ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Υβριδικός Αλγόριθμος Νυχτερίδας για προβλήματα δρομολόγησης οχημάτων

A Hybrid Bat Algorithm for solving vehicle routing problems

Διπλωματική Εργασία

Οικονομάκης Αίας Εφραίμ

Επιβλέπων καθηγητής:

Δρ. Ιωάννης Μαρινάκης

Χανιά 2024

Περιεχόμενα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου Δρ. Ι. Μαρινάκη για την άριστη επικοινωνία και πολύτιμη βοήθεια που μου προσέφερε ως προς την ολοκλήρωση της εργασίας και τον εργαστηριακό καθηγητή κ. Ν. Κυριακάκη για την επίσης σημαντική προσφορά του στην επίλυση απορίων σχετικά με τον κώδικα. Στη συνέχεια, οι γονείς μου αξίζουν ένα μεγάλο «ευχαριστώ» που με στήριξαν με κάθε δυνατό τρόπο σε όλη την πορεία της φοίτησης μου. Τέλος θα ήθελα να ευχαριστήσω του καθηγητές και τους συμφοιτητές μου που συνέβαλλαν σε μια όμορφη φοιτητική ζωή.

Περίληψη

Το πρόβλημα της δρομολόγησης οχημάτων υπάγεται στον κλάδο της εφοδιαστικής αλυσίδας. Πρωταρχικός στόχος των προβλημάτων αυτής της κατηγορίας είναι η εύρεση της βέλτιστης διαδρομής ενός στόλου οχημάτων ξεκινώντας από μια κεντρική αποθήκη, με σκοπό την εξυπηρέτηση μιας ομάδας πελατών με συγκεκριμένη ζήτηση, ελαχιστοποιώντας το κόστος διαδρομής. Τα προβλήματα δρομολόγησης ποικίλουν σε σχέση με διάφορες παραμέτρους και περιορισμούς (π.χ ζήτηση, χρονικά περιθώρια, χωρητικότητα οχημάτων). Στην συγκεκριμένη διπλωματική εργασία θα γίνει χρήση του αλγορίθμου της νυχτερίδας για την επίλυση ορισμένων βασικών προβλημάτων δρομολόγησης. Ο Bat Algorithm (BA) υπάγεται στην κατηγορία των μεθευρετικών αλγορίθμων βελτιστοποίησης και παρουσιάστηκε για πρώτη φορά από τον Xin-She Yang το 2010. Ο αλγόριθμος προήλθε από την προσεκτική παρατήρηση του τρόπου αναζήτησης τροφής από το σμήνος ενός είδους νυκτερίδων, με χρήση της ιδιότητας ηχοεντοπισμού που διαθέτουν. Κάθε νυχτερίδα προσανατολίζεται και βρίσκει την τροφή της εκπέμποντας υψηλής συχνότητας ηχητικά κύματα, τα οποία αντανακλούν πάνω σε αντικείμενα και υποψήφια θηράματα. Ο πρωτότυπος αλγόριθμος επινοήθηκε για την επίλυση προβλημάτων συνεχών μεταβλητών, όμως στην παρούσα εργασία θα χρησιμοποιηθεί μια υβριδοποιημένη παραλλαγή για διακριτές μεταβλητές. Η υλοποίηση του αλγόριθμου θα επιτευχθεί με τη χρήση γλώσσας προγραμματισμού Python έκδοσης 3.10 και στη συνέχεια θα αναλυθούν και θα αξιολογηθούν τα αποτελέσματα κάθε προβλήματος.

Λέξεις κλειδιά: Διαχείριση εφοδιαστικής αλυσίδας, Πρόβλημα Δρομολόγησης Οχημάτων, ευρετικοί, μεθευρετικοί αλγόριθμοι, υβριδικός, Αλγόριθμος Νυχτερίδας, νοημοσύνη σμήνους, τοπική αναζήτηση

Abstract

The Vehicle Routing Problem (VRP) belongs to the industrial supply chain sector. The primary objective of the problems in this category is to generate the optimal route plan of a fleet of vehicles starting from a central depot, aiming to serve a group of customers with specific demand, minimizing the total route cost. Vehicle routing problems vary with respect to several factors and constraints (eg demand, time windows and vehicle capacity). This thesis will demonstrate the use of the bat algorithm to solve some basic routing problems. The Bat Algorithm (BA) is a meta-heuristic algorithm and it was found by Xin-She Yang in 2010. The name of this algorithm indicates that it is inspired by nature, specifically by the echolocation ability of a bat swarm. Each bat orients itself and finds its prey by emitting sound waves at a frequency above human hearing that reflect off objects and potential prey. The original Bat Algorithm was created to solve continuous variable problems, but this thesis will present a hybrid variant for discrete variables. The algorithm will be implemented using Python 3.10, afterwards the results of each problem will be analyzed and evaluated.

Key words: Supply chain Management, Vehicle Routing Problem, heuristics, meta-heuristics, hybrid, Bat Algorithm, swarm intelligence

Κεφάλαιο 1ο Η εφοδιαστική αλυσίδα και τα logistics

Εισαγωγή

Το πρώτο κεφάλαιο της παρούσας εργασίας εισάγει την έννοια της εφοδιαστικής αλυσίδας στον χώρο των επιχειρήσεων και αναλύει τη σημαντικότητα της διαχείρισής της. Στη συνέχεια περιγράφεται ο ρόλος των logistics και η σημαντική προσφορά τους στη διαχείριση της εφοδιαστικής αλυσίδας για την επίτευξη κερδοφορίας σε μια επιχείρηση. Τέλος, γίνεται μια εισαγωγή στις μεταφορές, μια από τις σημαντικότερες δραστηριότητες της εφοδιαστικής αλυσίδας.

1.1 Η εφοδιαστική αλυσίδα (supply chain)

1.1.1 Ορισμός της εφοδιαστικής αλυσίδας

Με τον όρο "εφοδιαστική αλυσίδα" (supply chain) περιγράφονται τα εξής: α) Το δίκτυο που περιλαμβάνει όλες τις διαδικασίες που σχετίζονται με τη διαχείριση των πρώτων υλών μέχρι την διανομή του τελικού προϊόντος στον καταναλωτή. β) Η ροή της πληροφορίας ανάμεσα στις υποδομές που θα συμβάλουν στην εξυπηρέτηση και ικανοποίηση των αναγκών του πελάτη. Οι οντότητες που απαρτίζουν το δίκτυο της εφοδιαστικής αλυσίδας είναι οι προμηθευτές των πρώτων υλών, οι αποθήκες, οι μονάδες παραγωγής, τα τελικά προϊόντα, τα κέντρα διανομής, οι διανομείς, οι πωλητές και οι τελικοί πελάτες - καταναλωτές.

Όπως ορίζει ο Quinn (1997)[1], «η εφοδιαστική αλυσίδα περιλαμβάνει όλες τις δραστηριότητες που σχετίζονται με μεταφορά αγαθών από την μορφή πρώτης ύλης μέχρι τον τελικό καταναλωτή».

1.1.2 Διαχείριση της εφοδιαστικής αλυσίδας (Supply chain management)

Σύμφωνα με το Supply Chain Council (1996) «η διαχείριση της εφοδιαστικής αλυσίδας (SCM) αποτελεί την προσπάθεια παραγωγής και διανομής ενός τελικού προϊόντος από τον προμηθευτή του προμηθευτή στον πελάτη του πελάτη»[1]. Η διαχείριση της εφοδιαστικής αλυσίδας ορίστηκε επίσης ως "η ενσωμάτωση στον επιχειρησιακό χώρο μιας φιλοσοφίας που διασφαλίζει την συνολική ροή αγαθών ενός καναλιού διανομής από τον προμηθευτή στον τελικό καταναλωτή " (Ellmar, Cooper, 1993)[2]. Με βάση τους ορισμούς άλλων επιστημόνων η "φιλοσοφία" μεταφράζεται ως ένα σύνολο δραστηριοτήτων που συμβάλλουν στην παραγωγή και παράδοση του τελικού προϊόντος στον πελάτη (Quinn, 1997)[1].

Η σωστή διαχείριση της εφοδιαστικής αλυσίδας αποτελεί προσόν στο επιχειρηματικό περιβάλλον, καθώς δημιουργεί συνθήκες κερδοφορίας και ενθαρρύνει την ανάπτυξη νέων προϊόντων. Οι σύγχρονες επιχειρήσεις καλούνται να προσαρμόσουν την στρατηγική τους σε αρκετά ανταγωνιστικές συνθήκες, στοχεύοντας να προσφέρουν όσο το δυνατόν καλύτερες υπηρεσίες εξυπηρέτησης του πελάτη και να διαθέσουν προϊόντα σε ανταγωνιστικές τιμές.

Οι συντελεστές της διαχείρισης της εφοδιαστικής αλυσίδας εκτελούν διαχείριση σε επτά επιχειρησιακές δραστηριότητες (Cooper, 1997)[1],[2]:

- τις πελατειακές σχέσεις,
- την εξυπηρέτηση πελατών,
- τη ζήτηση και τα αποθέματα,
- την εκπλήρωση παραγγελιών,
- τη ροή παραγωγής,
- τη διαχείριση προμηθειών,
- την ανάπτυξη προϊόντων και την εμπορευματοποίηση τους.

1.2 Logistics

Με τον όρο "logistics" εννοούμε την «διαδικασία στρατηγικής διαχείρισης των προμηθειών, μετακίνησης και αποθήκευσης των πρώτων υλών, των ημικατεργασμένων και των τελικών προϊόντων (καθώς και της σχετικής πληροφορίας) δια μέσου μιας επιχείρησης και των καναλιών Marketing, με τέτοιο τρόπο, ώστε η τωρινή και μελλοντική κερδοφορία να μεγιστοποιούνται κατά την εκπλήρωση των παραγγελιών» (Christopher, 2016)[3].

Τα logistics σε μια επιχείρηση αποτελούν ουσιαστικά το τμήμα που ασχολείται με τη διαχείριση της εφοδιαστικής αλυσίδας, του οποίου θεμελιώδης σκοπός είναι η διασφάλιση της κερδοφορίας. Αυτό επιτυγχάνεται για παράδειγμα, μέσω της συνεχούς εξασφάλισης διαθεσιμότητας των πρώτων υλών και αποθεμάτων, της στρατηγικής διατήρησης και αποθήκευσης των αγαθών, του κατάλληλου σχεδιασμού της παραγωγής και τη μεταφορά των τελικών προϊόντων με τον βέλτιστο δυνατό τρόπο, ώστε να εξασφαλίζεται η καλύτερη εξυπηρέτηση με το ελάχιστο κόστος.

Για να επιτευχθεί κερδοφορία σε μια εφοδιαστική αλυσίδα, το τμήμα logistics θα πρέπει να υπακούει στις παρακάτω βασικές αρχές:

- Ελαχιστοποίηση του λειτουργικού κόστους
- Ελάχιστο δυνατό κόστος επενδύσεων
- Βελτιστοποίηση της ποιότητας προϊόντων και υπηρεσιών
- Υψηλό επίπεδο εξυπηρέτησης πελατών

Πρέπει να σημειωθεί πως η τελική αξία ενός προϊόντος που φτάνει στα χέρια του καταναλωτή δεν αποτελεί μόνο το κόστος της παραγωγικής διαδικασίας, μέσω της οποίας οι πρώτες ύλες μετατρέπονται στα τελικά προϊόντα, αλλά ένα μεγάλο ποσοστό προέρχεται από τις λειτουργίες της εφοδιαστικής αλυσίδας. Οι δαπάνες που επιφέρουν προσθήκη αξίας στο προϊόν προέρχονται από δραστηριότητες όπως η μεταφορά με χρήση ανθρώπινου δυναμικού και οχημάτων, η αποθήκευση και η συντήρηση, η χρήση κτιρίων και μηχανημάτων, η λειτουργία και ο έλεγχος ενός δικτύου ροής πληροφορίας κ.α.

Παρόλο που τα έσοδα μιας επιχείρησης πηγάζουν αποκλειστικά από την πώληση των προϊόντων της, το τμήμα logistics, προκειμένου να επιτύχει οικονομική αποδοτικότητα, επιδιώκει την ελαχιστοποίηση του κόστους παραγωγής. Είναι ζωτικής σημασίας η υιοθέτηση μιας καλά σχεδιασμένης στρατηγικής που θα διαχειρίζεται όχι μόνο το κόστος κάθε μιας εκ των επιμέρους δαπανών που προαναφέρθηκαν, ξεχωριστά, αλλά και του συνολικού κόστους. Η κατάλληλη διαχείριση του κόστους κατά σύνολο μπορεί να βοηθήσει στην αποφυγή

εμφάνισης αυξημένου ποσού δαπάνης σε μία δραστηριότητα και στην ύπαρξη οικονομικής ισορροπίας μεταξύ των δραστηριοτήτων της εφοδιαστικής αλυσίδας.

1.2.1 Δραστηριότητες των logistics

Το τμήμα logistics, όπως αναφέρθηκε παραπάνω, επιτελεί την διαχείριση της εφοδιαστικής αλυσίδας με σκοπό την ελαχιστοποίηση του συνολικού κόστους προϊόντος, της μεγιστοποίησης του κέρδους και της άριστης εξυπηρέτησης του πελάτη. Τα logistics μεσολαβούν και μεταχειρίζονται δραστηριότητες που ανήκουν στην εφοδιαστική αλυσίδα, αντιμετωπίζοντάς τες ως μια ολοκληρωμένη διαδικασία. Οι βασικές δραστηριότητες που διαχειρίζονται τα logistics είναι :

- η διαχείριση αποθεμάτων
- η διαχείριση παραγγελιών
- η διανομή και οι μεταφορές
- η αποθήκευση
- η πληροφόρηση

1.2.2 Συσχέτιση logistics με ΔΕΑ

Η βασική διαφορά μεταξύ των logistics και της διαχείρισης της εφοδιαστικής αλυσίδας είναι ότι τα πρώτα επικεντρώνονται συγκεκριμένα στις δραστηριότητες της επιχείρησης που προσθέτουν αξία στο προϊόν (αποθήκευση, μεταφορά κλπ) ενώ η διαχείριση της εφοδιαστικής αλυσίδας είναι ένα ευρύτερο πλαίσιο, που περιλαμβάνει τον συντονισμό όλων των διαδικασιών από τον προμηθευτή έως τον τελικό καταναλωτή. Για παράδειγμα, η διαχείριση της εφοδιαστικής αλυσίδας περιλαμβάνει τον συντονισμό των πληροφοριακών, οικονομικών και επιχειρησιακών πτυχών της εφοδιαστικής διαδικασίας. Συνεπώς, η διαχείριση της εφοδιαστικής αλυσίδας αποτελεί ένα στρατηγικό πλαίσιο που αποσκοπεί, πέρα από τη μείωση κόστους, στην επίτευξη βέλτιστης απόδοσης και ενίσχυσης της ευελιξίας σε όλη την αλυσίδα προμηθειών και διανομής.

1.2.3 Μεταφορές

Εφόσον τα έξοδα των μεταφορών αποτελούν ίσως το μεγαλύτερο μέρος των εξόδων της εφοδιαστικής αλυσίδας και επιφέρουν προστιθέμενο κόστος στην τιμή του προϊόντος, είναι πολύ σημαντική η στρατηγική διαχείρισή τους. Η επιχείρηση επιστρατεύοντας το σύστημα logistics, καλείται να επιλύσει ζητήματα όπως τον σχεδιασμό ενός κατάλληλου δικτύου διανομής και τον προγραμματισμό δρομολογίων, έτσι ώστε να ελαχιστοποιείται το κόστος καθώς και ο χρόνος μεταφοράς. Οι μεταφορές διακρίνονται σε δύο κατηγορίες, τις εσωτερικές και τις εξωτερικές (inbound and outbound logistics). Οι πρώτες αναφέρονται σε μετακινήσεις πρώτων υλών και προϊόντων μέσα από σταθμούς παραγωγής της επιχείρησης, καθώς και από τις αποθήκες σε σημεία πώλησης. Οι εξωτερικές μεταφορές αφορούν τη διανομή των τελικών προϊόντων από τις αποθήκες στους πελάτες, άμεσα ή έμμεσα, διαμέσου κέντρων διανομής.

Σε κάθε περίπτωση μεταφοράς προϊόντων η επιχείρηση καλείται να πάρει αποφάσεις σχετικά με την οργάνωση του στόλου διανομής, με τη σχεδίαση των διαδρομών και τον προγραμματισμό των δρομολογίων που θα αναλάβουν τα οχήματα. Συγκεκριμένα, λαμβάνονται αποφάσεις σχετικά με:

- Τη σχεδίαση του δικτύου διανομής, με κατάλληλη επιλογή των αποθηκών και κέντρων διανομής.
- Ανάθεση μεταφορών σε ιδιωτικό στόλο, είτε αγορά ή ενοικίαση από τρίτους.
- Επιλογή τύπου και μεγέθους οχημάτων, καθώς και ανθρώπινου δυναμικού, που θα στελεχώσει το στόλο.
- Σχεδίαση των βέλτιστων διαδρομών και κατάλληλων χρονικά προγραμματισμένων δρομολογίων.
- Επιλογή έμπιστου και κατάλληλα καταρτισμένου ανθρώπινου δυναμικού.

Κεφάλαιο 2ο: Το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem)

Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται αναλυτικά το πρόβλημα δρομολόγησης οχημάτων, το μαθηματικό του μοντέλο και οι περιορισμοί που προκύπτουν, κατά την επίλυση του. Επίσης παρατίθενται ορισμένες παραλλαγές προβλημάτων δρομολόγησης.

2.1 Το πρόβλημα του Πλανόδιου Πωλητή TSP (The Travelling Salesman Problem)

Για την κατανόηση του προβλήματος δρομολόγησης οχημάτων πρέπει πρώτα να γίνει μια εισαγωγή στο πρόβλημα του πλανόδιου πωλητή. Πρόκειται, ίσως, για το πιο συζητημένο πρόβλημα συνδυαστικής βελτιστοποίησης και αποτελεί τη βάση για πολλά πιο σύνθετα προβλήματα που απασχολούν την επιστήμη γύρω από την εφοδιαστική αλυσίδα. Το πρόβλημα περιέχει τα εξής τρία χαρακτηριστικά: ένα πωλητή, ο οποίος αναζητά να πουλήσει τα αγαθά του, ένα σημείο αφετηρίας, από το οποίο ξεκινά ο πωλητής και γεωγραφικά διάσπαρτα σημεία, έστω πόλεις, τις οποίες δύναται να επισκεφτεί[4]. Το ζητούμενο του προβλήματος είναι να βρεθεί η καλύτερη διαδρομή, που θα επιλέξει ο πλανόδιος πωλητής κατά την οποία περνάει από όλες τις πόλεις, ελαχιστοποιώντας την απόσταση που διένυσε. Τέλος, πρέπει να σημειωθεί πως ο πωλητής ξεκινά και τερματίζει αυστηρά από το σημείο αφετηρίας και επιτρέπεται να επισκεφτεί την κάθε πόλη μόνο μια φορά.

2.2 Το πρόβλημα δρομολόγησης οχημάτων (VRP)

Ο τομέας των logistics μιας επιχείρησης καλείται να αντιμετωπίσει πολυάριθμα ζητήματα. Ένα από τα πιο μελετημένα προβλήματα, και αρκετά πολύπλοκο, είναι αυτό της δρομολόγησης οχημάτων (VRP). Πρόκειται για ένα πρόβλημα βελτιστοποίησης των διαδρομών ενός στόλου οχημάτων, τα οποία εκτελούν παραδώσεις/παραλαβές προς/από ένα σύνολο πελατών σε συγκεκριμένη χρονική περίοδο, υπακούοντας σε διάφορους περιορισμούς.

Οι πρώτοι ερευνητές που ασχολήθηκαν με την επίλυση του VRP ήταν οι Danzing και Ramser το 1959, οι οποίοι εφαρμόζοντας γραμμικό προγραμματισμό προσέγγισαν μια αρκετά ικανοποιητική λύση του προβλήματος σε θεωρητικό επίπεδο. Πρέπει να σημειωθεί πως τα προβλήματα δρομολόγησης οχημάτων υπάγονται στην κατηγορία NP-Hard προβλημάτων, δηλαδή προβλήματα που χαρακτηρίζονται από μεγάλη υπολογιστική πολυπλοκότητα, που τα καθιστά έως και αδύνατον να επιλυθούν από έναν ανθρώπινο εγκέφαλο[5]. Με λίγα λόγια, η δυσκολία επίλυσης αυξάνει εκθετικά ανάλογα με το μέγεθος του προβλήματος.

2.3 Στόχος του Προβλήματος Δρομολόγησης Οχημάτων

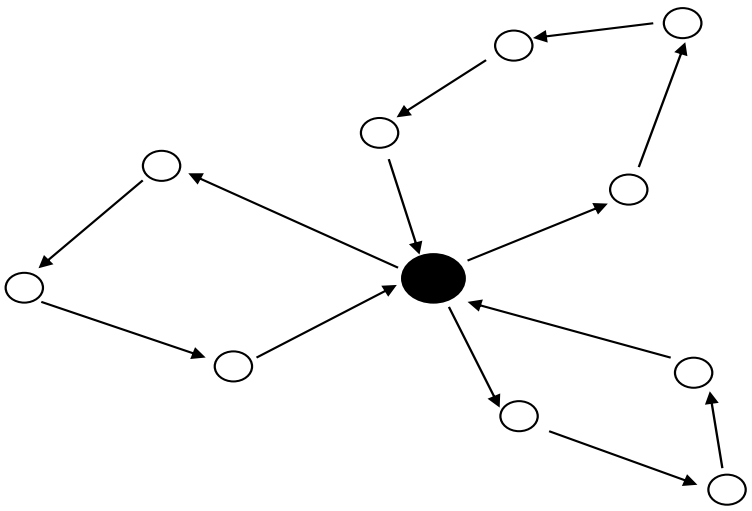
Σκοπός των VRPs είναι η κατάλληλη δρομολόγηση των οχημάτων που απαρτίζουν τον στόλο μεταφορών μιας επιχείρησης, ικανοποιώντας πλήρως τη ζήτηση των πελατών, ώστε να ελαχιστοποιηθεί το κόστος διαδρομής. Η επιχείρηση μέσα από την επίλυση του προβλήματος καλείται να εξυπηρετήσει άριστα τον εκάστοτε πελάτη, εξοικονομώντας όσο το δυνατόν περισσότερο κόστος προς όφελός της. Το κόστος στον τομέα των logistics μεταφράζεται συνήθως σε απαιτούμενο προσωπικό, σε ποσότητα καυσίμου που καταναλώθηκε και σε απόσβεση του οχήματος. Τέλος, όπως κάθε πρόβλημα, η επίλυση του VRP καλείται να υπακούει σε περιορισμούς που μπορεί να διαφέρουν από πρόβλημα σε πρόβλημα και ποικίλουν ανάλογα με τα χαρακτηριστικά των αγαθών, των οχημάτων, τη ζήτηση των πελατών και άλλα.

2.4 Χαρακτηριστικά του προβλήματος δρομολόγησης οχημάτων

2.4.1 Οδικό δίκτυο

Το σύνολο των διαδρομών που θα προκύψουν από την επίλυση του προβλήματος δρομολόγησης διαμορφώνονται πάνω σε ένα οδικό δίκτυο που αναπαρίσταται με γράφημα (Σχήμα 2.1). Στο γράφημα τα τόξα αντιπροσωπεύουν τις διαδρομές και οι κόμβοι τις αποθήκες και τους πελάτες. Κάθε τόξο περιέχει κόστος $c_{i,j,}$ το οποίο αντιστοιχεί στο μήκος της διαδρομής από τον κόμβο i μέχρι τον κόμβο j και, παράλληλα, στο χρόνο που χρειάζεται ένα όχημα για να τη διανύσει.

Στο σχήμα 2.1 απεικονίζεται ένα απλό παράδειγμα επίλυσης του προβλήματος δρομολόγησης οχημάτων με μία αποθήκη (μαύρος κύκλος στο κέντρο) και δέκα κόμβους εξυπηρέτησης. Οι διαδρομές που δημιουργούνται από τη λύση είναι τρεις.



Σχήμα 2.1 *Αναπαράσταση λύσης VRP*

Όλα τα προβλήματα της κατηγορίας VRP αποτελούνται από τρεις κύριες οντότητες: τις αποθήκες, τα οχήματα και τους πελάτες. Κάθε μια από τις τρεις οντότητες φέρει χαρακτηριστικά που δημιουργούν τα δεδομένα του προβλήματος.

2.4.2 Αποθήκες:

Οι αποθήκες αποτελούν σημείο αφετηρίας και τερματισμού για τα οχήματα και, σε ορισμένες περιπτώσεις προβλημάτων δρομολόγησης, τα οχήματα επιστρέφουν και αναπληρώνουν ή εναποθέτουν φορτίο πριν συνεχίσουν τη διαδρομή τους. Μερικά χαρακτηριστικά που διακρίνουν μια αποθήκη είναι:

- η γεωγραφική θέση στο γράφημα
- η χωρητικότητα της σε αγαθά
- ο αριθμός οχημάτων που μπορεί να φιλοξενήσει

2.4.3 Οχήματα:

Για κάθε όχημα είναι απαραίτητο να καταγράφονται πληροφορίες ώστε να προσδιοριστεί το κατάλληλο VRP. Για παράδειγμα:

- Ποια θα είναι η αποθήκη από την οποία θα ξεκινήσει τη διαδρομή και αν πρέπει να επιστρέψει στην ίδια ή διαφορετική (αν υπάρχουν πάνω από μία διαθέσιμες).
- Η χωρητικότητα του κάθε οχήματος, η οποία συνήθως συμβολίζει μέγιστο βάρος, όγκο ή αριθμό φορτίου, που μπορεί να μεταφέρει το όχημα, καθώς επίσης αν όλα τα οχήματα είναι ίσης χωρητικότητας, ώστε να κατανεμηθούν σε αντίστοιχες ομάδες.
- Σε διάφορες περιπτώσεις, το όχημα πέρα από παράδοση μπορεί να εκτελεί και παραλαβές.
- Αν τα οχήματα είναι χωρισμένα σε τμήματα, καθένα από τα οποία φορτώνεται διαφορετικά.

- Το σύνολο των δρόμων που είναι προσπελάσιμοι από τα οχήματα.
- Το κόστος που σχετίζεται με τη λειτουργία του οχήματος

Σύμφωνα με ευρωπαϊκή νομοθεσία, θεσπίστηκαν ορισμένοι κανονισμοί όσον αφορά τις εργατοώρες των οδηγών[7]. Περιληπτικά, ο κάθε οδηγός πρέπει:

- να οδηγεί λιγότερο από εννέα ώρες συνεχόμενα την ημέρα (μπορούν να αυξηθούν σε 10 αλλά όχι παραπάνω από δύο φορές τη βδομάδα),
- να μην κάνει οδήγηση περισσότερων από 56 ωρών την εβδομάδα (ή 6 μέρες τη βδομάδα),
- να ξεκουράζεται καθημερινά τουλάχιστον 11 ώρες, συμπεριλαμβανομένου και του χρόνου ύπνου.

2.4.4 Πελάτες:

Για να δομηθεί σωστά ένα πρόβλημα δρομολόγησης οχημάτων θα πρέπει να θεωρηθούν και τα χαρακτηριστικά των πελατών, τα οποία είναι τα εξής:

- Η γεωγραφική θέση, η οποία καταγράφεται με συντεταγμένες (x,y) στο γράφημα διανομής.
- Η ζήτησή του, δηλαδή η απαιτούμενη ποσότητα φορτίου που πρέπει να παραδώσει ή να λάβει το όχημα, προς ή από τον πελάτη, αντίστοιχα.
- Εάν θα πραγματοποιηθεί παραλαβή, παράδοση ή και τα δύο.
- Το επιτρεπόμενο χρονικό διάστημα που θα γίνει η εξυπηρέτησή του (time windows).
- Ο χρόνος που απαιτείται για την εξυπηρέτηση (uploading or loading time).

2.5 Κανόνες και περιορισμοί στο Πρόβλημα Δρομολόγησης Οχημάτων

Οι θεμελιώδεις κανόνες ενός προβλήματος δρομολόγησης οχημάτων είναι κοινοί για όλα τα προβλήματα βελτιστοποίησης αυτού του είδους. Κάθε όχημα πρέπει να ξεκινά και να επιστρέφει σε μια αποθήκη, κατά τη διάρκεια της βάρδιας του οδηγού. Επίσης, το όχημα επιβάλλεται να καλύψει πλήρως τη ζήτηση του πελάτη, στο πέρασμά του, κατά το δρομολόγιο και κάθε πελάτης εξυπηρετείται αυστηρά από ένα όχημα την πρώτη φορά στο πέρασμά του. Στη συνέχεια, βέλτιστη διαδρομή είναι αυτή με το χαμηλότερο κόστος, η οποία θα έχει διαμορφωθεί υπακούοντας στους περιορισμούς.

2.5.1 Μαθηματικό μοντέλο

Το πρώτο μαθηματικό μοντέλο του απλού προβλήματος δρομολόγησης οχημάτων παρουσιάστηκε από τους Fisher & Jaikumar (1981)[6].

Έστω ότι θεωρείται το δίκτυο G = (N,E), όπου N = {0,1,2,…,n}, το διάνυσμα που περιέχει τους κόμβους του προβλήματος, με την αποθήκη να λαμβάνει τον αριθμό δείκτη 0 και τους υπόλοιπους να αντιπροσωπεύουν τους πελάτες. Η δομή E = {(i, j)| i, j ∈ N, i = j}είναι ο πίνακας των αποστάσεων για κάθε ζεύγος κόμβων. Επίσης $c_{i,j}$ αντιπροσωπεύει την τιμή της ευκλείδειας απόστασης μεταξύ των κόμβων i, j . Τα οχήματα περιέχονται στο V = {1,2,…., m} και η χωρητικότητά τους είναι C_k = {k = 1,2,….m}. Στο απλό CVRP έχουμε για κάθε όχημα ίδια χωρητικότητα C. Έχουμε, επίσης:

$$x_{i,j,k} = \begin{cases} 1, & \textit{εαν το όχημα } k \textit{ επισκεφτεί τον πελάτη } j \textit{ αμέσως μετά τον } i \\ 0, & \textit{αλλιώς} \end{cases}$$

$$y_{j,k} = \begin{cases} 1, \textit{εάν ο πελάτης } i \textit{ εξυπηρετείται από το όχημα } k \\ 0, \textit{αλλιώς} \end{cases}$$

Αντικειμενική συνάρτηση:

$$\min \sum_{i,j} c_{i,j} \sum_{k} x_{i,j,k} \tag{2.4.1}$$

υπό

$$\sum_{k}^{K} y_{i,k} = \begin{cases} 1, i = 1, \dots, n \\ m, \quad i = 0 \end{cases} \tag{2.4.2}$$

$$\sum_{i} q_i y_{ij} \,, \leq \; C \qquad \forall k \in V \tag{2.4.3}$$

$$\sum_{j} x_{i,j,k} = \sum_{j} x_{j,i,k} = \; y_{i,k} \; \forall i \in N \; \& \; \forall k \in V \tag{2.4.4}$$

$$\sum_{i,j \in S} x_{i,j,k} \leq \; |S| - 1, \qquad \forall S \subseteq \; N \tag{2.4.5}$$

$$y_{i,k} \in \{0,1\} \quad \forall \; i,j \in N \forall k \in V \tag{2.4.6}$$

$$x_{i,j,k} \in \{0,1\} \quad \forall \; i,j \in N \forall k \in V \tag{2.4.7}$$

Η αντικειμενική συνάρτηση (2.4.1) παριστάνει την ελαχιστοποίηση του συνολικού κόστους. Ο περιορισμός (2.4.2) δείχνει ότι κάθε πελάτης μπορεί να εξυπηρετηθεί από ένα μόνο όχημα, ωστόσο την αποθήκη την επισκέπτονται όλα τα οχήματα. Το όριο χωρητικότητας καλύπτεται από τον περιορισμό (2.4.3) και η ισότητα (2.4.4) δείχνει ότι αν ένα όχημα επισκεφτεί ένα πελάτη φεύγει από αυτόν και συνεχίζει τη διαδρομή. Επίσης, ο περιορισμός (2.4.5) απαγορεύει διαδρομές που δεν περιέχουν την αποθήκη. Τέλος οι περιορισμοί (2.4.6) και (2.4.7) διασφαλίζουν ότι ένα όχημα μπορεί να φύγει ή να επιστέψει μόνο μια φορά στην αποθήκη.

Οι περιορισμοί σε ένα VRP πηγάζουν από τις νομοθετικές ρυθμίσεις, την πολιτική της επιχείρησης και τα χαρακτηριστικά των φορτίων και των οχημάτων. Μερικοί περιορισμοί που, επίσης, μπορεί να εμφανίζονται στα προβλήματα δρομολόγησης οχημάτων είναι οι εξής:

- Σχετικά με την φύση της εξυπηρέτησης, παραλαβή ή παράδοση.
- Χρονικά περιθώρια εξυπηρέτησης του κάθε πελάτη.
- Το ωράριο απασχόλησης των οδηγών.
- Η χωρητικότητα των οχημάτων.
- Η σειρά επίσκεψης των πελατών.
- Σχετικά με το αν θα πραγματοποιηθεί επαναφόρτωση του οχήματος στην αποθήκη

2.6 Παραδείγματα προβλημάτων δρομολόγησης οχημάτων

2.6.1 Το πρόβλημα δρομολόγησης οχημάτων με περιορισμό χωρητικότητας CVRP (Capacitated Vehicle Routing Problem)

Το CVRP που θα επιλυθεί στην παρούσα εργασία, αποτελεί μια επέκταση του VRP με την προσθήκη του περιορισμού του χρόνου. Διατίθεται ένας στόλος από οχήματα, τα οποία έχουν περιορισμένη χωρητικότητα και καλούνται να εξυπηρετήσουν ένα σύνολο πελατών, οι οποίοι έχουν συγκεκριμένη ζήτηση. Επιπλέον περιορισμό αποτελεί ο χρόνος, κατά τον οποίο το όχημα θα βρίσκεται στη διαδρομή. Για κάθε όχημα ορίζεται ένας μέγιστος χρόνος που μπορεί να παραμείνει σε μία διαδρομή και, παράλληλα, υπάρχει συγκεκριμένος χρόνος εξυπηρέτησης του κάθε πελάτη. Σε αυτό το πρόβλημα πρέπει να ληφθεί υπόψη και ο χρόνος μετάβασης από πελάτη σε πελάτη, ο οποίος, σε άθροισμα με τον χρόνο εξυπηρέτησης, ορίζει το χρόνο που βρίσκεται το όχημα μέσα στη διαδρομή. Επομένως, ξεκινώντας από την αποθήκη το κάθε όχημα οφείλει να επιστρέψει σε αυτήν σε δύο περιπτώσεις: όταν καλυφθεί το όριο της χωρητικότητάς του, ή όταν φτάσει στο μέγιστο χρονικό διάστημα, που μπορεί να βρίσκεται στη διαδρομή.

2.6.2 Το πρόβλημα δρομολόγησης οχημάτων με πολλαπλές επιστροφές στην αποθήκη

Σε αυτό το πρόβλημα, τα οχήματα έχουν τη δυνατότητα να πραγματοποιήσουν παραπάνω από μία διαδρομές, εφόσον δεν έχει ξεπεραστεί το χρονικό όριο παραμονής σε διαδρομή, αλλά έχει επέλθει το όριο του περιορισμού της χωρητικότητας. Τα οχήματα μπορούν να επιστρέψουν στην αποθήκη, να επαναφορτώσουν και να πραγματοποιήσουν μία νέα διαδρομή, εξυπηρετώντας άλλους πελάτες.

2.6.3 Το ανοιχτό πρόβλημα δρομολόγησης οχημάτων

Σε αυτό το πρόβλημα γίνεται η υπόθεση ότι τα οχήματα είναι ενοικιασμένα. Με άλλα λόγια, τα οχήματα έχουν κανονικά αφετηρία την αποθήκη αλλά, μόλις ολοκληρώσουν τη διαδρομή τους και καλύψουν όλη τη ζήτηση των πελατών, δεν επιστρέφουν στο χώρο αφετηρίας.

2.6.4 Το ανοιχτό κλειστό πρόβλημα δρομολόγησης οχημάτων

Αυτό το πρόβλημα αποτελεί μία παραλλαγή του προβλήματος δρομολόγησης οχημάτων όπου παρατηρούνται επιστροφές στην αποθήκη. Σε αυτή την περίπτωση, τα οχήματα που έχουν ολοκληρώσει μία διαδρομή, επιστρέφουν στην αποθήκη, εκτελούν επαναφόρτωση του εμπορεύματος και κάνουν μία νέα διαδρομή, ώσπου να φτάσει το όριο του χρονικού περιορισμού και να επιστρέψουν στην εταιρεία, η οποία τα νοικιάζει.

2.6.5 Το πρόβλημα δρομολόγησης οχημάτων για εξυπηρέτηση πελατών σε συγκεκριμένο χρονικό πλαίσιο

Στο πρόβλημα αυτό εμφανίζεται ο περιορισμός του χρονικού παραθύρου μέσα στο οποίο πρέπει να γίνει η εξυπηρέτηση του πελάτη. Κατά την επίλυση αυτού του προβλήματος ισχύει ο περιορισμός της χωρητικότητας του οχήματος, αλλά και η τήρηση των χρονικών πλαισίων, που δίνονται, για την εξυπηρέτηση του κάθε πελάτη.

2.6.6 Το πρόβλημα δρομολόγησης οχημάτων με δύο είδη πελατών

Σε αυτό το πρόβλημα, οι πελάτες χωρίζονται σε δύο ομάδες, σε αυτούς που δύνανται να παραλάβουν και σε αυτούς που επιθυμούν να παραδώσουν μία ποσότητα. Σε μία διαδρομή συνυπάρχουν και τα δύο είδη πελατών. Οι όροι του προβλήματος προϋποθέτουν ότι ο κάθε πελάτης πρέπει να ανήκει σε μία μόνο ομάδα, είτε να απαιτεί παραλαβή είτε διανομή, ενώ επίσης, η διανομή πραγματοποιείται πάντα πρώτη από το όχημα. Τέλος, απαγορεύεται η δημιουργία διαδρομών που να εξυπηρετεί μόνο παραλαβές.

2.6.7 Το πρόβλημα δρομολόγησης οχημάτων για διανομές και παραλαβές προϊόντων

Πρόκειται για πρόβλημα που μοιάζει με το πρόβλημα των δύο ειδών πελατών. Ωστόσο, στη συγκεκριμένη περίπτωση, όλοι οι πελάτες έχουν ταυτόχρονα ποσότητα να παραλάβουν και να παραδώσουν Το κάθε όχημα του στόλου, κατά το πέρασμά του από τον πελάτη εκτελεί ταυτόχρονα διανομή και παραλαβή προϊόντων. Είναι απαραίτητο να καλυφθεί πρώτα η διανομή των προϊόντων και μετά η παραλαβή τους.

2.6.8 Το πρόβλημα δρομολόγησης οχημάτων με πολλαπλές αποθήκες

Σε αυτό το πρόβλημα, η βασική διαφορά με τα προηγούμενα είναι ότι υπάρχουν παραπάνω από μία αποθήκες. Με αυτό τον τρόπο πραγματοποιείται επίλυση πολλαπλών προβλημάτων δρομολόγησης, όπου τα οχήματα ξεκινώντας από μία αποθήκη μπορούν να τερματίσουν σε μία διαφορετική και ενδιάμεσα στη διαδρομή τους να ανεφοδιαστούν από οποιαδήποτε κοντινότερη αποθήκη.

2.7 Μοντελοποίηση του CVRP

Το πρόβλημα δρομολόγησης οχημάτων με περιορισμό χωρητικότητας (CVRP), είναι το απλούστερο πρόβλημα αυτής της κατηγορίας. Οι περιορισμοί για την επίλυση και την σύνθεση των διαδρομών είναι η χωρητικότητα του κάθε οχήματος που θα χρησιμοποιηθεί καθώς και ο μέγιστος χρόνος παραμονής του οχήματος σε μια διαδρομή. Στόχος είναι η εύρεση των διαδρομών που θα ακολουθήσουν τα οχήματα, ώστε να ελαχιστοποιηθεί το κόστος μετάβασης και να καλυφθεί η ζήτηση των πελατών, υπακούοντας στους περιορισμούς.

2.7.1 Γενική περιγραφή

Δίνεται μια κύρια αποθήκη και ένας αριθμός κόμβων (πελάτες) προς εξυπηρέτηση. Αυτά τα στοιχεία είναι κατανεμημένα διάσπαρτα στο δισδιάστατο χώρο με συγκεκριμένες γεωγραφικές συντεταγμένες x, y. Για τους πελάτες, πέρα από τις συντεταγμένες, είναι γνωστή

και η ζήτηση q_i (i = 1,2, …, N), όπως και για τα οχήματα δίνεται η μέγιστη χωρητικότητα Q_k (k = 1,2 …, m). Κάθε όχημα ξεκινά από την κεντρική αποθήκη και επιστέφει σε αυτή και κατά την διαδρομή που εκτελεί δεν πρέπει να ξεπεράσει τον περιορισμό της χωρητικότητάς του. Επίσης, κάθε πελάτης εξυπηρετείται μόνο μια φορά ικανοποιώντας όλη την ποσότητα της ζήτησής του.

2.7.2 Μαθηματικό μοντέλο CVRP

Σε πρώτη φάση, θεωρείται το δίκτυο G = (N,E). Όπου N = $\{i_1,i_2,\ldots,i_n\}$ το διάνυσμα που περιέχει τους κόμβους του προβλήματος με i_1= 0 συνήθως να αναφέρεται στην κεντρική αποθήκη και για i_l, l∈{2,3,…,n} καταγράφονται οι πελάτες. Η δομή E = $\{(i_l, i_m)| i_l, j_m \in N\}$ είναι ο πίνακας των τόξων για κάθε ζεύγος κόμβων.
Κάθε πελάτης πρέπει να εξυπηρετηθεί αυστηρά από ένα όχημα k, του οποίου η χωρητικότητα δηλώνεται ως Q_k. Στο απλό CVRP ορίζεται ότι όλα τα οχήματα είναι ομοιογενή με ίδια χωρητικότητα. Η ζήτηση κάθε πελάτη i_l καταγράφεται με q_l και ο χρόνος εξυπηρέτησης με st_l. Επίσης η μεταβλητές $c_{l,m}$ και $tt_{l,m,k}$ αντιπροσωπεύουν την τιμή της ευκλείδειας απόστασης και το χρόνο μετάβασης μεταξύ των κόμβων i_l, i_m αντίστοιχα. Το ανώτερο χρονικό όριο που μπορεί ένα όχημα να βρίσκεται σε μια διαδρομή δηλώνεται με T_k. Τέλος, ορίζεται η παρακάτω δυαδική μεταβλητή ως:

$$x_{l,m,k} = \begin{cases} 1, & \text{αν το όχημα k περιλαμβάνει στη διαδρομή του το τόξο } (i_l, i_m) \\ 0, & \text{αλλιώς} \end{cases}$$

Αντικειμενική συνάρτηση:

$$\min \sum_{l,m=1}^{n} c_{l,m} \sum_{k=1}^{K} x_{l,m,k} \qquad (2.1)$$

υπό

$$\sum_{l=1}^{n}\sum_{k}^{K} x_{l,m,k} = 1 \qquad m = 1,2 \ldots, n \qquad (2.2)$$

$$\sum_{m=1}^{n}\sum_{k}^{K} x_{l,m,k} = 1 \qquad l = 1,2 \ldots, n \qquad (2.3)$$

$$\sum_{l}^{n} x_{l,f,k} - \sum_{m=1}^{n} x_{f,m,k} = 0 \quad k = 1 \ldots, K, f = 1, \ldots, n \qquad (2.4)$$

$$\sum_{l,m=1}^{n} q_j x_{l,f,k}, \leq Q_k \qquad k = 1, \ldots K \qquad (2.5)$$

$$\sum_{l=1}^{n}\sum_{m=1}^{n} st_l x_{l,m,k} + \sum_{l=1}^{n}\sum_{m=1}^{n} tt_{l,m,k} x_{l,m,k} \leq T_k \quad k = 1, \ldots K \qquad (2.6)$$

$$\sum_{m=2}^{n} x_{1,m,k} \leq 1, \qquad k = 1, \dots K \tag{2.7}$$

$$\sum_{l=2}^{n} x_{l,1,k} \leq 1, \qquad k = 1, \dots K \tag{2.8}$$

$$X \in S \tag{2.9}$$

Η αντικειμενική συνάρτηση (2.1) δείχνει την ελαχιστοποίηση του συνολικού κόστους, το οποίο προκύπτει από το άθροισμα του κόστους κάθε διαδρομής που πραγματοποιείται μεταξύ δύο διαδοχικών κόμβων i και j .

Ο περιορισμός (2.2) και (2.3) δείχνει ότι κάθε πελάτης μπορεί να εξυπηρετηθεί από ένα μόνο όχημα, ωστόσο την αποθήκη την επισκέπτονται όλα τα οχήματα. Ο περιορισμός (2.4) δείχνει ότι όταν ένα όχημα επισκέπτεται έναν πελάτη φεύγει από αυτόν. Ο επόμενος περιορισμός (2.5) αντιπροσωπεύει το άνω όριο χωρητικότητας των οχημάτων και ο (2.6) διασφαλίζει το χρονικό όριο που μπορεί να παραμείνει το όχημα στη διαδρομή. Τέλος οι περιορισμοί (2.7) και (2.8) διασφαλίζουν ότι ένα όχημα μπορεί να φύγει και να επιστρέψει μόνο μια φορά στην αποθήκη.

Κεφάλαιο 3 Αλγόριθμοι βελτιστοποίησης

Εισαγωγή

Το τρίτο κεφάλαιο της εργασίας περιλαμβάνει μια θεωρητική προσέγγιση στους αλγορίθμους, που έχουν σχεδιαστεί για την επίλυση προβλημάτων βελτιστοποίησης, όπως το πρόβλημα που εξετάζεται (VRP). Παρουσιάζονται οι κύριες κατηγορίες των αλγορίθμων και στη συνέχεια δίδονται μερικά παραδείγματα αλγορίθμων που έχουν επινοηθεί.

Καθώς η επίλυση των NP-Hard προβλημάτων βελτιστοποίησης ολοένα και γίνεται δυσκολότερη λόγω της πολυπλοκότητας τους, η επιστήμη οδηγήθηκε στην ανάπτυξη αλγορίθμων, οι οποίοι παράγουν καλές εφικτές λύσεις σε σύντομο υπολογιστικό χρόνο. Τα περισσότερα προβλήματα βελτιστοποίησης μπορεί να έχουν παραπάνω από μία λύση, δηλαδή να διαθέτουν πολλαπλά τοπικά ελάχιστα μέσα στα οποία βρίσκεται και η βέλτιστη λύση, το ολικό ελάχιστο. Οι αλγόριθμοι βελτιστοποίησης διακρίνονται σε δύο ευρύτερες κατηγορίες, τους ευρετικούς και τους μεθευρετικούς.

3.1 Ευρετικοί αλγόριθμοι (Heuristic Algorithms)

Οι ευρετικοί αλγόριθμοι δημιουργήθηκαν για να παράγουν καλές λύσεις σε σύντομο χρόνο, οι οποίες όμως δεν είναι απαραίτητα και οι βέλτιστες. Εκτελούν εξερεύνηση περιορισμένου χώρου αναζήτησης. Η παραγόμενη λύση γίνεται αποδεκτή όταν ικανοποιεί κάποια κριτήρια, όπως για παράδειγμα την ποιότητά της, δηλαδή αν η απόκλισή της από τη βέλτιστη είναι σχετικά μικρή. Οι ευρετικοί αλγόριθμοι χωρίζονται σε τρεις επιμέρους κατηγορίες: τους αλγόριθμους απληστίας (greedy algorithms), τους προσεγγιστικούς αλγόριθμους (approximation algorithms) και τους αλγόριθμους τοπικής αναζήτησης (local search algorithms).

3.1.1 Αλγόριθμοι απληστίας

Οι αλγόριθμοι απληστίας, κατά την προσπάθειά τους να φτάσουν σε μία εφικτή λύση, επιλέγουν την καλύτερη επόμενη ενέργεια, με βάση τα στοιχεία που είναι δεδομένα εκείνη τη στιγμή. Έχουν την ικανότητα να βρίσκουν μια καλή λύση, με απλό και γρήγορο τρόπο, αλλά το βασικό τους μειονέκτημα είναι ότι υπάρχει μεγάλη πιθανότητα να εγκλωβιστούν σε τοπικό ελάχιστο.

Οι αλγόριθμοι απληστίας, κατά την εφαρμογή τους, υπακούουν σε ορισμένες στρατηγικές οι οποίες κατατάσσονται στις παρακάτω ομάδες:

1. <u>Ομαδοποίηση πρώτα - δρομολόγηση έπειτα</u>:

Η συγκεκριμένη στρατηγική περιλαμβάνει την ομαδοποίηση των κόμβων με κριτήριο την απόστασή τους και, έπειτα, τη σχεδίαση των διαδρομών με το ελάχιστο δυνατό κόστος.

2. <u>Δρομολόγηση πρώτα - Ομαδοποίηση έπειτα:</u>

Πρακτικά είναι η αντίστροφη από την παραπάνω μέθοδο. Αυτή η στρατηγική περιλαμβάνει σε πρώτη φάση την δημιουργία μιας ενιαίας διαδρομής, η οποία σε δεύτερη φάση χωρίζεται σε μικρότερες, με όσο το δυνατόν χαμηλότερο κόστος.

3. Εξοικονομήσεις / καταχώρηση

Κατά την εφαρμογή αυτής της στρατηγικής συγκρίνονται σε κάθε επανάληψη δύο εναλλακτικές κινήσεις, οι οποίες μπορεί και να μην είναι εφικτές. Εν τέλει επιλέγεται η κίνηση στην οποία παρατηρείται η μεγαλύτερη εξοικονόμηση κόστους.

4. Βελτίωση ή ανταλλαγή

Η στρατηγική βελτίωση ή ανταλλαγή σχετίζεται με μεθόδους ανταλλαγής ακμών των λύσεων. Σε κάθε βήμα πραγματοποιείται η ανταλλαγή τόξων μέχρι την εύρεση μιας βελτιωμένης εφικτής λύσης.

5. Προσέγγιση μαθηματικού προγραμματισμού

Σε αυτή τη στρατηγική διαδικασία πραγματοποιείται η χρήση μαθηματικών μοντέλων, που είναι ικανά να επιλύσουν το εκάστοτε πρόβλημα δρομολόγησης οχημάτων

6. Αλληλοεπιδρών βελτιστοποίηση

Η στρατηγική αυτή προϋποθέτει την ενσωμάτωση και συνεργασία ενός έμπειρου αποφασίζοντα, ο οποίος σύμφωνα με την γνώση του καθορίζει και αναθεωρεί ο ίδιος κάποιες παραμέτρους και συμμετέχει ενεργά στην επίλυση.

7. Ακριβής διαδικασία

Οι ακριβείς διαδικασίες περιλαμβάνουν εξειδικευμένους αλγόριθμους, οι οποίοι διεξάγουν αναζήτηση σε ολόκληρο το χώρο λύσεων.

Μερικά παραδείγματα αλγορίθμων απληστίας είναι:

- Ο αλγόριθμος του πλησιέστερου γείτονα (Nearest Neighbor)
- αλγόριθμος Christofides
- Ο αλγόριθμος των εξοικονομήσεων Clarke and Write
- Ο αλγόριθμος των δύο φάσεων Fisher & Jaikumar
- Ο αλγόριθμος της διαδικασίας εισαγωγής κόμβων (Nearest Insertion)

3.1.2 Προσεγγιστικοί αλγόριθμοι

Οι προσεγγιστικοί αλγόριθμοι εγγυώνται την εύρεση μία προσεγγιστικά βέλτιστης λύσης, χρησιμοποιώντας την επιπλέον πληροφορία της ολικά βέλτιστης, ωστόσο μπορεί να μην είναι τόσο αποδοτικοί.

3.1.3 Αλγόριθμοι τοπικής αναζήτησης

Οι συγκεκριμένοι αλγόριθμοι υπακούουν σε μία πολύ απλή μέθοδο, ξεκινώντας δηλαδή από μία αρχική λύση προσπαθούν να τη βελτιώσουν εξερευνώντας τις γειτονικές της λύσεις. Πρόκειται για μια από τις παλαιότερες τεχνικές επίλυσης προβλημάτων, της δοκιμής και σφάλματος. Εφόσον μία γειτονική λύση αποδειχθεί χαμηλότερου κόστους, δηλαδή καλύτερη από την αρχική, την αντικαθιστά και η διαδικασία επαναλαμβάνεται μέχρι να φτάσει σε ένα κριτήριο τερματισμού. Το κριτήριο τερματισμού μπορεί να είναι είτε ένας μέγιστος αριθμός επαναλήψεων είτε έως ότου η λύση να μην βελτιώνεται άλλο, μετά από έναν αριθμό επαναλήψεων. Ορισμένοι αλγόριθμοι τοπικής αναζήτησης είναι οι εξής:

- 2-opt
- 3-opt
- 1-1 exchange
- 2-2 exchange
- 0-1 relocate
- 0-2 relocate

3.2 Μεθευρετικοί αλγόριθμοι (Metaheuristic algorithms)

Όπως προαναφέρθηκε, οι ευρετικοί αλγόριθμοι χαρακτηρίζονται από το μειονέκτημα του ότι υπάρχει μεγάλη πιθανότητα να παγιδευτούν σε ένα τοπικό ελάχιστο. Αυτό το μειονέκτημα των ευρετικών αλγορίθμων, σκοπεύουν να διορθώσουν οι μεθευρετικοί αλγόριθμοι. Κύρια διαφορά τους από τους ευρετικούς αλγόριθμους είναι ότι συνδυάζουν την τοπική αναζήτηση με άλλες στρατηγικές, έτσι ώστε να δημιουργήσουν μία διαδικασία που είναι ικανή να ξεφεύγει από τοπικά ελάχιστα. Οι περισσότεροι αλγόριθμοι αυτού του τύπου είναι στοχαστικοί, δηλαδή χρησιμοποιούν τυχαίες αναζητήσεις για να φτάσουν σε μία βέλτιστη λύση και για το λόγο αυτό δεν είναι σίγουρο ότι η λύση που θα βρεθεί θα είναι πάντα η ίδια ή θα είναι η βέλτιστη. Ωστόσο, λόγω της τυχαιότητας που τους χαρακτηρίζει, αυξάνεται η πιθανότητα θετικού αποτελέσματος. Χρησιμοποιούν είτε μία αρχική λύση, την οποία στοχεύουν να βελτιώσουν, είτε διαθέτουν έναν πληθυσμό από αρχικές λύσεις και προσπαθούν να κάνουν αναζήτηση σε όλο αυτό το χώρο των λύσεων.
Μερικοί από τους πιο γνωστούς μεθευρετικούς αλγόριθμους είναι οι εξής:

- ο αλγόριθμος περιορισμένης αναζήτησης (Tabu Search)
- γενετικός αλγόριθμος (Genetic Algorithm)
- τα νευρωνικά δίκτυα (Neutral Networks)
- ο αλγόριθμος προσομοιωμένης ανόπτησης (Simulated Annealing)

3.3 Εξελικτικοί αλγόριθμοι

Οι εξελικτικοί αλγόριθμοι αποτελούν μία υποκατηγορία των μεθευρετικών αλγορίθμων, το κύριο χαρακτηριστικό των οποίων είναι το ότι βασίζονται σε εξελικτικές μεθόδους που παρατηρούνται στη φύση. Δηλαδή υπακούουν στο φυσικό φαινόμενο της επιβίωσης του δυνατότερου. Μέσα από μια διαδικασία που επαναλαμβάνεται για πολλές γενιές, οι νέες γενιές χρωμοσωμάτων (λύσεων) που δημιουργούνται διαθέτουν ισχυρότερα χαρακτηριστικά που προέρχονται μέσω της ανταλλαγής πληροφοριών γονιδίων. Ένας ισχυρός και διαδεδομένος αλγόριθμος βελτιστοποίησης που ανήκει σε αυτή την κατηγορία είναι ο γενετικός αλγόριθμος GA (Genetic Algorithm).

3.4 Αλγόριθμοι εμπνευσμένοι από τη φύση (Nature inspired algorithms)

Μια επιπλέον υποκατηγορία των μεθευρετικών αλγορίθμων προκύπτει από εκείνους που εμπνέονται από φαινόμενα της φύσης. Πιο συγκεκριμένα, οι ερευνητές παρατηρώντας τις συμπεριφορές των ζωντανών οργανισμών, προσπάθησαν να τις αποδομήσουν και να τις προσαρμόσουν μέσω πληροφοριακών συστημάτων, με σκοπό την επίλυση διάφορων προβλημάτων βελτιστοποίησης. Ιδιαίτερο ενδιαφέρον αποδόθηκε σε κοινωνικές συμπεριφορές των οργανισμών, που συγκεντρώνονται και επιβιώνουν μέσα σε κοπάδια ή σμήνη (swarm intelligence). Έτσι εμπνεύστηκαν αρκετοί αλγόριθμοι όπως :

- Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων PSO (Particle Swarm Optimization) (Kennedy, Eberhart and Shi, 1995)
- Ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών ACO (Ant Colony Optimization) (Dorigo and Stutzle, 2004)
- Ο αλγόριθμος βελτιστοποίησης ζευγαρώματος μελισσών HBMO (Honey Bees Mating Optimization)
- Ο αλγόριθμος της νυχτερίδας BA (Bat Algorithm) (Yang, 2010)

3.5 Ο Αλγόριθμος της Νυχτερίδας (Bat Algorithm)

3.5.1 Γενική περιγραφή

Ο Bat Algorithm (BA)[8] ή στα ελληνικά Αλγόριθμος της Νυχτερίδας, είναι ένας μεθευρετικός αλγόριθμος βελτιστοποίησης εμπνευσμένος από τη φύση. Ο αρχικός BA επινοήθηκε από τον Xin-She Yang το 2010 και έχει εφαρμοστεί επιτυχώς στην επίλυση πολλών προβλημάτων βελτιστοποίησης. Ο Yang εμπνεύστηκε από την ικανότητα ηχοεντοπισμού (echolocation) της νυχτερίδας, η οποία αποτελεί μέθοδο προσανατολισμού και εύρεσης τροφής.

Στη φύση υπάρχουν δύο κατηγορίες νυχτερίδων ανάλογα με το μέγεθος, οι μικρές νυχτερίδες (microbats) και οι γιγάντιες νυχτερίδες (megabats). Η ικανότητα ηχοεντοπισμού εντοπίζεται στις μικρές νυχτερίδες. Αυτές οι νυχτερίδες εκπέμπουν υψηλής έντασης ηχητικά κύματα, τα οποία μέσω της ανάκλασής τους σε περιβάλλοντα αντικείμενα προσδίδουν την ικανότητα στα ιπτάμενα θηλαστικά να κατευθύνονται αποφεύγοντας τα εμπόδια και να βρίσκουν το θήραμά τους μέσα στο σκοτάδι. Η συχνότητα του ηχητικού κύματος που εκπέμπει μια νυχτερίδα είναι σταθερή και λαμβάνει τιμές συνήθως από 25 kHz έως 150 kHz, ωστόσο μερικά είδη μπορούν να εκπέμπουν κύματα με συχνότητες άνω των 150 kHz. Όταν οι

νυχτερίδες βρίσκονται σε κατάσταση αναζήτησης τροφής εκπέμπουν παλμούς ιδιαίτερα υψηλής έντασης, ώστε να θεωρηθούν υπέρηχοι. Η ένταση ελαττώνεται όταν πλησιάζουν σε κάποιο θήραμα, όπως έντομα. Το αντίθετο συμβαίνει με τον ρυθμό των ηχητικών παλμών, οι οποίοι μπορούν να φτάσουν έως και 200 το δευτερόλεπτο. Τέλος, έχει αποτελέσει μεγάλο ερευνητικό ενδιαφέρον ο τρόπος με τον οποίο οι νυχτερίδες επεξεργάζονται σε χιλιοστά του δευτερολέπτου τα ανακλώμενα ηχητικά κύματα, χαρτογραφούν σε τρεις διαστάσεις τον χώρο που τις περιβάλλει και γνωρίζουν την ακριβή απόσταση από το θήραμά τους σε σκοτεινό περιβάλλον.

3.5.2 Οι εφαρμογές του Bat Algorithm

Ο ΒΑ αποδείχθηκε πως μπορεί να αποδώσει κατάλληλα σε πολλά προβλήματα βελτιστοποίησης. Η ικανότητά του να διατηρεί μία ισορροπία ανάμεσα στην ανίχνευση και την αναζήτηση (exploration and exploitation) τον καθιστά έναν αλγόριθμο που έχει την άνεση να προσεγγίζει τη βέλτιστη λύση σε σύντομο χρόνο. Η αποδοτικότητά του οδήγησε πολλούς ερευνητές να επιδιώξουν να τον χρησιμοποιήσουν σε διάφορες εφαρμογές βελτιστοποίησης, ταξινόμησης, επεξεργασίας εικόνας, data mining και πολλές άλλες[20]. Ωστόσο, ορισμένα από τα χαρακτηριστικά του δεν επαρκούν για την επίλυση όλων των κατηγοριών των προβλημάτων, πόσο μάλλον για προβλήματα που διαθέτουν πολλαπλά τοπικά ελάχιστα και διακριτές μεταβλητές. Για αυτό, ερευνητές εργάστηκαν πάνω στη μετατροπή του σε διάφορες παραλλαγές.

Οι παραλλαγές του ΒΑ εφαρμόστηκαν στην επίλυση ποικίλων προβλημάτων τεχνολογίας. Στον τομέα της συνεχούς βελτιστοποίησης, ο αλγόριθμος Bat έχει εφαρμοστεί με επιτυχία σε προβλήματα μηχανολογικού σχεδιασμού, όπως σχεδίαση εξοπλισμού αυτοκινήτων, για παράδειγμα ηλεκτροκινητήρα DC από τον Bora το 2012. Ο αλγόριθμος έχει επίσης χρησιμοποιηθεί για την αντιμετώπιση προβλημάτων συνδυαστικής βελτιστοποίησης, όπως προβλήματα χρονοπρογραμματισμού NP-Hard με πολλαπλές μηχανές παραγωγής και πολλαπλά προϊόντα (Musikapun and Pongvharoen, 2012). Στον τομέα της ταξινόμησης, ο ΒΑ έχει εφαρμοστεί με επιτυχία για την συσταδοποίηση δεδομένων (clustering), ενώ στον τομέα της επεξεργασίας εικόνας έχει χρησιμοποιηθεί για την εκτίμηση της στάσης του ανθρώπινου σώματος, όπως πραγματοποίησε ο Abdel-Rahman το 2012.

3.5.3 Μεθοδολογία του πρωτότυπου ΒΑ

<u>Ηχοεντοπισμός (echolocation)</u>

Η ικανότητα ηχοεντοπισμού της νυχτερίδας είναι το χαρακτηριστικό γύρω από το οποίο δομήθηκε ο αρχικός Bat Algorithm. Σύμφωνα λοιπόν με αυτή τη μοναδική ιδιότητα των ιπτάμενων θηλαστικών, ο Yang παρέθεσε τρεις θεμελιώδεις κανόνες και για τις τεχνητές νυχτερίδες.

α) Κάθε νυχτερίδα χρησιμοποιεί ηχοεντοπισμό για να προσδιορίσει την απόσταση και να ξεχωρίσει την τροφή από τα εμπόδια, ακόμα και μέσα στο σκοτάδι.

β) Οι νυχτερίδες πετάνε τυχαία αναζητώντας θήραμα. Καθώς κάνουν τις τυχαίες πτήσεις παραδεχόμαστε ότι έχουν τυχαία αρχική θέση, τυχαία αρχική ταχύτητα, μια αρχική σταθερή συχνότητα (fmin), ένα αρχικό ρυθμό παλμών (r0) και μια αρχική ένταση (A0). Ωστόσο, η ένταση και ο ρυθμός παλμών μεταβάλλονται καθώς πλησιάζουν στο θήραμα.

γ) Η ένταση του ηχητικού κύματος ποικίλει με πολλούς τρόπους. Υποθέτουμε πως κυμαίνεται από μια μέγιστη θετική τιμή (A_0) μέχρι μια ελάχιστη θετική τιμή (A_{min}).

Κίνηση των Νυχτερίδων

Τυχαία αναζήτηση (global search)

Αρχικά οι τεχνητές νυχτερίδες, όπως και οι πραγματικές, βρίσκονται τυχαία κατανεμημένες στο χώρο και οι θέσεις τους αντιπροσωπεύουν μια λύση. Για να ενημερώσουν την θέση τους x_i, αρχικά προσαρμόζουν τη συχνότητα (f_i) και έπειτα την ταχύτητα (u_i). Οι ενημερωμένες θέσεις, για κάθε νυχτερίδα i του σμήνους, υπολογίζονται σύμφωνα με τις παρακάτω εξισώσεις:

$$f_i = f_{min} + (f_{max} - f_{min}) * \beta \quad (3.1)$$

$$u_i^{t+1} = u_i^t + \left(x_i^t - x_*\right) * f_i \quad (3.2)$$

$$x_i^{t+1} = u_i^{t+1} + x_i^t \quad (3.3)$$

Όπου $\beta \in [0,1]$ τυχαίος αριθμός μέσα από την ομοιόμορφη κατανομή, f_i η τρέχουσα συχνότητα, f_{min} και f_{max} η ελάχιστη και η μέγιστη συχνότητα και x_* η προσωρινά βέλτιστη λύση σε όλο το σμήνος.

Τοπική αναζήτηση (local search)

Αφού οι νυχτερίδες έχουν ενημερώσει την θέση τους πετώντας τυχαία στο χώρο, επακολουθεί η φάση της τοπικής αναζήτησης. Σε αυτή τη φάση επιλέγεται μια υποψήφια λύση ανάμεσα στις καλύτερες και δημιουργείται μια καινούρια γειτονική σε αυτή λύση.

$$x_{new} = x_{old} + \varepsilon A^t \quad (3.4)$$

Όπου $\varepsilon \in [-1,1]$ τυχαίος αριθμός και A^t η μέση ένταση του ήχου για όλες τις νυχτερίδες του σμήνους τη χρονική στιγμή t.

Ένταση και ρυθμός εκπομπής παλμών

Με το πέρασμα κάθε χρονικής στιγμής t, δηλαδή σε κάθε επανάληψη του αλγορίθμου ενημερώνονται η ένταση του ήχου και ο ρυθμός εκπομπής των παλμών. Όταν μια νυχτερίδα πλησιάζει το στόχο, η ένταση σταδιακά μειώνεται και ο ρυθμός αυξάνεται. Η παραπάνω διαδικασία υπακούει στις ακόλουθες εξισώσεις:

$$A_i^{t+1} = A_i^t \alpha \quad (3.5)$$

$$r_i^{t+1} = r_i^0 [1 - \exp(\gamma t)] \quad (3.6)$$

Όπου A_i^t και A_i^{t+1} η προηγούμενη και η ενημερωμένη τιμή της έντασης αντίστοιχα, r_i^{t+1} η νέα τιμή του ρυθμού εκπομπής στην χρονική στιγμή t+1. Επίσης, οι α και γ είναι σταθερές για τις οποίες ισχύει $0 < \alpha < 1$ και $\gamma > 0$. Τέλος, καθώς $t \to \infty$: $A_i^t \to 0$ και $r_i^t \to r_i^0$

Παρακάτω παρουσιάζεται ο ψευδοκώδικας του αρχικού Bat Algorithm, όπως δημιουργήθηκε από τον Xin-She Yang:

Αλγόριθμος της νυχτερίδας (BA)

Ορισμός αντικειμενικής συνάρτησης f(x), $x = (x_1,\ldots,x_d)^T$
Αρχικοποίησή του πληθυσμού των νυχτερίδων και των παραμέτρων θέσης x_i και ταχύτητας u_i
Αρχικοποίηση της συχνότητας f_i, έντασης A_i και ρυθμού παλμών r_i για κάθε νυχτερίδα.
Επανάλαβε (για t < Μέγιστος_αριθμός_επαναλήψεων)
Επανάλαβε (για κάθε νυχτερίδα x_i στον πληθυσμό)
Δημιουργία νέων λύσεων ενημερώνοντας τη συχνότητα και υπολογίζοντας νέα ταχύτητα και θέση με τη χρήση των εξισώσεων (3.1),(3.2) και (3.3)
Εάν (τυχαίος_αριθμός > r_i)
Επιλογή μια εκ των καλύτερων λύσεων
Δημιουργία μιας γειτονικής λύσης γύρω από την επιλεγμένη μέσω τοπικής αναζήτησης με τη μέθοδο (3.4)
Τέλος εάν
Εάν (τυχαίος_αριθμός < A_i) και $f(x_i) < f(x_*)$
Αποδοχή της καινούριας λύσης
Ενημέρωση των A_i και r_i σύμφωνα με τις εξισώσεις (3.5) και (3.6)
Τέλος εάν
Τέλος
Κατάταξη των λύσεων και εύρεση της καλύτερης x_*
Τέλος

Αλγόριθμος 3.1 *Ο αρχικός αλγόριθμος νυχτερίδας (BA)*

Κεφάλαιο 4 - Υβριδικός Αλγόριθμος Νυχτερίδας (Hybrid Bat Algorithm)

Εισαγωγή

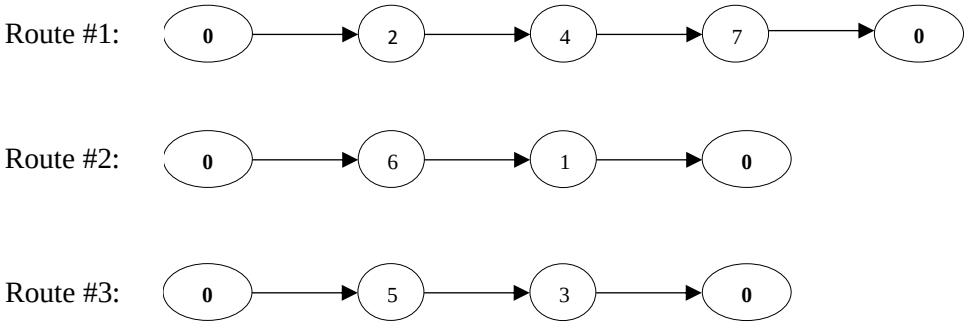
Στο παρόν κεφάλαιο παρουσιάζεται ο προτεινόμενος Υβριδικός Αλγόριθμος Νυχτερίδας (Hybrid Bat Algorithm). Ο HBA μιμείται την ιδιότητα των ζωντανών οργανισμών που ομαδοποιούνται σε σμήνη και αναζητούν τροφή ή φωλιά, ακολουθώντας συνήθως την πορεία των πιο έμπειρων οργανισμών. Πρόκειται για μια παραλλαγή του αρχικού BA και μετατροπή του από το συνεχές φάσμα λύσεων στο διακριτό. Για την κατασκευή του αρχικού πληθυσμού, χρησιμοποιήθηκε ο αλγόριθμος GRASP με τοπική αναζήτηση 3-opt. Η κίνηση των νυχτερίδων επιτεύχθηκε με τη χρήση μιας μεθόδου εμπνευσμένης από τον αλγόριθμο Path Relinking και την προσαρμογή της ταχύτητας σύμφωνα με την απόσταση hamming μεταξύ των λύσεων. Η συγκεκριμένη μέθοδος δίνει τη δυνατότητα στις νυχτερίδες να προσανατολίζονται στον χώρο των λύσεων με τυχαιότητα, αξιοποιώντας, παράλληλα, πληροφορία από τις καλύτερες λύσεις. Η προηγούμενη μέθοδος συμπληρώνεται με τακτικές τοπικής αναζήτησης με τη χρήση ενός ισχυρού αλγόριθμου, του VNS και με αυτόν τον τρόπο, η κάθε νυχτερίδα βελτιώνει την ατομική της θέση σε σύντομο χρόνο. Ο συνδυασμός κίνησης σμήνους και ατομικής τοπικής αναζήτησης που εφαρμόζονται, καθιστούν τον αλγόριθμο αρκετά αποδοτικό. Τέλος, ο HBA σεβόμενος τις εξισώσεις ηχοεντοπισμού του αρχικού BA, ως προς την ενημέρωση του παλμικού ρυθμού και της έντασης, καταφέρνει να οδηγήσει το σμήνος σε ολικό ελάχιστο.

4.1 Αναπαράσταση της λύσης

Ο αρχικός Bat Algorithm, επινοήθηκε και εφαρμόστηκε για την επίλυση προβλημάτων βελτιστοποίησης συνεχών μεταβλητών. Ωστόσο, για την επίλυση του προβλήματος δρομολόγησης, απαιτείται τροποποίησή του σε μια υβριδοποιημένη μορφή. Με άλλα λόγια, οι λύσεις του προβλήματος πρέπει να έχουν την κατάλληλη δομή, όπως φαίνεται στο παράδειγμα 4.1. Μια υποψήφια λύση αποτελείται από διαδρομές (routes), όπου η κάθε διαδρομή είναι μια λίστα αριθμών με αρχή και τέλος το 0, δηλαδή την αποθήκη (depot). Οι αριθμοί αντιπροσωπεύουν τους πελάτες και κατανέμονται στη διαδρομή με σειρά εξυπηρέτησης. Το παρακάτω παράδειγμα 4.1 παρουσιάζει ένα παράδειγμα λύσης με τρεις διαδρομές:

Αναπαράσταση της λύσης : [[0,2,4,7,0],[0,6,1,0],[0,5,3,0]]

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



Παράδειγμα 4.1 *Αναπαράσταση λύσης σε διακριτή μορφή με διαδρομές*

Για τη διευκόλυνση των υπολογιστικών διαδικασιών του κώδικα που θα παρουσιαστεί παρακάτω, η λύση λαμβάνει και μια πιο απλή διακριτή μορφή σε τύπο διανύσματος όπως:

x = [2,4,7,6,1,5,3]

ή

2	4	7	6	1	5	3

Παράδειγμα 4.2 *Διακριτή μορφή σε διάνυσμα*

4.2 Υπολογισμός αξίας μιας λύσης.

Το κόστος του CVRP ταυτίζεται με τη συνολική απόσταση που θα διανύσουν τα οχήματα, κατά τη διέλευση από τα γεωγραφικά σημεία όπου βρίσκονται πελάτες. Πρόκειται για το άθροισμα των αποστάσεων μεταξύ των διαδοχικών κόμβων – πελατών που βρίσκονται στις διαδρομές. Μια λύση με το χαμηλότερο κόστος αποτελεί τη βέλτιστη λύση του προβλήματος.

Αρχικά δημιουργείται ο πίνακας αποστάσεων (distance_matrix), ο οποίος είναι ένας συμμετρικός τετραγωνικός πίνακας. Κάθε κελί του πίνακα ($c_{i,j}$) λαμβάνει τιμή σύμφωνα με:

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (4.1)$$

Όπου i, j δύο διαδοχικοί πελάτες στη διαδρομή, συμπεριλαμβανομένης και της αποθήκης, με x, y συντεταγμένες στο δισδιάστατο επίπεδο. Στην περίπτωση που i = j στον πίνακα τοποθετείται η τιμή απείρου (∞) για προγραμματιστική διευκόλυνση.

Εν συνεχεία, υπολογίζεται το άθροισμα των τιμών $c_{i,j}$ μέσα σε κάθε διαδρομή και το συνολικό άθροισμα αποτελεί το κόστος της λύσης.

4.3 Δημιουργία αρχικού πληθυσμού

Ο αρχικός πληθυσμός λύσεων δημιουργείται με μια τυχαιοποιημένη διαδικασία. Σε αυτό το κομμάτι, επιδιώκεται η κατασκευή εφικτών υποψήφιων λύσεων. Κάθε νυχτερίδα, με τη θέση της (x_i) αντιπροσωπεύει μια εφικτή λύση. Η διαδικασία αυτή πραγματοποιείται μια

φορά στην αρχή του ΒΑ, κατά την αρχικοποίηση του πληθυσμού. Η μέθοδος που χρησιμοποιείται είναι η μέθοδος GRASP με τοπική αναζήτηση 3-opt.

4.3.1 Μέθοδος GRASP

Ο αλγόριθμος GRASP (Greedy Randomized Adaptive Search) ή στα Ελληνικά διαδικασία άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης, είναι ένας ευρετικός επαναληπτικός αλγόριθμος αναζήτησης. Αποτελείται από δύο φάσεις. Η πρώτη φάση αποτελεί την φάση κατασκευής (construction phase) μιας αρχικής εφικτής λύσης και στη δεύτερη φάση (local search phase) διεξάγεται τοπική αναζήτηση για τη βελτίωση της αρχικής αυτής λύσης. Οι δύο φάσεις επαναλαμβάνονται πολλές φορές και στο τέλος επιλέγεται η καλύτερη λύση που προέκυψε μέσω της επαναληπτικής διαδικασίας.

Μέθοδος GRASP στον υβριδικό ΒΑ

Στον προτεινόμενο αλγόριθμο χρησιμοποιείται ο GRASP[9] για την κατασκευή της αρχικής θέσης (x_i) της κάθε νυχτερίδας, με σκοπό τον καθορισμό του αρχικού πληθυσμού. Η δημιουργία του αρχικού πληθυσμού με τον GRASP γίνεται για να αποφευχθεί η εμφάνιση υποψήφιων λύσεων με πολύ υψηλό κόστος, οι οποίες είναι σχεδόν αδύνατον να βελτιωθούν. Η λύση κατασκευάζεται με την προσθήκη ενός στοιχείου τη φορά, το οποίο επιλέγεται τυχαία από μια λίστα RCL (restricted candidate list). Αρχικά επιλέγεται μια συνάρτηση απληστίας (greedy function), σύμφωνα με την οποία κατατάσσονται τα στοιχεία στην RCL. Η RCL έχει προκαθορισμένο σταθερό μέγεθος D και περιέχει τα D στοιχεία με την καλύτερη τιμή στη συνάρτηση απληστίας. Σαν πρώτο βήμα, κατασκευάζεται μια ενιαία διαδρομή με όλους τους πελάτες, η οποία επιλύει το πρόβλημα του πλανόδιου πωλητή (TSP). Έπειτα ξεκινά η δεύτερη φάση του αλγόριθμου GRASP, όπου εφαρμόζεται 3-opt αλγόριθμος τοπικής αναζήτησης για τη βελτιστοποίηση του μονοπατιού του TSP. Στη συνέχεια, η ενιαία διαδρομή χωρίζεται σε επιμέρους διαδρομές, που υπακούουν στους περιορισμούς του προβλήματος δρομολόγησης οχημάτων. Το πρώτο στοιχείο στη διαδρομή πρέπει να είναι ο κόμβος «0», δηλαδή θεωρείται ότι το πρώτο όχημα έχει ως αφετηρία την αποθήκη. Έπειτα, με την σειρά που εμφανίζεται στην ενιαία διαδρομή-λύση του TSP, επιλέγεται εάν ο επόμενος πελάτης μπορεί να εξυπηρετηθεί από το όχημα με βάση τους περιορισμούς, αλλιώς το όχημα επιστρέφει στην αποθήκη και ξεκινά ένα νέο όχημα από την αφετηρία προς μια καινούρια διαδρομή.

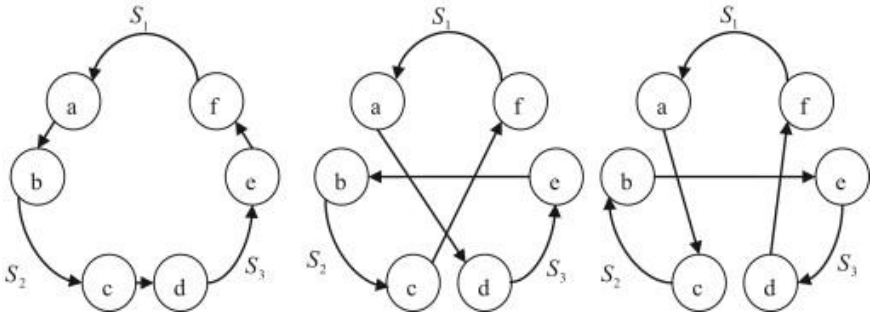
Η συνάρτηση απληστίας, στον αλγόριθμο που μελετάται, είναι η ευκλείδεια απόσταση των υποψήφιων πελατών από τον προηγούμενο πελάτη, που προστέθηκε στη διαδρομή σε κάθε επανάληψη, όπως φαίνεται στην εξίσωση 4.1. Τα D υποψήφια στοιχεία που κατατάσσονται στην RCL είναι οι πελάτες που δεν έχουν εξυπηρετηθεί και έχουν τη μικρότερη ευκλείδεια απόσταση από τον προηγούμενο.

4.3.2 Η μέθοδος αναζήτησης 3-opt

Ο αλγόριθμος 3-opt εφαρμόζεται, όπως αναφέρθηκε παραπάνω, στη λύση του TSP που προέκυψε με την κατασκευαστική μέθοδο του GRASP. Πρόκειται για μια διαδικασία αφαίρεσης τριών τόξων και της επανασύνδεσής του με κάθε πιθανό διαφορετικό τρόπο, μέχρι την εύρεση καλύτερου κόστους διαδρομής. Η μέθοδος 3-opt προϋποθέτει την ύπαρξη τουλάχιστον 8 πελατών σε μία διαδρομή για να εφαρμοστεί, για αυτό η εφαρμογή της σε μια

λύση του VRP δεν είναι τόσο αποδοτική. Κατά την εφαρμογή του 3-opt στον υβριδικό αλγόριθμο νυχτερίδας (HBA), ξεκινά μια επαναληπτική διαδικασία με μέγιστο αριθμό επαναλήψεων. Ξεκινώντας από τα πρώτα τρία τόξα, ελέγχεται η επανασύνδεση τους με διαφορετικούς τρόπους και η διαδικασία συνεχίζεται έως ότου προκύψει συντομότερη διαδρομή – λύση του TSP. Η λύση που βρέθηκε αντικαθιστά την προηγούμενη και η αναζήτηση ξεκινά από την αρχή μέχρι να ικανοποιηθεί το κριτήριο τερματισμού του ορίου επαναλήψεων. Ο μέγιστος αριθμός επαναλήψεων τέθηκε σε Imax=80.

Παράδειγμα 3-opt:



Εικόνα 4.3 *Μέθοδος 3-opt [23]*

Η εικόνα 4.3 αναπαριστά την διαγραφή των τόξων (a,b) , (c,f) και (c,d) και την επανασύνδεσή τους με δύο εναλλακτικούς τρόπους.

Ο αλγόριθμος κατασκευής αρχικών λύσεων του πληθυσμού περιγράφεται στους παρακάτω ψευδοκώδικες.

```
Procedure construct_TSP_solution()
## construction of the TSP solution path
TSP_path ← {depot}
not_yet_visited ← customer_list
while not_yet_visited ≠ Ø:
        c ← last insertion in path
        RCL ←create_rcl (not_yet_visited, c) # construct the RCL
        cj ← randomly chosen from RCL
        TSP_PathU {cj}
        not_yet_visited \ {cj}
TSP_path U {depot}
end
return TSP_path
```

Αλγόριθμος 4.2 *Κατασκευή μονοπατιού TSP*

```
Procedure create_rcl( not_yet_visited, current_customer)
RCL = Ø
D ← maximum RCL size
for i = 0: min(D, length(not_yet_visited):
        dmin ← minimum distance from current_customer
        for c in not_yet_visited:
                if distance(c,current_customer) < dmin
                        RCL U c
                end if
        end
end
Return RCL
```

Αλγόριθμος 4.1 *Δημιουργία λίστας RCL*

```
Procedure construct_GRASP_solution()
## construction of the initial routes solution with GRASP
TSP_solution ← construct_TSP_solution()
TSP_solution ←3_opt_Local_Search()
x ←[ ]  ## initialize the solution as an empty list
sub_route ←[0]
for c_i in TSP_solution:
        if CVRP constraints are not violated:
                sub_route U {c_j}
        else
                sub_route U {0} ## vehicle returns to the depot
                x U {sub_route}

                sub_route ←{0} ## new vehicle starts from depot

        end if
end
return x
```

Αλγόριθμος 4.3 *Κατασκευή αρχικής λύσης*

4.4 Κίνηση των νυχτερίδων

4.4.1 Η απόσταση hamming (hamming distance)

Η απόσταση hamming (hamming distance) αποτελεί το ποσοστό των θέσεων ανάμεσα σε δύο λίστες αριθμών ίδιου μήκους, οι οποίες περιέχουν διαφορετικά στοιχεία.
Παράδειγμα hamming distance:
έστω a και b λύσεις σε διακριτή μορφή :

a =	**2**	1	**3**	4	5	6
b =	**3**	1	**2**	4	5	6

Παράδειγμα 4.3 *Υπολογισμός hamming distance*

παρατηρείται ότι hamming_distance(a,b) = 0,33 , διότι το 1ο και 3ο στοιχείο στις δύο λίστες περιέχουν διαφορετική τιμή. Στη συνέχεια, πολλαπλασιάζοντας με το μήκος των διανυσμάτων μετατρέπεται σε ακέραιο αριθμό.

4.4.2 Ο αλγόριθμος επανασύνδεσης μονοπατιών (Path Relinking)

Ο συγκεκριμένος μεθευρετικός αλγόριθμος προϋποθέτει την ύπαρξη δύο ή περισσότερων εφικτών λύσεων για τη λειτουργία του. Η διαδικασία ξεκινά επιλέγοντας μία λύση (starting solution), πάνω στην οποία θα γίνουν αλλαγές και μία δεύτερη λύση η οποία θεωρείται καλύτερη (target solution). Έπειτα καλείται μία επαναληπτική διαδικασία κατά την οποία ο αλγόριθμος PR προσπαθεί να μετατρέψει την πρώτη λύση ως ένα αντίγραφο της λύσης στόχου. Αναλυτικότερα η διαδικασία παρουσιάζεται στο παρακάτω παράδειγμα:

Αρχική:	x_t =	1	2	3	4	5
	x_s =	3	1	4	2	5
$1^η$ επανάληψη:	x_t =	1	2	3	4	5
	x_s =	1	3	4	2	5
$2^η$ επανάληψη:	x_t =	1	2	3	4	5
	x_s =	1	2	4	3	5
$3^η$ επανάληψη:	x_t =	1	2	3	4	5
	x_s =	1	2	3	4	5

Παράδειγμα 4.4 *Περιγραφή Path Relinking*

Έστω ότι επιλέχθηκε η λύση x_s και η βέλτιστη είναι η x_t όπως φαίνεται στο παράδειγμα 4.4. Ακολουθώντας τον αλγόριθμο PR το πρώτο βήμα είναι η αντικατάσταση του κόμβου με τον αριθμό 3 με τον κόμβο με τον αριθμό 1, όπως φαίνεται στην $1^η$ επανάληψη. Έπειτα, στη δεύτερη θέση της x_s πρέπει να αντικατασταθεί ο κόμβος 3 με τον κόμβο 2 πού βρίσκεται στην τέταρτη θέση. Η διαδικασία επαναλαμβάνεται και κατά την $3^η$ επανάληψη.

Κατά την εφαρμογή του αλγόριθμου PR, μετά από κάθε επανάληψη υπολογίζεται η αξία της νέας λύσης που δημιουργήθηκε και σε περίπτωση που το νέο κόστος αποδειχτεί χαμηλότερο, τότε σταματά ο αλγόριθμος και γίνεται αποδεκτή η νέα λύση. Αναλυτικά ο αλγόριθμος του Path Relinking παρουσιάζεται ως εξής:

```
Procedure PathRelinking(target, starting)
x_s ← starting
x_t ←target
for i = 0 : length(x_t):
        if x_s[i] ≠ x_t[i]:
                x_s[j] ← find_element(x_t[i], x_s) // find the element of x_s in the position i of x_t
                x_s' ← swap(x_s, x_s[i],x_s[j])
        end if
        if f(x_s') < f(x_s) :
                x_s' ← x_s
                break
        end if
end
return x_s'
```

Αλγόριθμος 4.4 *Path Relinking*

4.4.3 Ενημέρωση θέσης της κάθε νυχτερίδας

```
Procedure update_movement(x,x_best)
f'(x) ← update frequency of x using (1)
x_beast ← suffle_routes(x_best)  ##alter the sequence in which the routes are presented in the solution
hamming_elements = hamming_distance(xi , x_best)* len(x)
rnd ← random number in uniform distribution
if λ ≤ f'(x):
        #larger search space
        u_ι' ← random_number ∈ int(hamming_elements * f_j', hamming_elements * f_max)
end if
else if λ> f'(x): # λ is a number randomly chosen in (0,1) uniform
        #shorter search space
        u_i' ← random integer ∈ int(hamming_elements * f_min, hamming_elements * f_j')
end else if
Δ ←find_different_indices(x, x_best)
for i = 0 : v' do
        j ←find_element_position(x, x_best, Δ_i) ## find the index of the element of x_best in Δ_i which is in x
        xi ←swap(x, j, i])  ##replace the Δ_i index of x with the j element of x_best
        if F(x') < F(x):
                x' ← x_i
                break
        end if
end
end
return x'
```

Αλγόριθμος 4.5 *Κίνηση νυχτερίδων*

Σε αυτή την παραλλαγή του αλγόριθμου νυχτερίδας εφαρμόζεται μια εναλλακτική μέθοδος για την ενημέρωση της νέας θέσης της κάθε νυχτερίδας, η οποία είναι εμπνευσμένη από τον αλγόριθμο path relinking. Η συχνότητα f_i ενημερώνεται με τη μέθοδο (3.1) (σελ.25), όπως προβλέπει ο αρχικός BA. Εφόσον οι εξισώσεις (3.2) και (3.3) (σελ.25) της τυχαίας αναζήτησης δεν μπορούν να εφαρμοστούν στη διακριτή μορφή, επινοήθηκε μια μέθοδος ανακατασκευής της λύσης.

$$u_j = \begin{cases} \text{random}\left(\text{hamming}_{\text{elements}} * f_j', \text{hamming}_{\text{elements}} * f_{max}\right), & f_j \geq \lambda \\ random\left(\text{hamming}_{\text{elements}} * f_{min}, \text{hamming}_{\text{elements}} * f_j'\right), & αλλιώς \end{cases} \quad (4.4)$$

Αρχικά γίνεται ανακατανομή της σειράς των διαδρομών στη x_{best}. Αυτό συμβαίνει για να υπάρξει ποικιλότητα στις λύσεις που δημιουργούνται με τη μέθοδο που ακολουθεί. Στη συνέχεια, οι λύσεις x_{best} και x λαμβάνουν τη διακριτή μορφή διανύσματος, όπως στο παράδειγμα 4.2. Για τον υπολογισμό της νέας ταχύτητας u_i της νυχτερίδας, σε κάθε επανάληψη υπολογίζεται αρχικά η απόσταση hamming, από τη θέση της βέλτιστης νυχτερίδας του πληθυσμού (x_{best}). Εφόσον υπολογιστεί η hamming απόσταση μεταξύ x_{best} και x σε ακέραια μορφή, ο επανυπολογισμός της ταχύτητας εξαρτάται από τη νέα συχνότητα (4.4). Αν η συχνότητα είναι μικρότερη από ένα σταθερό αριθμό λ (στον συγκεκριμένο αλγόριθμο λ=0,5), η νυχτερίδα εκτελεί μεγαλύτερο εύρος κίνησης, ενώ στην αντίθετη περίπτωση μικρότερο. Με τον όρο «εύρος κίνησης» εννοείται ο αριθμός των στοιχείων που θα εξεταστούν για επανατοποθέτηση.

Η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί καλύτερη λύση για την εξεταζόμενη νυχτερίδα ή μέχρι να εξεταστούν όλες οι κινήσεις. Σε περίπτωση όμως που η νέα λύση γίνει αντίγραφο της x_{best} τότε απορρίπτεται.

4.5 Τοπική αναζήτηση (local search)

Για τη βελτίωση της θέσης κάθε νυχτερίδας, ξεχωριστά, χρησιμοποιήθηκε ο αλγόριθμος μεταβλητής γειτονιάς αναζήτησης VNS (Variable Neighborhood Search). Ο αλγόριθμος VNS προτάθηκε από τους Hansen και Mlandenovic[10]. Σκοπός του αλγόριθμου είναι η εύρεση καλύτερης λύσης, μετά από μία επαναληπτική αναζήτηση σε ένα σύνολο από γειτονιές. Με άλλα λόγια, η μέθοδος συνδυάζει διαφορετικούς αλγόριθμους τοπικής αναζήτησης για να επιτευχθεί έξοδος από πιθανό τοπικό ελάχιστο. Η διαδικασία ξεκινά αρχικά με την επιλογή των μεθόδων τοπικής αναζήτησης που θα χρησιμοποιηθούν. Οι γειτονιές αναζήτησης επιλέγονται τυχαία. Εάν βρεθεί καλύτερη λύση σε μια γειτονιά τότε η αναζήτηση επαναλαμβάνεται από την αρχή, με σκοπό την περαιτέρω βελτίωση της λύσης στην ίδια γειτονιά. Εάν η αναζήτηση σε μια γειτονιά ξεπεράσει τον μέγιστο αριθμό επαναλήψεων, τότε αλλάζει η γειτονιά. Με αυτόν τον τρόπο, επιτυγχάνεται η συνεχής βελτίωση της λύσης και η αποφυγή τοπικών ελαχίστων, στα οποία θα οδηγούσαν οι αλγόριθμοι τοπικής αναζήτησης ατομικά.

Για την εφαρμογή του VNS στον HBA, χρησιμοποιήθηκαν οι παρακάτω μέθοδοι τοπικής αναζήτησης, ομαδοποιημένοι σε τοπική αναζήτηση (local search) και εντατικοποιημένη τοπική αναζήτηση (intense local search):

Local Search	Intense Local Search
• 2-opt • 1-1 exchange • 2-2 exchange • 0-1 relocate	• 0-2 relocate • 1-1 swap intra-route • 2-2 swap intra-route

```
Procedure Variable_Neighborhood_Search(x)
{N_l} ← choose local search neighborhoods (1,…,l_nax)
l = 1
while l ≤ l_max :
        x' ← Local_Search(x,N_l)  ## find neighbor of x using local search in N_l
        if f(x') < f(x):
                x ← x'
                l = 1
        else:
                l += 1
        end if
end
return x'
```

Αλγόριθμος 4.6 *Αλγόριθμος VNS*

Βήματα αναζήτησης:

Βήμα 1: Αναζήτηση ανά διαδρομή και ανά κόμβο.
Βήμα 2: Επιλογή του συνδυασμού κόμβων που αποφέρουν μείωση κόστους.
Βήμα 3: Εφαρμογή της μεθόδου τοπικής αναζήτησης στους επιλεγμένους κόμβους.
Βήμα 4: Επανάληψη της διαδικασίας έως ότου ικανοποιηθεί το κριτήριο τερματισμού.

4.5.1 Μέθοδος 2-opt:

Κατά την μέθοδο 2-opt πραγματοποιείται η διαγραφή δύο ακμών και η επανασύνδεση δύο νέων μονοπατιών, παράγοντας μια νέα διαδρομή. Κατά την εφαρμογή της μεθόδου, ελέγχεται εάν το νέο κόστος που προκύπτει από την επανασύνδεση των τόξων είναι μικρότερο και μόνο τότε πραγματοποιείται η συγκεκριμένη κίνηση. Με αυτόν τον τρόπο αποφεύγεται η παραγωγή λύσεων μεγαλύτερου κόστους.

Παράδειγμα:

Πριν την εφαρμογή 2-opt:	0	7	1	3	5	0
Μετά την εφαρμογή 2-opt:	0	3	1	7	5	0

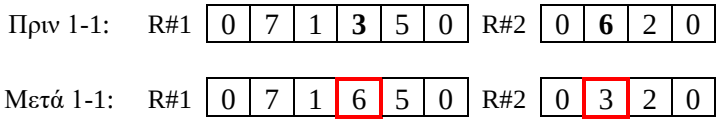
Παράδειγμα 4.5 *Μέθοδος 2-opt*

Όπως φαίνεται στο παράδειγμα 4.1, τα τόξα που διαγράφονται είναι τα 0 – 7 και 3 – 5, έπειτα το μονοπάτι 7 – 1 – 3 αντιστρέφεται και επανασυνδέεται με τις ακμές 0 και 5.

4.5.2 Μέθοδος 1-1 exchange και 2-2 :

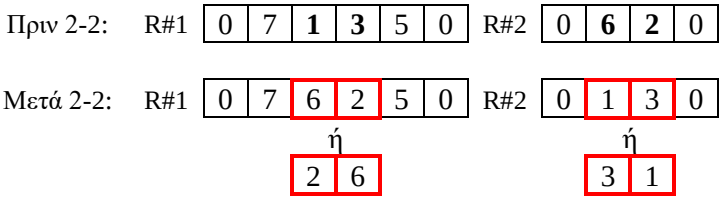
Η μέθοδος 1-1 exchange είναι ένας ακόμη τρόπος τοπικής αναζήτησης. Σε αυτή τη μέθοδο πραγματοποιείται ανταλλαγή 2 στοιχείων από 2 διαφορετικές διαδρομές με αποτέλεσμα τη δημιουργία μιας νέας λύσης. Πριν από την ανταλλαγή των δύο ακμών, εξετάζεται αν το νέο κόστος διαδρομής συμφέρει. Κατά αυτόν τον τρόπο αποφεύγεται η δημιουργία χειρότερης λύσης.

Παράδειγμα 1-1 exchange:



Παράδειγμα 4.6 *Μέθοδος 1-1 exchange*

Παράδειγμα 2-2 exchange:



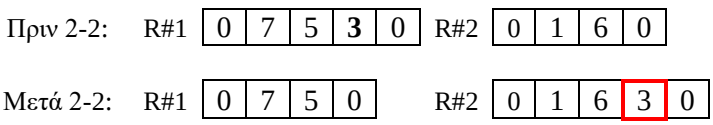
Παράδειγμα 4.7 *Μέθοδος 2-2 exchange*

Στο παράδειγμα 4.6 επιλέγεται ο κόμβος με την τιμή 3 από την πρώτη διαδρομή (R#1) και ο κόμβος με την τιμή 6 από την δεύτερη διαδρομή R#2. Έπειτα πραγματοποιείται η ανταλλαγή των δύο αυτών κόμβων. Στο παράδειγμα 4.7 πραγματοποιείται η ανταλλαγή δύο ζευγαριών διαδοχικών κόμβων σε δύο διαφορετικές διαδρομές και εξετάζεται επίσης και η επανατοποθέτησή τους σε αντίστροφη σειρά.

4.5.3 Μέθοδος 0-1 relocate

Η μέθοδος 0-1 relocate είναι επίσης ένας αλγόριθμος τοπικής αναζήτησης, κατά τον οποίο επιλέγεται ένας κόμβος μιας διαδρομής και μετατοπίζεται σε μια άλλη, παράγοντας μια νέα λύση. Προτού γίνει η κίνηση, ελέγχεται όπως και στις παραπάνω μεθόδους το νέο κόστος που προκύπτει για να εξεταστεί αν η νέα λύση είναι αποδεχτή. Επίσης, ελέγχεται η διαδρομή από την οποία μετατοπίζεται ο κόμβος και στην περίπτωση που έχει πλέον μόνο δύο κόμβους (απαρτίζεται μόνο από τον κόμβο της αποθήκης), διαγράφεται ολόκληρη η διαδρομή.

Παράδειγμα 0-1 relocate:



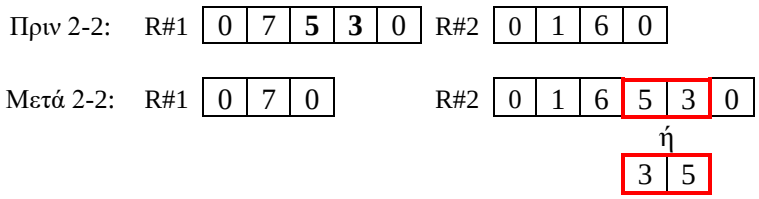
Παράδειγμα 4.7 *Μέθοδος 0-1 relocate*

Στο παράδειγμα 4.7 επιλέχθηκε ο κόμβος με τον αριθμό 3 από την πρώτη διαδρομή R#1, ώστε να διαγραφεί από αυτή και να προστεθεί μετά τον κόμβο με τον αριθμό 6 στην δεύτερη διαδρομή. Το παράδειγμα 4.8 αναπαριστά την μέθοδο 0-2 relocate, όπου εξετάζεται η επανατοποθέτηση με ή χωρίς αντιστροφή ενός ζεύγους διαδοχικών κόμβων μιας διαδρομής (5 κα 3 στην R#1) σε μία άλλη διαδρομή (R#2).

4.6 Εντατικοποιημένη τοπική αναζήτηση (intense local search)

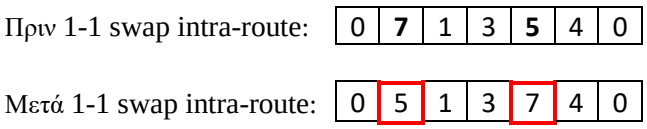
Η intense local search είναι μια διαδικασία που συμπληρώνει την τοπική αναζήτηση, προσθέτοντας γειτονιές που μπορεί να βελτιώσουν πιο εντατικά τις λύσεις, αλλά κατά την εφαρμογή της απαιτείται ελάχιστα πιο αυξημένος υπολογιστικός χρόνος. Η intense local search χρησιμοποιεί τον VNS που περιέχει τις μεθόδους 0-2 relocate, 1-1 swap intra-route και 2-2 swap intra-route. Τα βήματα που ακολουθούνται στις δύο τελευταίες είναι τα ίδια με τις 1-1 και 2-2 exchange, όπως παρουσιάστηκαν παραπάνω, με μόνη διαφορά ότι εφαρμόζονται μεταξύ κόμβων της ίδιας διαδρομής.

Παράδειγμα 0-2 relocate:



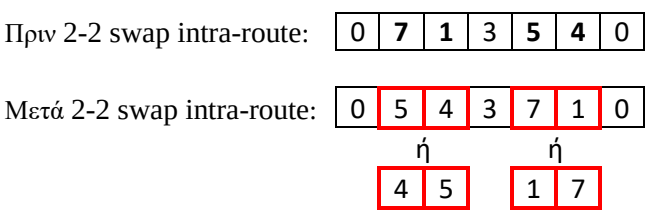
Παράδειγμα 4.8 *Μέθοδος 0-2 relocate*

Παράδειγμα 1-1 swap intra-route:



Παράδειγμα 4.9 *1-1 swap intra-route*

Παράδειγμα 2-2 swap intra-route:



Παράδειγμα 4.10 *2-2 swap intra-route*

4.7 Διαδικασία προτεινόμενου αλγορίθμου

Ο υβριδικός αλγόριθμος νυχτερίδας (HBA) που προτείνεται σε αυτή την εργασία φέρει ορισμένα όμοια στοιχεία με τον αρχικό BA. Σε πρώτη φάση, γίνεται αρχικοποίηση των παραμέτρων και των νυχτερίδων που απαρτίζουν τον αρχικό πληθυσμό. Η αρχική θέση x_j της κάθε νυχτερίδας αποτελεί μια εφικτή λύση, δημιουργημένη από τη μέθοδο GRASP που αναλύθηκε παραπάνω (αλγόριθμος 4.3), λαμβάνοντας επίσης την αρχική τιμή κόστους της $F(x_j)$ υπολογισμένη από την αντικειμενική συνάρτηση. Η συχνότητα f_j λαμβάνει αρχικές τυχαίες τιμές σύμφωνα με την ομοιόμορφη κατανομή στο διάστημα $[f_{min},f_{max}]$, στη συνέχεια η ταχύτητα u_j αρχικοποιείται με 0 και τέλος καθορίζονται τυχαία στο διάστημα [0,1] και [1,2] ο αρχικός παλμικός ρυθμός $r0_j$ και η ένταση ld_j αντίστοιχα . Εφόσον οι παραπάνω αρχικοποιήσεις πραγματοποιούνται με τυχαίο τρόπο για κάθε νυχτερίδα, επιτυγχάνεται η εξάπλωση τους σε ένα ευρύτερο πεδίο τιμών της αντικειμενικής συνάρτησης και αυξάνεται η πιθανότητα εύρεσης της βέλτιστης λύσης, μέσα από τις μεθόδους αναζήτησης.

Εν συνεχεία της πρώτης φάσης, ακολουθεί η ενημέρωση της θέσης των νυχτερίδων μέσω της διαδικασίας update_movement (αλγόριθμος 4.5). Σύμφωνα με τη διαδικασία αυτή, οι νυχτερίδες εκτελούν μια ευρύτερη αναζήτηση (global search) στο πεδίο λύσεων. Κατά την ευρύτερη αναζήτηση, κάθε νυχτερίδα πραγματοποιεί ταυτόχρονα εξερεύνηση και εκμετάλλευση (exploration – exploitation) στο ευρύτερο πεδίο λύσεων. Ο συνδυασμός εξερεύνησης και εκμετάλλευσης επιτυγχάνει τόσο την εξερεύνηση νέων λύσεων με τυχαίες κινήσεις (όπως με την μέθοδο επιλογής μιας τυχαίας γειτονιάς κόμβων) όσο και την εκμετάλλευση της καλύτερης λύσης του πληθυσμού κάθε γενιάς για τη δόμηση μιας ακόμα καλύτερης λύσης.

Στη συνέχεια, ακολουθεί η διαδικασία της τοπικής αναζήτησης για κάθε νυχτερίδα που μπορεί να οδηγήσει σε περεταίρω βελτίωση της θέσης της, με τον αλγόριθμο VNS. Είναι σημαντικό να σημειωθεί πως σε προβλήματα με διακριτές μεταβλητές, όπως το CVRP, η εφαρμογή τοπικής αναζήτησης σε όλα τα μέλη του πληθυσμού του Bat Algorithm εγγυάται μεγαλύτερη πιθανότητα αποφυγής των τοπικών ελαχίστων. Το συγκεκριμένο γεγονός μπορεί να αιτιολογηθεί λόγω του συνδυασμού της ευρύτερης και τοπικής αναζήτησης. Όπου σύμφωνα με την πρώτη, η κάθε λύση αποκτά κοινά «καλά» στοιχεία με την προσωρινά βέλτιστη (x_{best}) και με την δεύτερη μπορεί γρήγορα να αναζητήσει στη δική της γειτονιά καλύτερες λύσεις. Χωρίς την τοπική αναζήτηση σε κάθε νυχτερίδα, αυξάνεται η πιθανότητα ο αλγόριθμος να εγκλωβιστεί σε τοπικό ελάχιστο. Για παράδειγμα, μια x_{best} η οποία αδυνατεί να βελτιωθεί ενδέχεται να οδηγήσει τις υπόλοιπες λύσεις στη γειτονιά του τοπικού ελάχιστου. Εφόσον έχει εξεταστεί η ενημέρωση της x_j, που εξετάζεται, ακολουθεί όπως και στον αρχικό BA, η διαδικασία της τοπικής αναζήτησης σε ένα από τα καλύτερα μέλη του σμήνους. Αυτή η φάση ενεργοποιείται, όταν ο παλμικός ρυθμός (pr_j) της νυχτερίδας που εξετάζεται είναι μικρότερος από ένα τυχαίο αριθμό της ομοιόμορφης κατανομής στο πεδίο [0,1]. Σε αυτή την περίπτωση, επιλέγεται μια λύση από τις καλύτερες, δηλαδή από την elite list, στην οποία θα εφαρμοστεί μια εντατικοποιημένη τοπική αναζήτηση (intense local search), παράλληλα με την τοπική αναζήτηση. Στην επόμενη φάση εξετάζεται αν θα γίνει αποδεχτή μια νέα καλύτερη λύση από την εξεταζόμενη νυχτερίδα. Η αποδοχή της νέας λύσης εξαρτάται από το εάν η ένταση της ld_j είναι μεγαλύτερη από ένα τυχαίο αριθμό. Τέλος, εφόσον έχει εξεταστεί και η τελευταία νυχτερίδα του πληθυσμού της κάθε γενιάς, πραγματοποιείται κατάταξη των λύσεων και

βρίσκεται η βέλτιστη λύση. Η διαδικασία επαναλαμβάνεται μέχρι να εκπληρωθεί κάποιο κριτήριο τερματισμού. Στην παρούσα περίπτωση, ο τερματισμός επέρχεται είτε μετά από ένα ανώτερο όριο επαναλήψεων (MAX_GEN) είτε μετά από ένα αριθμό διαδοχικών επαναλήψεων (STOP_GEN) χωρίς την ύπαρξη νέας λύσης.

```
Procedure Hybrid_Bat_Algorithm()
ps ←set the population size
MAX_GEN ← set the maximum generation number
STOP_GEN ← set the maximum generation number without new solution
## swarm initialization
for i = 1: ps :
        x_i ← construct_GRASP_solution()
        F(x_I) ← evaluate the cost for each bat
        f_i ← random uniform ∈ [f_min,f_max]## initialize frequency randomly
        u_i ← 0                          ## set velocity to 0 for each bat
        pr_i ← r0                        ## initialize the pulse rate of each individual
        ld_i ← random uniform ∈ [1,2]
end
EL ← construct_elite_list()
x_best ← find the best solution in the population
k ← counter
while i < MAX_GEN or k == STOP_GEN:
        for j = 1: ps :
                x_j ← update_movement(x_best, x_i)
                x_i ← local_search(xj)
                if pr_i < rnd_n1 :
                        x_* ← select_one_from_elite (EL)
                        x_* ← intense_local_search(χ_*)
                end
                if  ld_j  > rnd_n2 and F(x_j) < F(x_best):
                        x_j ← x_best
                        ld_j, pr_j   ← update_pr_ld() # use function (1), (2)
                end
        x_best ← the best solution among the population of the generation
        if new solution:
                k = 0
        else
                k += 1
        i += 1
        end
end
return x_best
```

Αλγόριθμος 4.7 *Υβριδικός Αλγόριθμος Νυχτερίδας κύριο μέρος*

Κεφάλαιο 5 - Παρουσίαση και ανάλυση αποτελεσμάτων

Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα της εφαρμογής του προτεινόμενου αλγορίθμου για την επίλυση ορισμένων προβλημάτων δρομολόγησης. Για την υλοποίηση του HBA αναπτύχθηκε κώδικας σε γλώσσα Python 3.10. Ο κώδικας που δημιουργήθηκε έτρεξε σε υπολογιστή με λειτουργικό Windows 11, επεξεργαστή AMD Ryzen 5 5625U 2.30 GHZ και μνήμη RAM 16 GB.

5.1 Παράμετροι του αλγόριθμου

Οι παράμετροι που χρησιμοποιήθηκαν για την εφαρμογή του προτεινόμενου αλγόριθμου παρουσιάζονται στον πίνακα 5.1.1. Ο μέγιστος αριθμός επαναλήψεων του προγράμματος τέθηκε σε 300, καθώς και ο αριθμός των επαναλήψεων που θα τερματίζει χωρίς νέα καλύτερη λύση σε 50. Το μήκος της RCL ρυθμίστηκε σε 4. Οι παράμετροι fmin, fmax τέθηκαν σε 0 και 1 αντίστοιχα. Οι παράμετροι α και γ των εξισώσεων παλμικού ρυθμού (3.5) και έντασης (3.6) τίθενται στην ίδια τιμή που παρουσιάζονται στον αρχικό BA του Yang 0.9 και τα δύο. Η μόνη παράμετρος που πρέπει να εξεταστεί προς ρύθμιση, για την επίλυση των προβλημάτων που θα ακολουθήσουν, είναι ο αριθμός των νυχτερίδων που θα αποτελέσουν τον αρχικό πληθυσμό.

Παράμετροι	**Τιμές**
MAX_ITER	300
STOP_ITER	50
L (RCL size)	4
F_{min}	0
F_{max}	1
α	0.9
γ	0.9

Πίνακας 5.1.1 Παράμετροι HBA

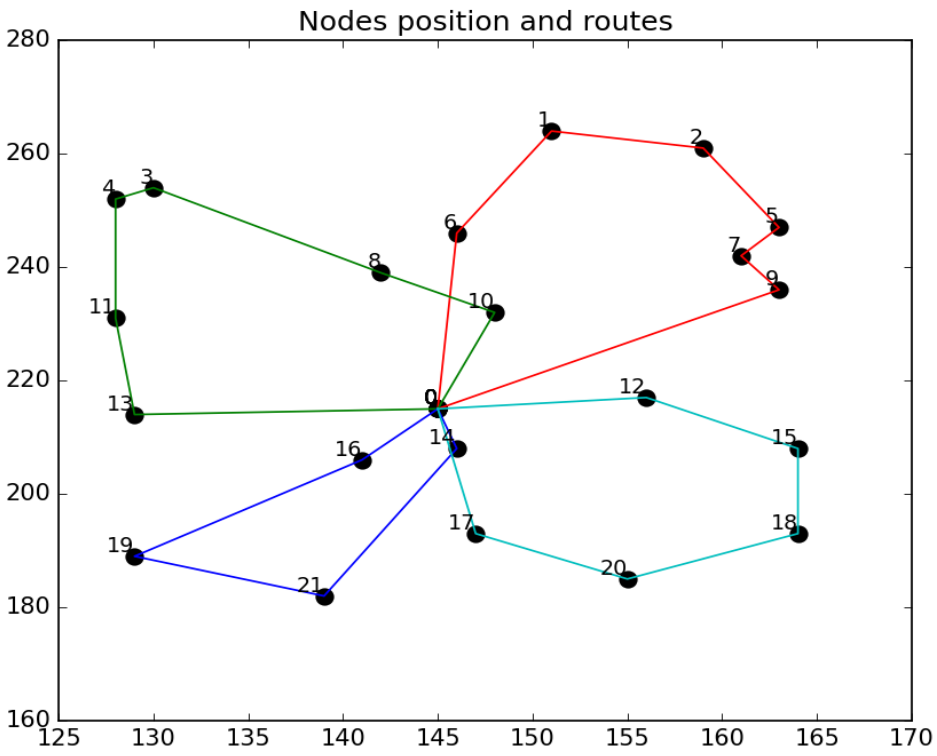
Για τον καθορισμό του πληθυσμού (ps) που αποδίδει τα καλύτερα αποτελέσματα, εξετάστηκαν πειραματικά πληθυσμοί των 20, 50, 100 και 200 νυχτερίδων σε ένα σμήνος για την επίλυση των προβλημάτων από τη δομή E (Set E Christofides and Elion, 1969), E-n022-k04, E-n033-k04 και E-n051-k05. Τα αποτελέσματα παρουσιάζονται ως εξής:

5.1.1 Πρόβλημα E-n022-k04 :

Number of Nodes	22
Vehicle Capacity	6000
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.1.2 *Δεδομένα προβλήματος* ***E-n022-k04***

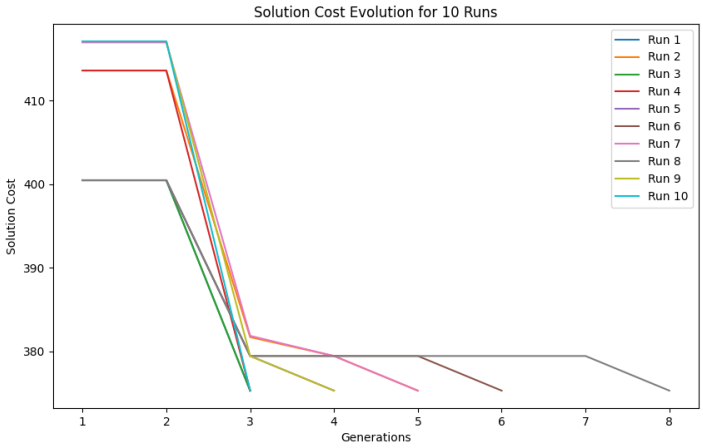
Best solution cost found: 375,2798



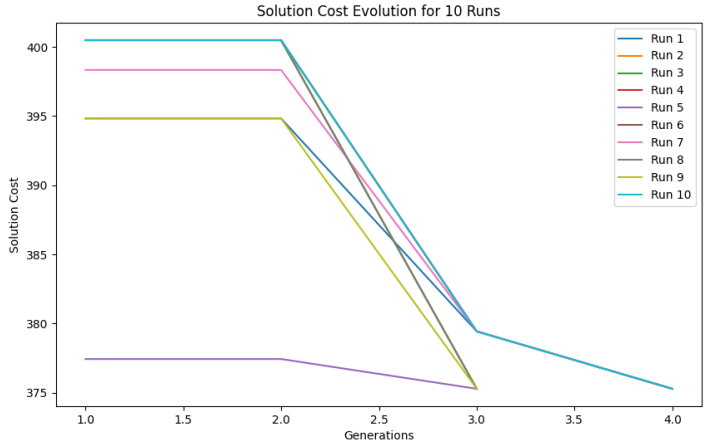
Γράφημα 5.1.1 *Λύση* ***E-n022-k04*** *στο επίπεδο xy*

	ps = 20		ps = 50		ps = 100		ps =200	
RUNS	BEST COST	RUNTIME	BEST COST	RUNTIME	BEST COST	RUNTIME	BEST COST	RUNTIME
1	375,2798	0,8	375,2798	2	375,2798	1	375,2798	1,4
2	375,2798	1	375,2798	1,8	375,2798	0,7	375,2798	1,4
3	375,2798	0,8	375,2798	2	375,2798	1	375,2798	1,5
4	375,2798	0,8	375,2798	2	375,2798	0,4	375,2798	1
5	375,2798	0,9	375,2798	2	375,2798	0,4	375,2798	1,4
6	375,2798	1	375,2798	2	375,2798	0,3	375,2798	1,3
7	375,2798	0,9	375,2798	2	375,2798	0,7	375,2798	1,4
8	375,2798	1	375,2798	2	375,2798	1	375,2798	1,4
9	375,2798	1	375,2798	2	375,2798	0,5	375,2798	1,4
10	375,2798	0,8	375,2798	2	375,2798	0,7	375,2798	1,2
BEST	**375,2798**	**0,8**	**375,2798**	**1,8**	**375,2798**	**0,3**	**375,2798**	**1**
AVG	**375,2798**	**0,9**	**375,2798**	**1,98**	**375,2798**	**0,67**	**375,2798**	**1,34**
PERFORMANCE	**1**		**1**		**1**		**1**	

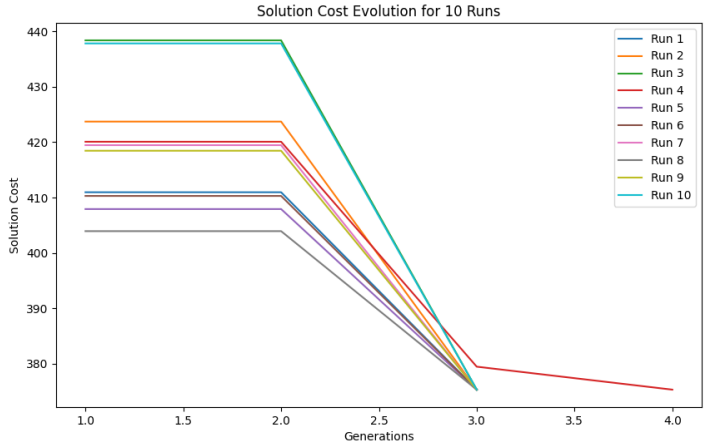
Πίνακας 5.1.3 *επίλυση προβλήματος* ***E-n022-k04***



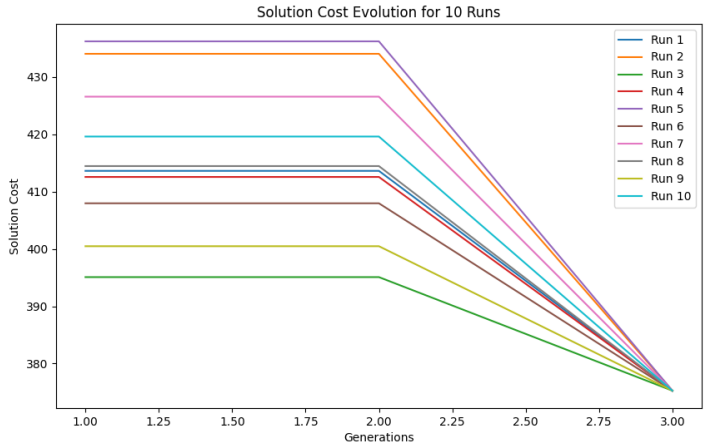
Γράφημα 5.1.2 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 20* ***E-n022-k04***



Γράφημα 5.1.3 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 50* ***E-n022-k04***



Γράφημα 5.1.4 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 100* ***E-n022-k04***



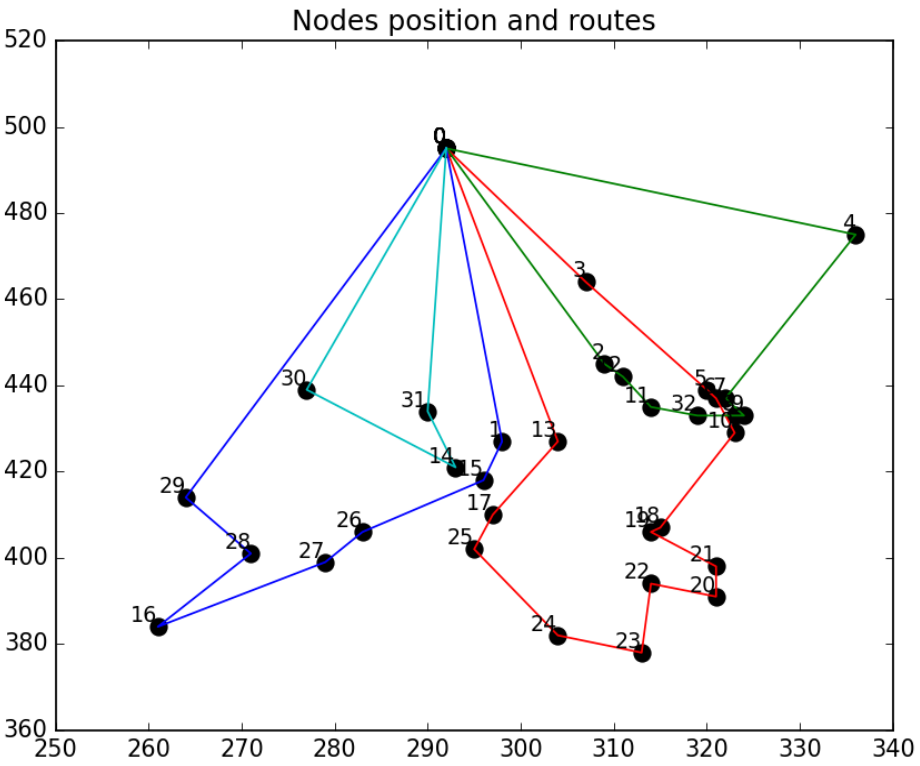
Γράφημα 5.1.5 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 200* ***E-n022-k04***

5.2.2 Πρόβλημα **E-n033-k04** :

Number of Nodes	22
Vehicle Capacity	6000
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.1.4 *Δεδομένα προβλήματος* ***E-n033-k04***

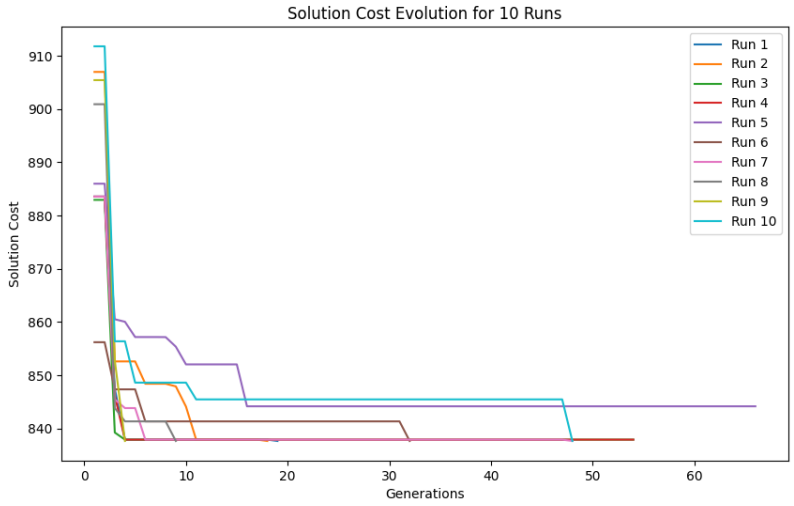
Best solution cost found: 837,6717



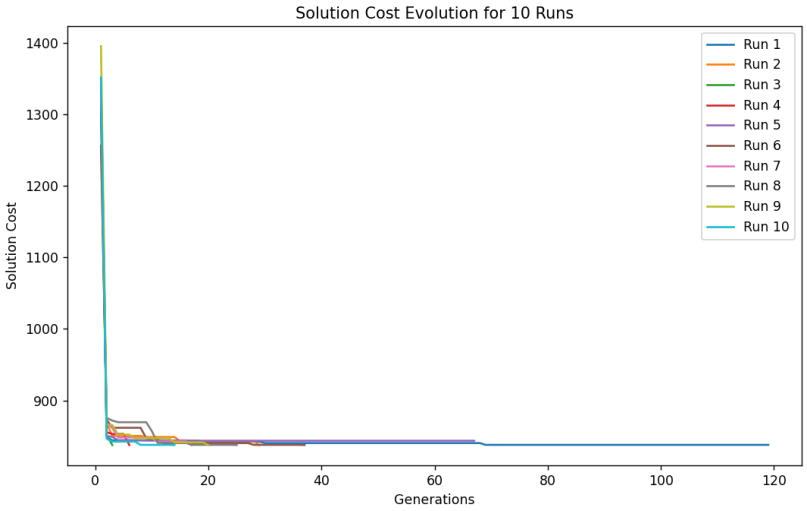
Γράφημα 5.1.7 *Λύση* ***E-n033-k04*** *στο επίπεδο xy*

	ps = 20		ps = 50		ps = 100		ps =200	
RUNS	BEST COST	RUNTIME	BEST COST	RUNTIME	BEST COST	RUNTIME	BEST COST	RUNTIME
1	837,6717	8	837,6717	90	837,6717	18	837,6717	8
2	837,6717	7	837,6717	35	837,6717	20	837,9255	7
3	837,9255	20	837,6717	4	837,6717	19	837,6717	65
4	837,9255	19	837,6717	5	837,6717	18	837,6717	25
5	844,1853	25	837,6717	87	837,6717	12	837,6717	4
6	837,6717	13	837,6717	45	837,6717	22	837,6717	14,9
7	837,6717	19	837,6717	16	837,6717	20	837,6717	69
8	837,6717	5	837,6717	24	837,6717	19	837,6717	8
9	837,6717	3	837,6717	18	837,6717	11	837,6717	100
10	837,6717	20	837,6717	12	837,6717	20	837,6717	10
BEST	**837,6717**	**3**	**837,6717**	**4**	**837,6717**	**11**	**837,6717**	**4**
AVG	**838,37382**	**13,9**	**837,6717**	**33,6**	**837,6717**	**17,9**	**837,69708**	**31,09**
PERFORMANCE	**0,7**		**1**		**1**		**0,9**	

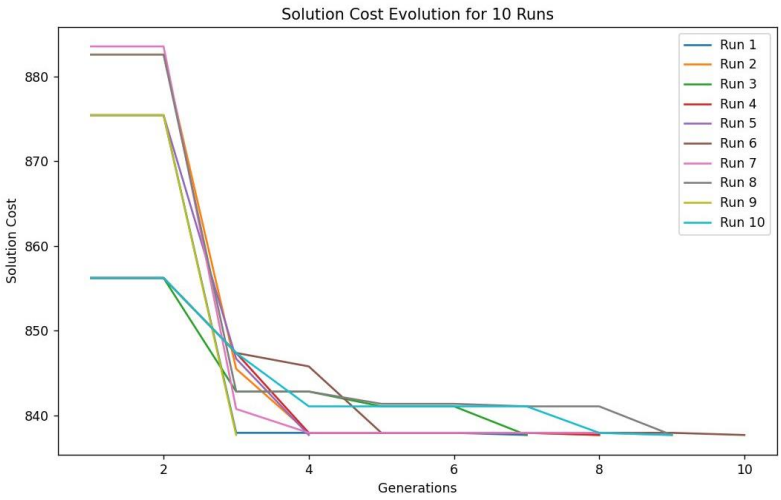
Πίνακας 5.1.5 *επίλυση προβλήματος* ***E-n033-k04***



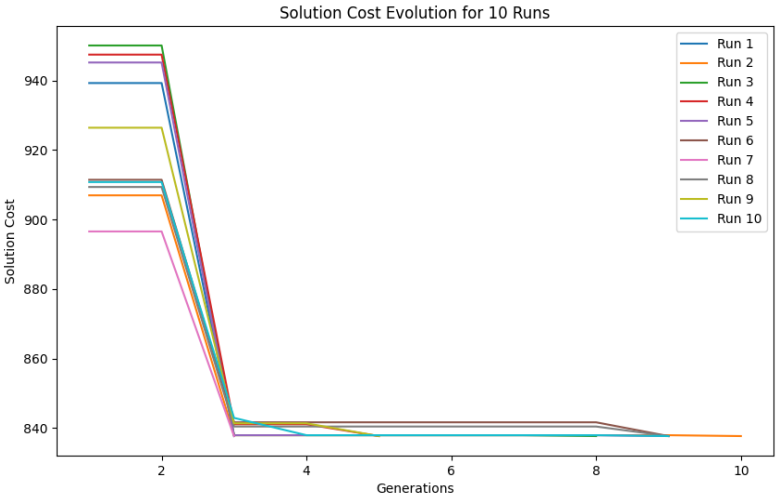
Γράφημα 5.1.8 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 20* ***E-n033-k04***



Γράφημα 5.1.9 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 50* ***E-n033-k04***



Γράφημα 5.1.10 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 100* ***E-n033-k04***



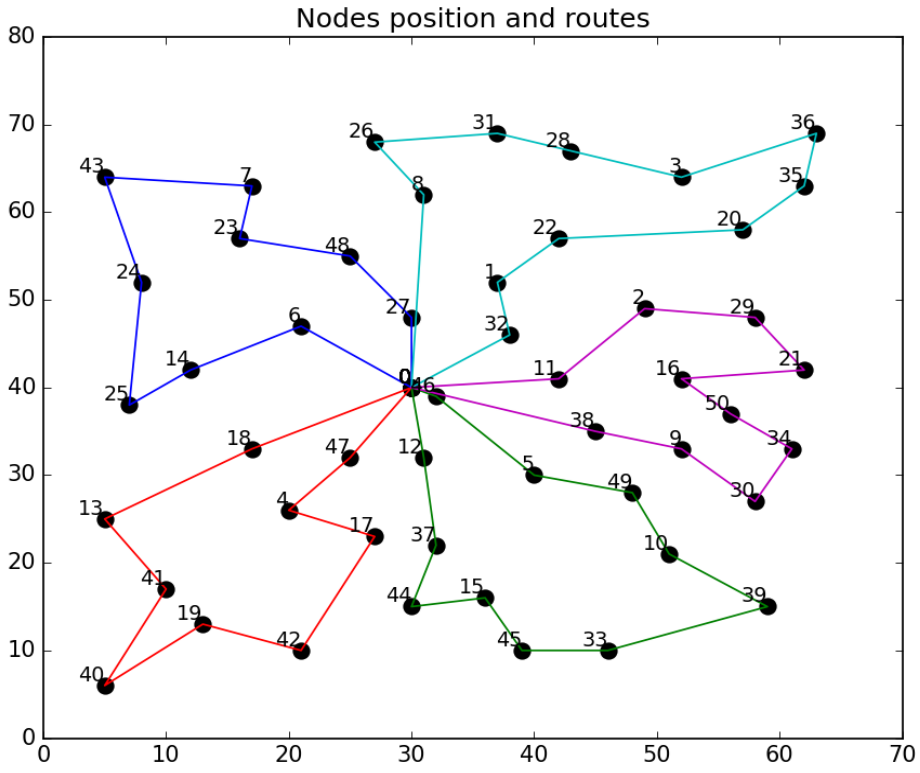
Γράφημα 5.1.11 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 200* ***E-n033-k04***

5.2.3 Πρόβλημα **CMT01** :

Number of Nodes	51
Vehicle Capacity	160
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.1.6 *Δεδομένα προβλήματος* ***CMT01***

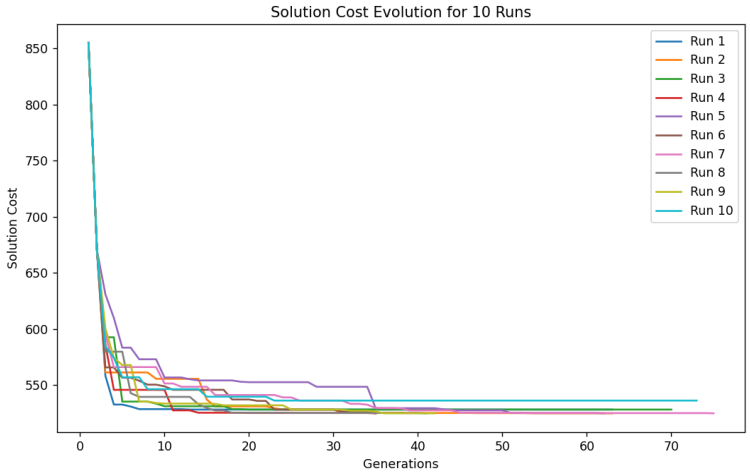
Best solution cost found: 524,6113



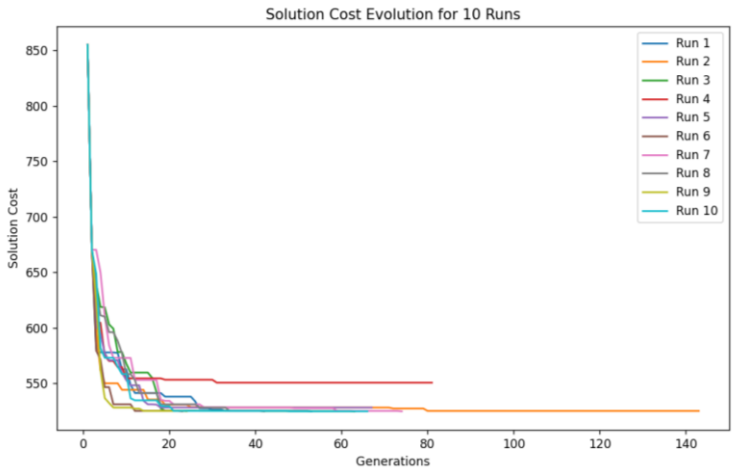
Γράφημα 5.1.12 *Λύση* ***CMT01*** *στο επίπεδο xy*

	ps = 20		ps = 50		ps = 100		ps =200	
RUNS	BEST COST	RUNTIME(sec)	BEST COST	RUNTIME	BEST COST	RUNTIME	BEST COST	RUNTIME
1	**524,6113**	53	**524,6113**	53	**524,6113**	108	**524,6113**	190
2	538,4949	44	524,9269	160	**524,6113**	79	**524,6113**	176
3	**524,6113**	33	**524,6113**	70	**524,6113**	96	**524,6113**	166
4	524,6286	78	550,4584	102	**524,6113**	70	**524,6113**	210
5	537,6067	47	527,6751	77	530,2934	141	527,9778	530
6	537,806	56	**524,6113**	20	524,6286	143	**524,6113**	152
7	**524,6113**	18	**524,6113**	81	**524,6113**	79	**524,6113**	251
8	546,3219	38	**524,6113**	35	**524,6113**	123	**524,6113**	431
9	524,6269	45	**524,6113**	20	**524,6113**	68	532,9959	580
10	553,727	35	**524,6113**	76	**524,6113**	85	**524,6113**	161
BEST	**524,6113**	**18**	**524,6113**	**20**	**524,6113**	**68**	**524,6113**	**152**
AVERAGE	**533,70459**	**44,7**	**527,53395**	**69,4**	**525,18124**	**99,2**	**525,78641**	**284,7**
PERFORMANCE	**0,3**		**0,7**		**0,8**		**0,8**	

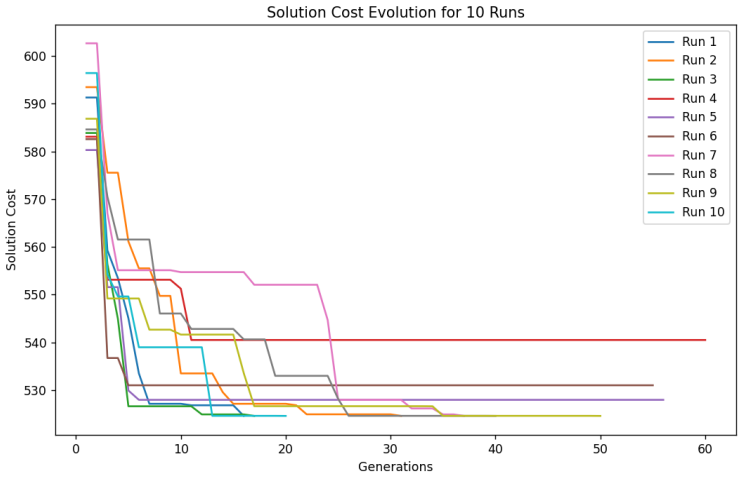
Πίνακας 5.1.7 *επίλυση προβλήματος* ***CMT01***



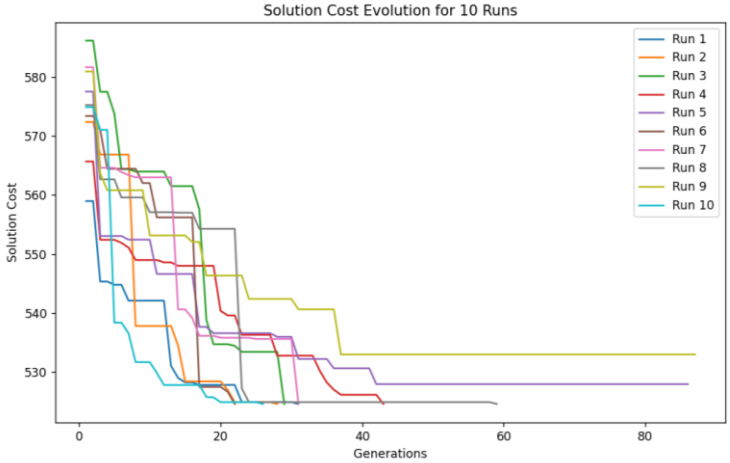
Γράφημα 5.1.13 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 20* ***CMT01***



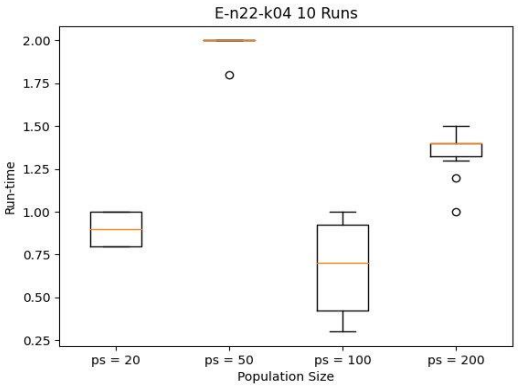
Γράφημα 5.1.14 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 50* ***CMT01***



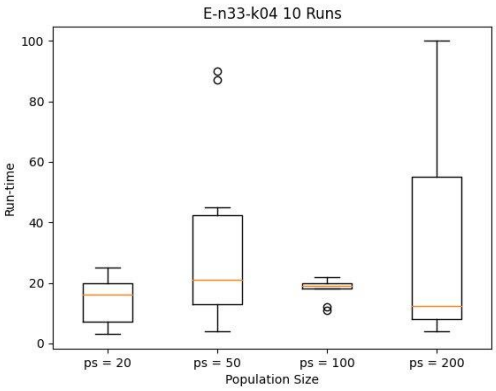
Γράφημα 5.1.15 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 100* ***CMT01***



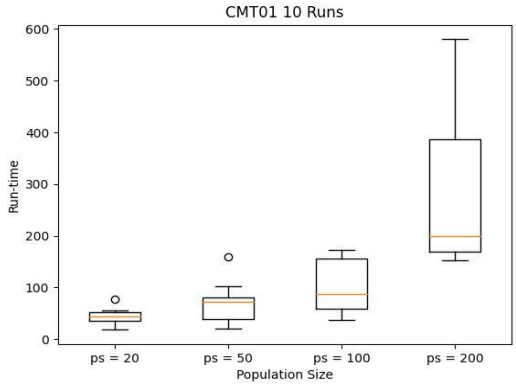
Γράφημα 5.1.16 *Εξέλιξη βέλτιστου κόστους ανά γενιά ps = 200* ***CMT01***



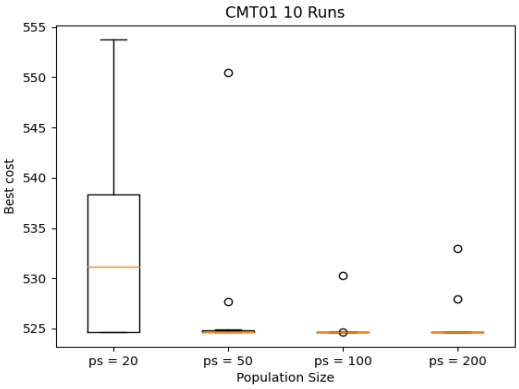
Γράφημα 5.1.17 *Χρόνος εκτέλεσης προγράμματος ανά πληθυσμό* ***E-n22-k04***



Γράφημα 5.1.18 *Χρόνος εκτέλεσης προγράμματος ανά πληθυσμό* ***E-n33-k04***



Γράφημα 5.1.19 *Χρόνος εκτέλεσης προγράμματος ανά πληθυσμό* ***CMT01***



Γράφημα 5.1.20 *Καλύτερη λύση* ***CMT01*** *ανά πληθυσμό*

5.2.4 Ρύθμιση παραμέτρου πληθυσμού νυχτερίδων

Σύμφωνα με τον πίνακα 5.2.3 και τα γραφήματα από 5.2.2 έως 5.2.5 μπορεί να παρατηρηθεί ότι ανεξαρτήτου μεγέθους πληθυσμού, ο κώδικας δεν εγκλωβίστηκε σε τοπικό ελάχιστο και βρήκε την βέλτιστη γνωστή λύση του προβλήματος E-n22-k04 σε σύντομο χρονικό διάστημα και στις 10 επαναλήψεις. Ωστόσο ο χρόνος που χρειάστηκε για την εύρεση της λύσης αποδεικνύεται κατά μέσο όρο λιγότερος στον πληθυσμό των 100 νυχτερίδων σύμφωνα με το box γράφημα 5.2.17. Στη συνέχεια, όσον αφορά το πρόβλημα E-n33-k04 παρατηρείται πως ο πληθυσμός ps = 20 δεν είναι αρκετός για την 100% απόδοση του αλγορίθμου (εύρεση του ολικού ελάχιστου), όπως συμβαίνει με τους πληθυσμούς 50, 100 και 200. Επίσης, αυτό το πρόβλημα, με βάση τον πίνακα 5.2.5 και το box γράφημα 5.2.18, φαίνεται να επιλύεται σε συντομότερο χρόνο με τον πληθυσμό ps = 100. Λαμβάνοντας υπόψη και τα αποτελέσματα από την επίλυση του μεγαλύτερου από τα τρία παραπάνω προβλήματα, του CMT01 στον πίνακα 5.2.7, τα γραφήματα 5.2.13 έως 5.2.16 και το box γράφημα 5.2.20, παρατηρείται ότι η απόδοση του προγράμματος είναι καλύτερη με τη χρήση 100 και 200 νυχτερίδων σε αρχικό πληθυσμό, ενώ με ps = 100 δίδει αποτελέσματα σε συντομότερο χρόνο, όπως δείχνει το γράφημα 5.2.19. Επομένως, για την επίλυση των παρακάτω προβλημάτων μεγαλύτερου εύρους κόμβων επιλέγεται το μέγεθος πληθυσμού ps = 100.

5.2 Παρουσίαση αποτελεσμάτων από μεγαλύτερα προβλήματα

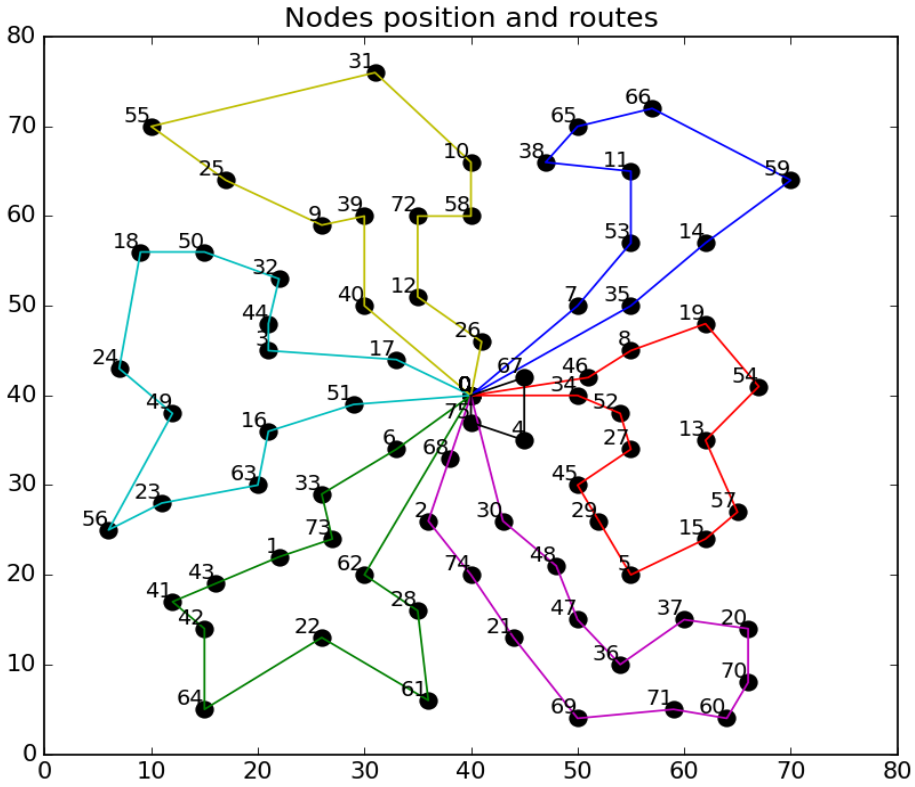
Για την αξιολόγηση της απόδοσης του αλγόριθμου σε μεγαλύτερα και πιο απαιτητικά προβλήματα χρησιμοποιήθηκε ένα (1) παράδειγμα από το Set E (Christophides and Elion, 1969) με 76 κόμβους και επτά (7) παραδείγματα από τις 14 βάσεις δεδομένων (benchmark problems) των Christofides, Mingozzi και Toth (1979). Η επίλυση του κάθε προβλήματος εκτελέστηκε 10 φορές και καταγράφηκε η μέση τιμή, καθώς και η καλύτερη λύση που βρέθηκε στον πίνακα 5.3.1.

5.2.1 Πρόβλημα **E-n076-k07** :

Number of Nodes	76
Vehicle Capacity	220
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.2.1 *Δεδομένα προβλήματος* ***E-n076-k07***

Best solution cost found: 687,6025



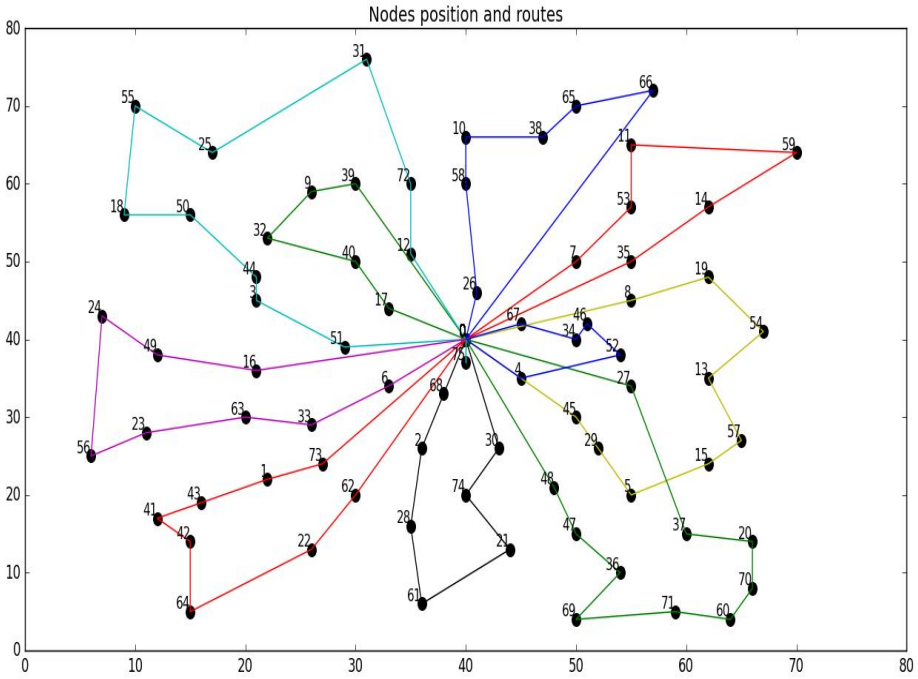
Γράφημα 5.2.1 *Λύση προβλήματος* ***E-n076-k07*** *στο επίπεδο xy*

5.2.2 Πρόβλημα **CMT02** :

Number of Nodes	76
Vehicle Capacity	140
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.2.2 *Δεδομένα* ***CMT02***

Best solution cost found: 837,4514



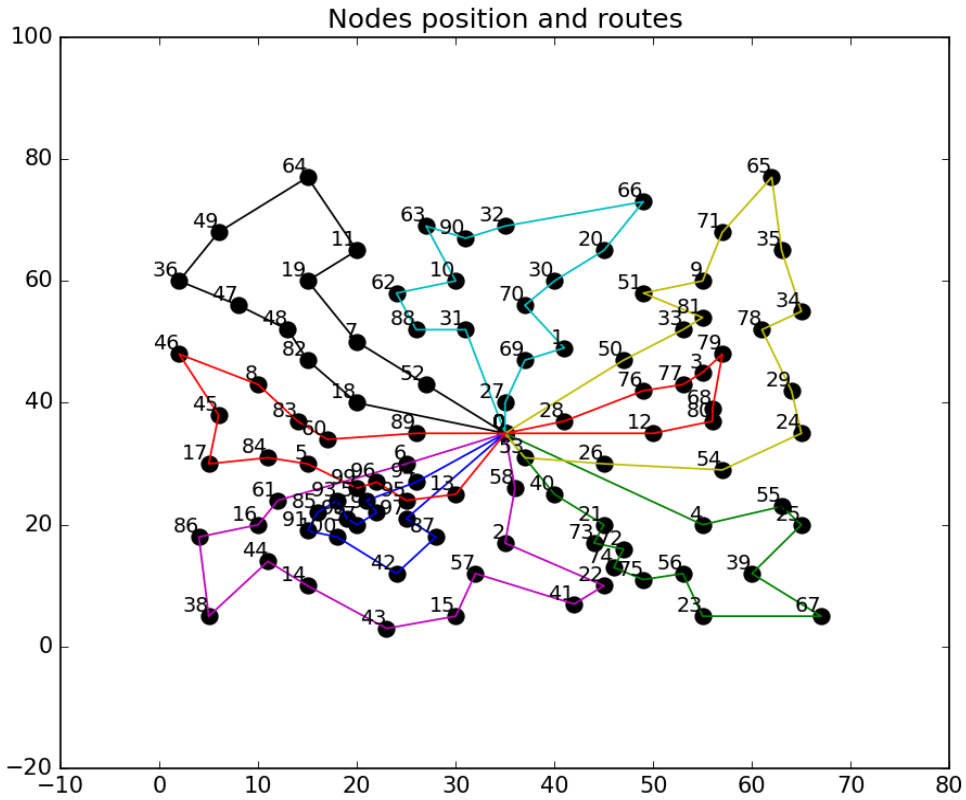
Γράφημα 5.2.2 *Λύση προβλήματος* ***CMT02*** *στο επίπεδο xy*

5.2.3 Πρόβλημα **CMT03** :

Number of Nodes	101
Vehicle Capacity	200
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.2.3 *Δεδομένα* ***CMT03***

Best solution cost found: 835,0497



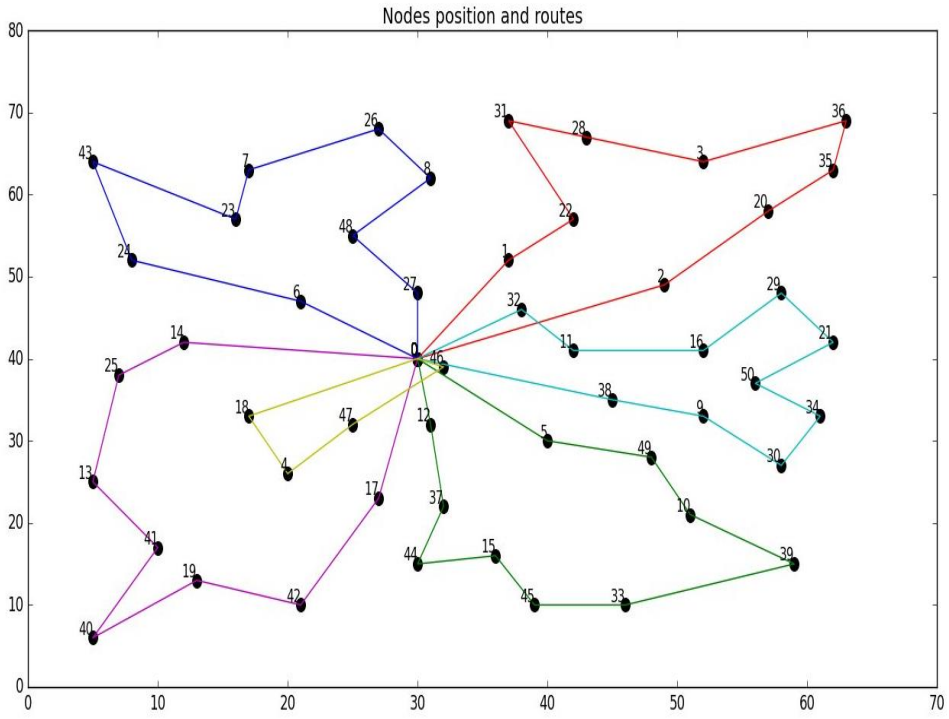
Γράφημα *5.2.3 Λύση προβλήματος* ***CMT03*** *στο επίπεδο xy*

5.2.4 Πρόβλημα **CMT06** :

Number of Nodes	51
Vehicle Capacity	160
Maximum Route Time	200
Service Time	10
Number of Depots	1

Πίνακας 5.2.4 *Δεδομένα προβλήματος* ***CMT06***

Best solution cost found: 555,8237



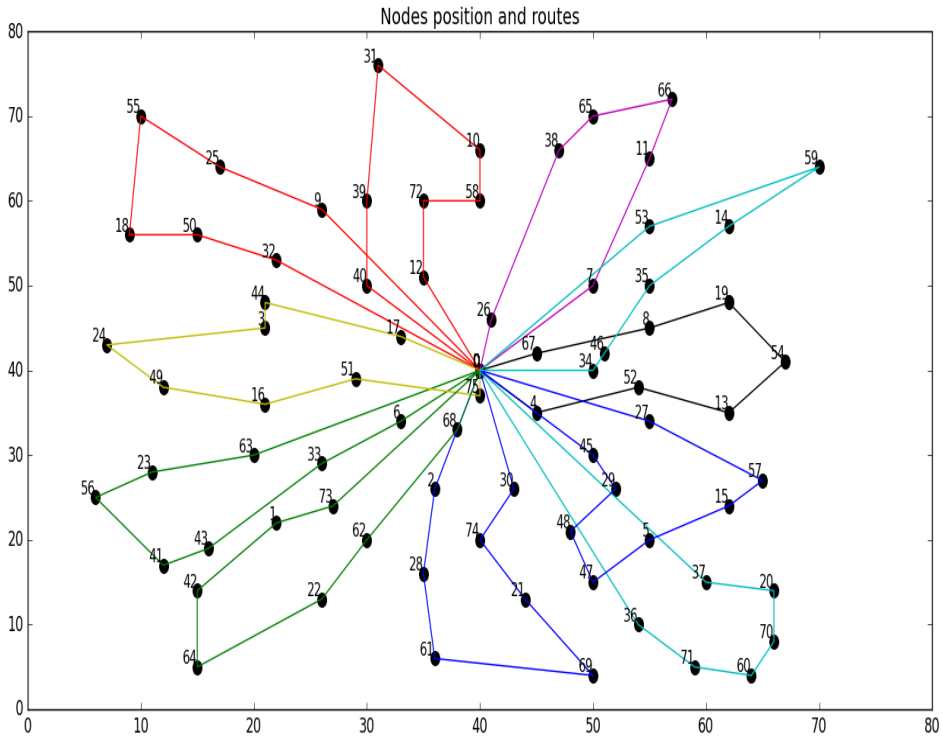
Γράφημα 5.2.4 *Λύση προβλήματος* ***CMT06*** *στο επίπεδο xy*

5.2.5 Πρόβλημα **CMT07** :

Number of Nodes	76
Vehicle Capacity	140
Maximum Route Time	160
Service Time	10
Number of Depots	1

Πίνακας 5.2.5 *Δεδομένα προβλήματος* ***CMT07***

Best solution cost found: 913,2314



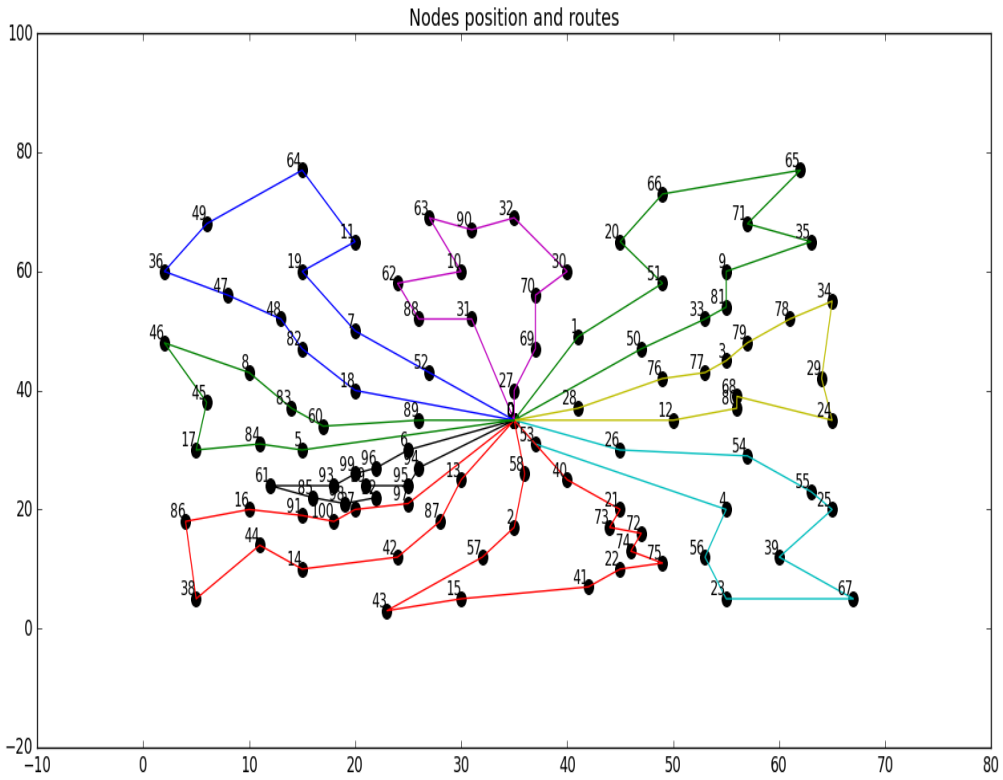
Γράφημα 5.2.5 *Λύση προβλήματος* ***CMT07*** *στο επίπεδο xy*

5.2.6 Πρόβλημα **CMT08** :

Number of Nodes	101
Vehicle Capacity	200
Maximum Route Time	230
Service Time	10
Number of Depots	1

Πίνακας 5.2.6 *Δεδομένα προβλήματος* ***CMT08***

Best solution cost found: 870,8606



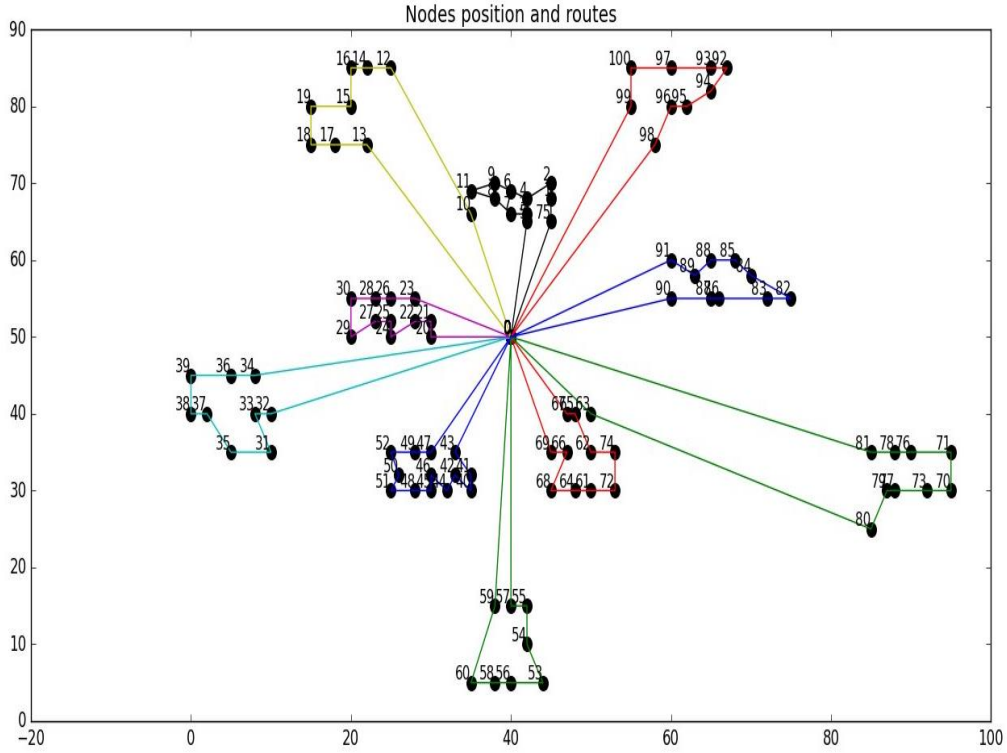
Γράφημα 5.2.6 *Λύση προβλήματος* ***CMT08*** *στο επίπεδο xy*

5.2.7 Πρόβλημα **CMT12** :

Number of Nodes	101
Vehicle Capacity	200
Maximum Route Time	∞
Service Time	0
Number of Depots	1

Πίνακας 5.2.7 *Δεδομένα προβλήματος* ***CMT08***

Best solution cost found: 825,855



Γράφημα 5.2.7 *Λύση προβλήματος* ***CMT12*** *στο επίπεδο xy*

5.3 Πίνακας αποτελεσμάτων

Πρόβλημα	Αριθμός κόμβων	Χωρητικότητα	Μέγιστος χρόνος διαδρομής	Μέση τιμή	Καλύτερη λύση	Κατώτερο Όριο	Απόκλιση
SET E (CHRISTOFIDES AND ELION, 1969)							
E-n22-k04	22	6000	∞	375,2798	**375,2798**	**375,2798**	0,00%
E-n33-k04	33	8000	∞	837,6717	**837,6717**	**837,6717**	0,00%
E-n76-k07	76	220	∞	694,7966	**687,6025**	**687,6025**	0,00%
CHRISTOFIDES MINGOZZI AND TOTH (1979)							
CMT01	51	160	∞	525,7864	**524,6113**	**524,6113**	0,00%
CMT02	76	140	∞	851,2324	837,4514	835,26	0,26%
CMT03	101	200	∞	839,178	835,0497	826,14	1,08%
CMT06	51	160	200	562,989	555,8237	555,43	0,07%
CMT07	76	140	160	924,1738	913,2364	909,68	0,39%
CMT08	101	200	230	882,7842	870,8606	865,94	0,57%
CMT12	101	200	∞	843,37024	825,855	819,56	0,77%

Πίνακας 5.3 *Αποτελέσματα HBA*

Συμπεράσματα

Ο HBA αποτελεί έναν μεθευρετικό αλγόριθμο που επιλύει το πρόβλημα δρομολόγησης οχημάτων στοχαστικά. Με άλλα λόγια, εξαιτίας των τυχαιοποιημένων μεθόδων που χρησιμοποιεί δεν εγγυάται ότι θα παράγει σταθερά τις ίδιες λύσεις για ένα πρόβλημα. Ωστόσο αποδεικνύεται ότι ο αλγόριθμος καταφέρνει να αυξάνει την πιθανότητα αποφυγής τοπικών ελαχίστων και καταφέρνει να προσεγγίζει σε μεγάλο βαθμό το ολικό ελάχιστο. Ο HBA κατάφερε να αποφέρει ικανοποιητικά αποτελέσματα επιλύοντας προβλήματα δρομολόγησης οχημάτων για 22 έως 101 κόμβους. Η απόκλιση από τις καλύτερες γνωστές λύσεις ή τα κατώτερα όρια, όπως αναγράφονται στον πίνακα 5.3, αναδείχθηκε αρκετά ικανοποιητική με τιμές ποσοστού 1% και μικρότερες. Στα προβλήματα του set E 22 και 33 κόμβων επιτεύχθηκε η εύρεση της βέλτιστης λύσης σε σύντομο χρονικό διάστημα (μικρότερο του ενός λεπτού). Επιπλέον ικανοποιητική ήταν η απόδοση του προγράμματος κατά την επίλυση του CMT01, όπου η καλύτερη γνωστή λύση βρέθηκε με μεγάλη συχνότητα (πίνακας 5.1.7) και σε ικανοποιητικό υπολογιστικό χρόνο. Σημαντική είναι επίσης και η απόδοση του HBA για τα πιο απαιτητικά προβλήματα 76 και 101 κόμβων, καθώς το κατώτερο όριο προσεγγίστηκε σε μεγάλο βαθμό με επιτυχή επίσης εύρεση της λύσης για το E-n76-k07. Ο υπολογιστικός χρόνος ωστόσο αυξήθηκε σημαντικά, κατά την επίλυση αυτών των προβλημάτων, διότι όσο αυξάνεται η πολυπλοκότητα του προβλήματος ανάλογα επεκτείνεται και ο χώρος λύσεων. Για την επίλυση των μεγάλων προβλημάτων και με σκοπό τη μείωση του υπολογιστικού χρόνου, θα μπορούσε να επιλεχθεί μικρότερος αριθμός πληθυσμού, αλλά τότε δεν υπάρχει εγγύηση ότι τα αποτελέσματα που θα βρεθούν θε είναι παρεμφερή με αυτά που εμφανίζονται στον πίνακα 5.3.

Βιβλιογραφία

[1]Larson, P., & Rogers, D. (1998*). Supply Chain Management: Definition, Growth and Approaches. Journal of Marketing Theory and Practice*, 6, 1–5.

[2]Cooper, M., & Ellram, L. (1993*). Characteristics of Supply Chain Management & the Implications for Purchasing & Logistics Strategy. International Journal of Logistics Management*, The, 4, 13–24.

[3]Christopher, Martin. (2005). *Logistics and supply chain management : creating value-added networks. FT Prentice Hall*.

[4]Laporte, G. (1992*). The Traveling Salesman Problem: An overview of exact and approximate algorithms. In European Journal of Operational Research* (Vol. 59).

[5]Knuth, D. E. (1974). *Postscript about NP-hard problems. SIGACT News*, 6(2), 15–16.

[6]Fisher, M. L., & Jaikumar, R. (1981). *A generalized assignment heuristic for vehicle routing. Networks*, 11(2), 109–124.

[7]Goel, A., & Gruhn, V. (2006). *Drivers' working hours in vehicle routing and scheduling. In Transportation Science* (Vol. 43).

[8]Yang, X.-S. (2010). *A New Metaheuristic Bat-Inspired Algorithm. in: Nature Inspired Coop-erative Strategies for Optimization*. 284, pp.65–74.

[9]Layeb, A., Ammi, M., & Chikhi, S. (2013). *A GRASP algorithm based on new randomized heuristic for vehicle routing problem. Journal of Computing and Information Technology*, 21(1), 35–46. https://doi.org/10.2498/cit.1002085

[10]Hansen, P., & Mladenović, N. (2016). *Variable Neighborhood Search. In R. Martí, P. Panos, & M. G. C. Resende (Eds.), Handbook of Heuristics* (pp. 1–29). *Springer International*

[11]Μαρινάκης, Ι., Μαρινάκη, Μ. and Μυγδαλάς, Α. (2019). *Προβλήματα Δρομολόγησης Οχημάτων στη Διαχείρηση της Εφοδιαστικής Αλυσίδας. Μοντελοποίηση και Αλγόριθμοι Επίλυσης*. Αθήνα: Εκδόσεις νέων τεχνολογιών.

[12]Σαρτζετάκη Κ. (2013), Logistics και Εφοδιαστική Αλυσίδα σε μια επιχείρηση, Πτυχιακή Εργασία, Επιβλέπων Καθηγήτρια: Γιαννακόπουλου Ελένη, Τμήμα Λογιστικής, Τ.Ε.Ι. Κρήτης

[14]Lummus, R.R. and Vokurka, R.J. (1999). Defining supply chain management: a historical perspective and practical guidelines. *Industrial Management & Data Systems*, [online] 99(1), pp.11–17.

[15]Marinakis, Y. and Marinaki, M. (2010). A hybrid genetic – Particle Swarm Optimization Algorithm for the vehicle routing problem. *Expert Systems with Applications*, [online] 37(2), pp.1446–1455.

[16]Agarwal, P., Zhou, Y., Xie, J. and Zheng, H. (2013). A Hybrid Bat Algorithm with Path Relinking for Capacitated Vehicle Routing Problem. *Mathematical Problems in Engineering*, [online] 2013, p.392789.

[17]Taha, Anass, Mohamed Hachimi, and Ali Moudden. "Adapted bat algorithm for capacitated vehicle routing problem." *International Review on Computers and Software (IRECOS)* 10.6 (2015): 610-619.

[18]Marinakis, Y. (2012). Multiple Phase Neighborhood SearchGRASP for the Capacitated Vehicle Routing Problem. *Expert Systems with Applications*, [online] 39(8), pp.6807–6815.

[19]Kassem, Sally, et al. "A hybrid bat algorithm to solve the capacitated vehicle routing problem." 2019 Novel Intelligent and Leading Emerging Sciences Conference (NILES). Vol. 1. IEEE, 2019.

[20]Yang, X. (2013). Bat Algorithm: Literature Review and Applications. *International Journal of BioInspired Computation*, 5.

[21]Yu Li, Qian Guo, and Jingsen Liu. Improved Bat Algorithm for Vehicle Routing Problem [J]. Int J Performability Eng, 2019, 15(1): 317-325.

[22]Osaba, E., Carballedo, R., Yang, X., jr, F., López García, Pedro and Ser, D. (2018). On Efficiently Solving the Vehicle Routing Problem with Time Windows Using the Bat Algorithm with Random Reinsertion Operators. In: *Studies in Computational Intelligence*. pp.69–89.

[23] Zhou, Y., Luo, Q., Chen, H., He, A., & Wu, J. (2015). A discrete invasive weed optimization algorithm for solving traveling salesman problem. Neurocomputing, 151(P3), 1227–1236.