
7	1	75	73	56	58	60	55	54	1	0	0
8	1	30	31	37	40	39	38	36	1	0	0
9	1	91	90	89	86	85	84	83	87	1	0
10	1	92	88	100	101	98	93	13	1	0	0
11	1	14	18	19	20	16	17	15	1	0	0
12	1	57	59	61	81	80	78	1	0	0	0
13	1	82	79	77	72	71	74	1	0	0	0

seira	1	2	3	4	5	6	7	8	9	10	11	12
1	1	22	24	27	23	1	0	0	0	0	0	0
2	1	68	66	63	67	1	0	0	0	0	0	0
3	1	6	4	5	7	10	11	1	0	0	0	0
4	1	76	2	3	100	94	93	95	96	1	0	0
5	1	44	43	45	46	47	49	48	1	0	0	0
6	1	50	53	51	52	32	36	34	33	28	26	1
7	1	75	73	62	58	60	41	42	1	0	0	0
8	1	29	31	37	40	39	38	35	30	25	21	1
9	1	91	92	90	89	86	85	84	83	88	87	1
10	1	99	97	98	101	13	12	9	8	1	0	0
11	1	14	18	19	20	16	17	15	1	0	0	0
12	1	61	59	57	55	54	56	69	65	70	1	0
13	1	82	79	77	72	71	74	80	78	81	64	1

Routecost		Routetime		Demand
44,0000		944,0000		160
57,5672		957,5672		150
51,7672		951,7672		140
91,7421		901,7421		190
59,0133		959,0133		130
88,2274		898,2274		160
104,4162		824,4162		180
98,9376		818,9376		130
75,5970		885,5970		160
153,4364		873,4364		140
94,1632		814,1632		200
162,2368		792,2368		120
65,4342		605,4342		120

NRoutecost		NRoutetime		Q
33,2118		393,2118		70
38,1447		398,1447		60
44,3679		584,3679		90
92,5471		812,5471		190
52,0304		682,0304		110
91,0746		991,0746		200
83,8558		713,8558		140
91,5581		991,5581		180
81,9599		981,9599		190
117,0497		837,0497		160
94,1632		724,1632		200
111,1847		921,1847		190
73,7726		973,7726		200

6.3 Συμπεράσματα

Καθοριστικό ρόλο στην εξέλιξη του προγράμματος και κατά συνέπεια και στα αποτελέσματα του έχει η επιλογή των μεθόδων που χηρσιμοποιούνται για τη βελτιστοποίηση του. Στη παρούσα διπλωματική εργασία υλοποιήθηκε και εκτελέστηκε αλγόριθμος επίλυσης του προβλήματος δρομολόγησης οχημάτων με περιορισμένη χωρητικότητα. Ο αλγόριθμος εφαρμόστηκε για 14 παραδείγματα CMT (Πίνακας 6.1), τα οποία χωρίστηκαν σε δύο κατηγορίες με βάση τα δεδομένα τους όπως παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Σε πρώτη φάση βρέθηκε μια αρχική λύση χρησιμοποιώντας τον αλγόριθμο του Πλησιέστερου Γείτονα, η οποία όμως ήταν αρκετά μακριά από τη γνωστή βέλτιστη τιμή ειδικά για τα παραδείγματα που επηρεάζονται και από τους δύο περιορισμούς, δηλαδή τη χωρητικότητα και το μέγιστο επιτρεπτό χρονικό όριο. Παρατηρήθηκε ότι ο αλγόριθμος του Πλησιέστερου Γείτονα δε διευκολύνει το έργο των τοπικών αναζητήσεων και της προσομοιωμένης ανόπτησης αντίθετα το δυσχαιρένει. Αυτό συμβαίνει διότι πρόκειται για έναν απλό ευρετικό αλγόριθμο που λειτουργεί με απληστία προσπαθώντας να επισκέπτεται πάντα τον κοντινότερο πελάτη με συνέπεια να αγνοεί διαδρομές με μειωμένο κόστος. Η αρχική λύση βελτιώθηκε σε μεγάλο βαθμό με τη χρήση των μεθόδων τοπικής αναζήτησης 1-0 επανατοποθέτηση, 1-1 ανταλλαγή και εσωτερική ανταλλαγή. Στο τέλος της διαδικασίας εφαρμόστηκε ο μεθευρετικός αλγόριθμος της Προσομοιωμένης Ανόπτησης για περαιτέρω βελτίωση της λύσης. Για να επιτευχθεί κάτι τέτοιο όμως χρειάστηκαν πολλές δοκιμές κατά την εκτέλεση του προγράμματος έτσι ώστε να δωθούν από τον χρήστη τα κατάλληλα ορίσματα σε μεταβλητές που ήταν καθοριστικές για την πορεία του. ‘Ετσι δόθηκε η δυνατότητα να προσεγγιστούν οι βέλτιστες λύσεις σύμφωνα με τη βιβλιογραφία με ένα μέσο ποσοστό απόκλισης 9% για τα σενάρια της $1^{ης}$ κατηγορίας στα οποία δεν υπάρχει χρόνος εξυπηρέτησης και δεν λαμβάνεται υπόψη ο περιορισμός του χρονικού ορίου, κάτι που οδηγεί στο συμπέρασμα ότι ο αλγόριθμος λειτούργησε εξαιρετικά για τα συγκεκριμένα παραδείγματα. Οσον αναφορά τα παραδείγματα της $2^{ης}$ κατηγορίας, ο αλγόριθμος λειτούργησε και εκεί ικανοποιητικά με μέσο ποσοστό απόκλισης 21%. Σε αυτή τη κατηγορία με βάση την επιλογή των μεθόδων βελτιστοποίησης ήταν εκ φύσεως αρκετά δύσκολη η προσέγγιση των ήδη γνωστών βέλτιστων τιμών καθώς υπήρχαν περιορισμοί. Ειδικότερα ο περιορισμός της χωρητικότητας αποτέλεσε το κυριότερο πρόβλημα, διότι γίνεται προσπάθεια να χρησιμοποιηθούν οι μικρότερες δυνατές αποστάσεις χωρίς να δίνεται η δυνατότητα να χωρέσουν, δηλαδή να εξυπηρετηθούν αρκετοί πελάτες σε μια διαδρομή. ’Ετσι πολλές φορές απορρίπτονται συνδιασμοί πελατών που ελαχιστοποιούν το κόστος εξαιτίας του συγκεκριμένου περιορισμού, με το όχημα να αναγκάζεται να επιστρέφει συχνά στην αποθήκη με αποτέλεσμα τη δημιουργία πολλών νέων μικρότερων διαδρομών. Για αυτό τον λόγο χρειάστηκε να γίνουν περισσότερες επαναλήψεις προκειμένου να μειωθεί το ολικό κόστος, αλλά και το πλήθος των διαδρομών των παραδειγμάτων αυτης της κατηγορίας, κάτι που σε αυτή τη περίπτωση μεγαλώνει τη χρονική διάρκεια της εκτέλεσης του προγράμματος. Παρατηρείται λοιπόν, ότι με τη προσθήκη περισσότερων περιορισμών επηρεάζεται τόσο η απόδοση όσο και η διάρκεια του αλγορίθμου. Τέλος, μελλοντικά για ακόμα μεγαλύτερη βελτίωση της ποιότητας της λύσης θα μπορούσε να γίνεται στοχευμένη και όχι τυχαιοποιημένη επιλογή εξέτασης κόμβων στις μεθόδους βελτίωσης της αρχικής λύσης.

Βιβλιογραφία

[1]: Μαρινάκης Ιωάννης, Μυγδαλάς Αθανάσιος (2008). *Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας* . Σελίδες 17-25, 135-142, 373-376.

[2]: Μαλινδρέτος Γ. (2015). *Εφοδιαστική αλυσίδα, logistics και εξυπηρέτηση πελατών*. Σελίδες 25-48

[3]: Σαρτζετάκη Κ. (2013), *Logistics και Εφοδιαστική Αλυσίδα σε μια επιχείρηση*, Πτυχιακή Εργασία, Τμήμα Λογιστικής, Τ.Ε.Ι. Κρήτης. Σελίδες 7-20.

[4]: Μπεχράκη Κ. (2016) ,*Οργάνωση και Διαχείριση Αλυσίδας Εφοδιασμού με τη χρήση Συστημάτων ERP*,Μεταπτυχιακή Εργασία ,Τμήμα Βιομηχανικής Διοίκησης και Τεχνολογίας, ΠΑ.ΠΕΙ. Σελίδες 11-24.

[5]: https://www.supplychain.gr/βιβλιοθήκη/26-τι-είναι-τα-logistics.html

[6]: https://www.slideshare.net/alexandrospapaspurou/logistics-48011400

[7]: Fisher, M. L., Jaikumar, R., and Wassenhove, L. N. V (1986). *A Multiplier Adjustment Method for the Generalized Assigment Problem, Management Science.* Σελίδες 1095-1103.

[8]: https://learningactors.com/unsupervised-learning-randomized-optimization/

[9]: https://www.slideshare.net/idforjoydutta/simulated-annealing-24528483

[10]: Emmanouil E. Zachariadis, Chris T. Kiranoudis (2010), *A Strategy for Reducing the Computational Complexity of Local Search-Based Methods, and its Application to the Vehicle Routing Problem*, Department of Process Analysis and Plant Design, National Technical University of Athens. Σελίδες 7-10.

[11]: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

[12]: Σιδεριάδης Κ. (2016) , *Μελέτη περίπτωσης ενός προβλήματος δρομολόγησης οχημάτων με περιορισμένη χωρητικότητα (CVRP)*, Μεταπτυχιακή Διατριβή, Τμήμα Πληροφορικής, ΠΑ.ΠΕΙ. Σελίδες 62-68.