ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΛΕΤΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΥΡΕΤΙΚΩΝ ΜΕΘΟΔΩΝ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΠΕΡΙΟΡΙΣΜΟ ΧΩΡΗΤΙΚΟΤΗΤΑΣ

Διπλωματική Εργασία

του

Αρβανίτη Συμεών

Θεσσαλονίκη , 06/2019

ΜΕΛΕΤΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΥΡΕΤΙΚΩΝ ΜΕΘΟΔΩΝ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΠΕΡΙΟΡΙΣΜΟ ΧΩΡΗΤΙΚΟΤΗΤΑΣ

Αρβανίτης Συμεών

Πτυχίο Μαθηματικών, Πανεπιστήμιο Ιωαννίνων, 2013

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Σιφαλέρας Άγγελος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25/06/2019

Σιφαλέρας Άγγελος	Σακελλαρίου Ηλίας	Μαντάς Μιχαήλ
.....	.....	.....

Αρβανίτης Συμεών

.....

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την μελέτη και την υλοποίηση ευρετικών μεθόδων κατασκευής και βελτιστοποίησης για το πρόβλημα δρομολόγησης οχημάτων με περιορισμούς χωρητικότητας. Στην πρώτη ενότητα αναλύεται η σημαντικότητα του θέματος καθώς επίσης και οι διάφορες παραλλαγές του προβλήματος.

Στη συνέχεια εξετάζονται οι αλγόριθμοι κατασκευής αρχικής λύσης του εγγύτερου γείτονα, ο αλγόριθμος σάρωσης και ο αλγόριθμος εξοικονόμησης των Clarke και Wright. Έπειτα αναλύονται οι αλγόριθμοι βελτιστοποίησης μια αρχικής λύσης 2-opt,3-opt,Swap,Relocate. Στις επόμενες ενότητες αναλύονται οι πιο δημοφιλείς μεθευρετικές μέθοδοι και δίνεται έμφαση στην αναζήτηση μεταβλητής γειτνίασης (Variable Neighborhood Search - VNS) πάνω στην οποία βασίζεται η υλοποίηση των αλγορίθμων.

Τέλος παρουσιάζονται τα υπολογιστικά αποτελέσματα των αλγορίθμων κατασκευής (σφάλμα, χρόνος εκτέλεσης) και της μεθόδου VNS για μια πληθώρα στιγμιοτύπων της TSP-Lib.

Λέξεις Κλειδιά: Συνδυαστική βελτιστοποίηση, Ευρετικές μέθοδοι, Μεθευρετικές μέθοδοι, Ευρετικές μέθοδοι κατασκευής, Προβλημα δρομολόγησης οχημάτων, Αναζήτηση μεταβλητής γειτνίασης, VNS

Abstract

This diploma thesis deals with the study and implementation of heuristic methods of construction and optimization for the problem of routing vehicles with capacity constraints. The first section analyzes the importance of the subject as well as the most known variants of the problem. Then the nearest neighbor algorithm, the sweep algorithm, and the savings algorithm of Clarke and Wright are examined for the construction of an initial solution. Then the optimization algorithms for an initial solutions 2-opt, 3-opt, Swap, Relocate are presented and analyzed. The following sections analyze the most popular methods and emphasize the Variable Neighborhood Search (VNS) on which the implementation of algorithms is based. Finally, the computational results of the construction algorithms (error, execution time) and the VNS method for a variety of TSP-Lib snapshots are presented.

Keywords: Logistics Optimization, Combinatorial optimization, Metaheuristics, Capacitated Vehicle Routing Problem, Construction heuristics, Variable Neighborhood Search, VNS, Heuristic methods

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον κο Σιφαλέρα Άγγελο για την ευκαιρία που μου έδωσε, για την καθοδήγηση του κατά διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας καθώς επίσης και για τις πολύτιμες συμβουλές του.

Επίσης θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την στήριξη που μου παρείχαν, καθώς επίσης για την υπομονή και την πίστη τους στο πρόσωπο μου.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

ΚΕΦΑΛΑΙΟ 1

Το πρόβλημα δρομολόγησης οχημάτων

1.1 Εισαγωγή

Ο επιστημονικός κλάδος των logistics περιλαμβάνει την αποτελεσματική διαχείριση ενός συνόλου οχημάτων με σκοπό να εξυπηρετηθούν οι αποθήκες, οι έμποροι και οι τελικοί καταναλωτές που χρησιμοποιούν ένα δίκτυο μεταφορών. Το επιθυμητό απότελεσμα είναι να παρθούν οι βέλτιστες αποφάσεις οι οποίες θα ελαχιστοποιήσουν το κόστος λειτουργίας και συντήρησης αυτού του δικτύου μεταφορών. Οι αποφάσεις αυτές περιλαμβάνουν τους τρόπους δρομολόγησης των οχημάτων, τους τρόπους διαχείρησης των αποθηκών, του καταμερισμού των εργασιών και πολλών ακόμα παραγόντων που επηρεάζουν σημαντικά τα λειτουργικά έξοδα μιας επιχείρησης (CHRISTOPHER 1999).

Έτσι προκειμένου να αποφευχθούν περιττά έξοδα έχουν αναπτυχθεί αρκετές μέθοδοι για την λύση αυτών των προβλημάτων όμως η εύρεση της βέλτιστης λύσης είναι υπολογιστικά δύσκολη για τα περισσότερα προβλήματα βελτιστοποίησης στην βιομηχανία και στην ακαδημαϊκή κοινότητα. Δουλεύοντας σε πραγματικές συνθήκες, δηλαδή με μια πληθώρα συνδυαστικών προβλημάτων και περιορισμών, οι διαχειριστές τέτοιων δικτύων είναι πολλές φορές ικανοποιημένοι με ‘‘καλά’’ αποτελέσματα, τα οποία πλησιάζουν όσο είναι δυνατόν τις βέλτιστες λύσεις (Pearl 1984).

Οι προσεγγιστικές αυτές λύσεις προέρχονται από ευρετικούς ή μεθευρετικούς αλγορίθμους που ο σκοπός τους είναι να βρουν μια σχετικά ικανοποιητική λύση μέσα σε ένα δεδομένο χρονικό περιθώριο (Osman and Laporte 1996). Οι μεθευρετικοί αλγόριθμοι είναι ένα σύνολο αλγορίθμων βελτιστοποίησης που έγινε ιδιαίτερα δημοφιλές κατά τις τελευταίες δεκαετίες. Είναι μία από τις πιο επιτυχημένες τεχνικές αντιμετώπισης τέτοιων προβλημάτων. Οι μεθευρετικοί αλγόριθμοι παρέχουν λύσεις με σχετικά μικρή απόκλιση απο το βέλτιστο σε ένα σύντομο δεδομένο χρονικό διάστημα για την επίλυση δύσκολων και πολύπλοκων προβλημάτων στον τομέα της επιστήμης και της μηχανικής. Έτσι το βάρος της επιστημονικής έρευνας έπεσε πάνω σε αυτές τις τεχνικές και αυτό εξηγεί την ανάπτυξη και το στο συγκεκριμένο τομέα.

1.2 Η βασική μορφή του VRP

Το πρόβλημα δρομολόγησης οχημάτων (VRP) είναι ένα από τα πιο γνωστά προβλήματα συνδυαστικής βελτιστοποίησης και ακέραιου προγραμματισμού. Πρόκειται δηλαδή για μια ειδική περίπτωση προβήματος γραμμικού προγραμματισμού στο οποίο όμως οι τιμές των μεταβλητών είναι αριθμοί ακέραιοι και όχι πραγματικοί. Πραγματεύεται την εύρεση του βέλτιστου συνόλου διαδρομών που θα εκτελέσει ένας στόλος οχημάτων με σκοπό την επίσκεψη όλων των κόμβων μέσα σε ένα δίκτυο (Toth and Vigo 2002).

Πρωτοεμφανίστηκε σε μια ερευνητική εργασία των George Dantzig και John Ramster (Dantzig and Ramser 1959) το 1959 και αποτελεί τη γενίκευση του πασίγνωστου προβλήματος του περιοδεύοντος πωλητή (TSP) αφού στο TSP έχουμε μόνο ένα όχημα. Τις τελευταίες δεκαετίες έχει αυξηθεί κατακόρυφα το ακαδημαϊκό ενδιαφέρον για το προβλήμα δρομολόγησης οχημάτων και για τις διάφορες παραλλαγές του. Αυτό συμβαίνει διότι εκτός από επιστημονικό και ερευνητικό ενδιαφέρον, η επίλυση του προβλήματος VRP έχει προφανή οφέλη για τη βιομηχανία. Η μεταφορά ενός προϊόντος είναι ένας σημαντικός παράγοντας αύξησης του κόστους του και υπολογίζεται πως η χρησιμοποίηση προγραμμάτων βελτιστοποίησης μπορούν να μειώσουν το κόστος διανομής από 5 έως και 20%.

Η βασική δυσκολία στην επίλυση είναι πως το VRP είναι ένα πρόβλημα το οποίο ανήκει στην κλάση πολυπλοκότητας NP-hard. Τα προβλήματα αυτής της κλάσης είναι δύσκολο να επιλυθούν αφού οι ακριβείς αλγόριθμοι που είναι σε θέση να τα λύσουν έχουν πολύ μεγάλες πολυπλοκότητες όσον αφορά τον αριθμό των πράξεων που απαιτούνται. Αυτό σημαίνει όσο αυξάνουμε το μέγεθος του προβλήματος, τα βήματα που απαιτούνται για την επίλυση του αυξάνονται με ρυθμό όχι πολυωνυμικό αλλά εκθετικό ή παραγοντικό. Τα προβλήματα της κλάσης NP-hard είναι τουλάχιστον όσο δύσκολα όσο τα δυσκολότερα της κλάσης NP. Η κλάση NP είναι μία κλάση πολυπλοκότητας που χρησιμοποιείται για να κατηγοριοποιήσει προβλήματα απόφασης τα οποία έχουν ως απάντηση NAI ή OXI (Knuth 1974). Πρόκειται δηλαδή για το σύνολο των προβλημάτων που είναι πολυωνυμικά επαληθεύσιμα. Αυτό σημαίνει πως αν σε ένα πρόβλημα έχουμε μια υποψήφια λύση μπορούμε σε πολυωνυμικό χρόνο να επαληθεύσουμε το αν αυτή η λύση είναι σωστή. Σε αντίθεση με την κλάση P στην οποία ανήκουν τα προβλήματα που επιλύονται σε πολυωνυμικό χρόνο, στην κλάση NP μπορούμε μόνο να επαληθεύσουμε την ορθότητα μιας λύσης. Είναι ακόμα ανοιχτό το πρόβλημα της σχέσης μεταξύ των 2 αυτών κλάσεων (Cook 2006).

Μέχρι σήμερα οι ειδικοί της θεωρητικής επιστήμης της πληροφορικής και της ανάλυσης αλγορίθμων και της πολυπλοκότητας αυτών έχουν καταλήξει στο ότι η κλάση P αποτελεί υποσύνολο της κλάσης NP αφού ένα πρόβλημα πολυωνυμικά επιλύσιμο θα είναι σίγουρα και πολυωνυμικά επαληθεύσιμο. Το ερώτημα που παραμένει όμως είναι το αν οι 2 κλάσεις είναι ίσες ή αν η μια είναι γνήσιο υποσύνολο της άλλης. Παρόλο που οι περισσότεροι ερευνητές του πεδίου αυτού πιστεύουν πως οι κλάσεις P και NP δεν είναι ίσες και συνεπώς η κλάση P ανήκει εξ΄ ολοκλήρου στην κλάση

NP, ακόμη δεν έχει διατυπωθεί μια αυστηρή μαθηματική απόδειξη που να επαληθεύει τον ισχυρισμό αυτό (Fortnow 2009).

Έτσι για να μπορούμε να ανταπεξέλθουμε στις απαιτήσεις των προβλημάτων πρέπει να έχουμε στη διάθεση μας ταράστια υπολογιστική ισχύ πράγμα που είναι πρακτικά ανέφικτο. Σήμερα λόγω της ραγδαίας τεχνολογικής προόδου των τελευταίων δεκαετιών μπορούμε να εφαρμόσουμε ακριβείς αλγορίθμους για την επίλυση δύσκολων υπολογιστικών προβλημάτων όπως το VRP όμως αυτό είναι εφικτό μόνο έως ένα όριο διάστασης του προβλήματος. Σε μεγάλου μεγέθους στιγμιότυπα είμαστε αναγκασμένοι να καταφύγουμε σε άλλες μεθόδους με σκοπό την προσέγγιση της λύσης. Στα επόμενα κεφάλαια θα αναλυθούν εκτενέστερα κάποιες από αυτές τις μεθόδους.

Η κεντρική ιδέα του προβλήματος είναι ότι τα οχήματα ξεκινάνε από μια κεντρική αποθήκη και το καθένα από αυτα εκτελεί μια διαδρομή εξυπηρετώντας κάποιους από τους πελάτες του δικτύου οι οποίοι έχουν παραγγείλει κάποια αγαθά. Ο στόχος του VRP είναι να ελαχιστοποιηθεί η συνολική απόσταση που θα διανύσουν τα οχήματα. Οι βασικοί περιορισμοί του προβλήματος είναι ότι έχουμε μια και μοναδική αποθήκη, ότι όλοι οι κόμβοι είναι κόμβοι ζήτησης και ότι πρέπει να εξυπηρετηθούν όλοι από ένα και μοναδικό όχημα. Φυσικά η ποσότητα ζήτησης του κάθε κόμβου καθώς και η ακριβής θέση του είναι γνωστά από την αρχή και αποτελούν τα βασικά δεδομένα του προβήματος (Golden, Raghavan, and Wasil 2008). Αυτή είναι η βασική μορφή του προβλήματος από την οποία προκύπτουν, με μερικές τροποποιήσεις των περιορισμών, διάφορες παραλλαγές του προβλήματος. Οι πιο γνωστές από αυτές θα εξεταστούν στη συνέχεια. Το πρόβλημα μοντελοποιείται με τη βοήθεια μαθηματικών σχέσεων για να εκφράζουμε τους περιορισμούς αλλά και το ζητούμενο του προβληματος. Οι μαθηματικές εκφράσεις του VRP είναι οι εξής:

$$min \sum_{i,j} c_{i,j} \sum_{k} x_{i,j,k} \tag{1}$$

$$x_{i,j,k} = \begin{cases} 1 & \text{αν οι κόμβοι } i,j \text{ είναι συνεχόμενοι στη διαδρομή του οχήματος } k \\ 0 & \text{διαφορετικά} \end{cases} \tag{2}$$

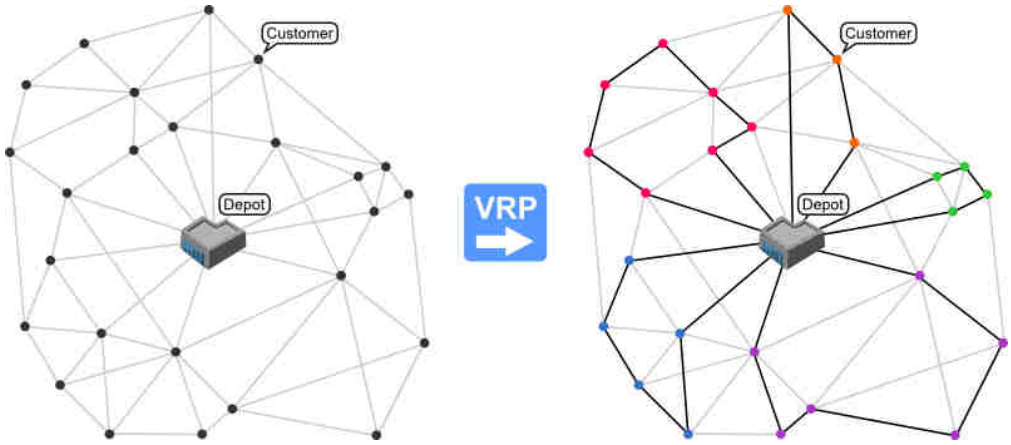
$$y_{i,k} = \begin{cases} 1 & \text{αν ο κόμβος } i \text{ εξυπηρετείται από το όχημα } k \\ 0 & \text{διαφορετικά} \end{cases} \tag{3}$$

$$\sum_{k} y_{ik} = 1 \quad k = 1, 2...m \tag{4}$$

$$\sum_j x_{ijk} = \sum_j x_{ijk} y_{ik} \quad i = 1, 2, ..., n \quad k = 1, 2, ..., m \tag{5}$$

$$\sum_j x_{ijk} \leq |s| - 1 \quad \forall k \in \{1...m\} \quad \forall s \in \{2...n\} \tag{6}$$

Η εξίσωση 1 εκφράζει την αντικειμενική συνάρτηση του κόστους την οποία έχουμε σαν στόχο να ελαχιστοποιήσουμε, ενώ οι υπόλοιπες εκφράζουν τους περιορισμούς του προβήματος. Η εξίσωση 2 μας δείχνει αν δύο κόμβοι i,j είναι συνεχόμενοι στη διαδρομή ενός οχήματος ή όχι. Η εξίσωση 3 δείχνει αν ο κόμβος i εξυπηρετείται από το όχημα k ή όχι και η εξίσωση 4 συμπληρώνει τον περιορισμό εκφράζοντας το ότι ο κάθε κόμβος εξυπηρετείται από ένα και μοναδικό όχημα. Η εξίσωση 5 δείχνει ότι ένα όχημα που επισκέπτεται έναν κόμβο, στη συνέχεια φεύγει από τον κόμβο αυτόν. Τέλος η εξίσωση 6 δείχνει τον άριθμό των όχημάτων. Ακολουθεί το σχήμα **1.2.1** με το στιγμιότυπο ενός VRP και την αναπαράσταση της λύση του (Πηγή εικόνας: `http://neo.lcc.uma.es/vrp/vehicle-routing-problem`).



Σχήμα **1.2.1** Στιγμιότυπο ενός VRP και η αναπαράσταση της λύση του

1.3 VRP με περιορισμούς χωρητικότητας

Μια από τις πιο γνωστές παραλλαγές του βασικού VRP είναι η παραλλαγή του με τους περιορισμούς χωρητικότητας. Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την αποτελεσματική επίλυση της συγκεκριμένης παραλλαγής του VRP. Οι μέθοδολογίες που μελετήθηκαν και υλοποιήθηκαν αναλύονται εκτενέστερα σε επόμενα κεφάλαια. Στο CVRP-(Capacitated vehicle routing problem) κάθε κόμβος ζήτησης χαρακτηρίζεται από έναν συγκεκριμένο και γνωστό από την αρχή

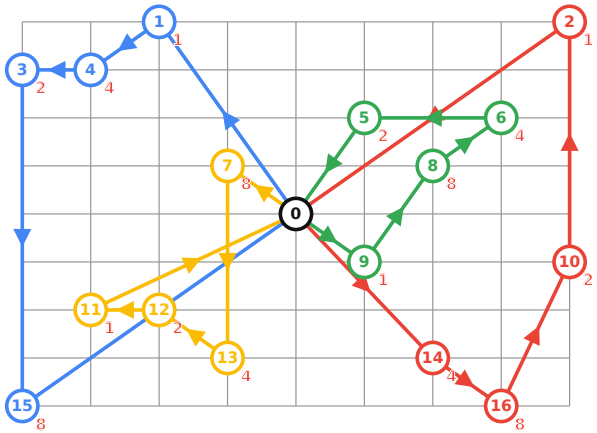
αριθμό ο οποίος ποσοτικοποιεί τη ζήτηση. Με άλλα λόγια ο αριθμός αυτός εκφράζει το πόσα αγαθά ζητάει ένας κόμβος. Παράλληλα τα οχήματα χαρακτηρίζονται από έναν αριθμό που εκφράζει την χωρητικότητα τους σε αγαθά.

Όλα τα οχήματα που έχουμε στη διάθεσή μας έχουν την ίδια χωρητικότητα και έτσι είναι λογικό πως ένα όχημα δεν μπορεί να εξυπηρετήσει απεριόριστους κόμβους ζήτησης παρά μόνο όσους του επιτρέπει η χωρητικότητα του. Εδώ πρέπει να επισημανθεί ότι ένα στιγμιότυπο CVRP είναι επιλύσιμο μόνο εάν η χωρητικότητα των οχημάτων, η οποία είναι ίδια για όλα τα οχήματα, είναι μεγαλύτερη ή ίση από τη ζήτηση κάθε κόμβου ξεχωριστά. Αυτό συμβαίνει διότι δεν επιτρέπεται ένας κόμβος ζήτησης να εξυπηρετηθεί από δύο ή παραπάνω οχήματα. Συνεπώς αν ένας κόμβος έχει ζήτηση παραπάνω από τη χωρητικότητα του οχήματος τότε το πρόβλημα δεν είναι επιλύσιμο (Ralphs, Kopman, Pulleyblank, and Trotter 2003).

Ακριβώς όπως και στην απλή εκδοχή του VRP έτσι και εδώ, ο στόχος είναι η ελαχιστοποίηση της αντιεκιμενικής συνάρτησης κόστους η οποία μας δείχνει την συνολική απόσταση που διανύουν τα οχήματα σε μία λύση του προβήματος. Όσον αφορά την μαθηματική μοντελοποίηση του προβλήματος αυτή είναι η ίδια με την μοντελοποίηση του απλού VRP προσθέτοντας τον εξής περιορισμό:

$$\sum_{i=0}^{n} d_i \leq C \tag{7}$$

Ο περιορισμός αυτός δηλώνει πως σε μια διαδρομή το σύνολο της ζήτησης των κόμβων που περιλαμβάνονται στη διαδρομή αυτή δεν πρέπει να υπερβαίνει την χωρητικότητα του οχήματος. Είναι λοιπόν φανερό πως σε αντίθεση με το VRP δεν είναι όλες οι λύσεις εφικτές ούτε μπορούμε να κατασκευάσουμε μια εντελώς τυχαία εφικτή λύση. Παρακάτω παρατίθεται ένα στιγμιότυπο CVRP (Σχήμα **1.3.2**) 16 κόμβων ζήτησης με χωρητικότητα οχημάτων 15, μαζί με την αναπαράσταση μίας εφικτής λύσης (Πηγή εικόνας: `https://developers.google.com/optimization/routing/cvrp`).



Σχήμα **1.3.2** Ένα στιγμιότυπο CVRP και η λύση του

1.4 Το πρόβλημα του πλανόδιου πωλητή

Το πρόβλημα του πλανόδιου πωλητή Travelling salesman problem - TSP αποτελεί την παλιότερη και ίσως βασικότερη μορφή της οικογένειας των προβλημάτων δρομολόγησης. Στην ουσία πραγματεύεται την εύρεση της βέλτιστης σειράς με την οποία πρέπει να επισκεφθούμε ένα σύνολο κόμβων (Gavish and Graves 1978). Οι πρώτες αναφορές στο πρόβλημα αυτό έγιναν τον 19ο αιώνα από τον διάσημο μαθηματικό William Hamilton ο οποίος εξέταση το πρόβλημα της ύπαρξης μιας κλειστής διαδρομής μέσα σε έναν γράφο περνώντας από κάθε κορυφή ακριβώς μια φορά. Μια τέτοια διαδρομή ονομάζεται κύκλος Hamilton. Το πρόβλημα της ελαχιστοποίησης του κύκλου Hamilton έγινε διάσημο περίπου έναν αιώνα αργότερα όταν παρουσιάστηκαν τα προφανή οικονομικά οφέλη από την αποτελεσματική δρομολόγηση οχημάτων.

Με τους νόμους της συνδυαστικής εύκολα μπορούμε να υπολογίσουμε ότι το πλήθος των κύκλων Hamilton σε έναν πλήρη γράφο (δηλαδή έναν γράφο στον οποίο κάθε κορυφή συνδέεται με όλες τις υπόλοιπες) με n κορυφές είναι $(n-1)!/2$. Αυτό σημαίνει ότι ακόμα και σε αυτή την κατηγορία προβλημάτων που θεωρείται από τις πιο απλές τόσο σε πολυπλοκότητα όσο και σε μαθηματική μοντελοποίηση, οι πιθανές λύσεις αυξάνονται με πολύ μεγάλο ρυθμό όσο αυξάνεται το μέγεθος του γράφου (Papadimitriou 1977). Έτσι καθίσταται δύσκολη η εύρεση της συντομότερης διαδρομής ακόμα και με ακριβείς αλγορίθμους οι οποίοι απαιτούν μεγάλη υπολογιστική ισχύ και αρκετό χρόνο εκτέλεσης για να εντοπίσουν τη βέλτιστη διαδρομή. Σήμερα με τη βοήθεια ευρετικών μεθόδων μπορούμε να φτάσουμε γρήγορα σε μια καλή λύση του προβλήματος ακόμα και σε πολύ μεγάλου μεγέθους στιγμιότυπα με χιλιάδες κορυφές, και μάλιστα με πολύ μικρή απόκλιση της τάξης του 2-3% από το πραγματικό βέλτιστο.

Η μαθηματική μοντελοποίση του TSP όπως και στα προηγούμενα παραδείγματα περιλαμβάνει την ελαχιστοποίηση μιας αντικειμενικής συνάρτησης υπό κάποιους περιορισμούς.

$$min \sum_{i,j \in E} c_{i,j} x_{i,j} \tag{8}$$

Υπό τους περιορισμούς:

$$x_{i,j} = \begin{cases} 1 & \text{αν οι κόμβοι } i,j \text{ είναι συνεχόμενοι στη διαδρομή} \\ 0 & \text{διαφορετικά} \end{cases} \tag{9}$$

$$\sum_{j=1}^{n} x_{i,j} = 2 \quad \forall k = 1,2,...,n \tag{10}$$

$$\sum_{j=1}^{n} x_{1,j} = 2 \tag{11}$$

$$\sum_{i,j \in E} x_{i,j} = 2 \tag{12}$$

$$X(C) \leq |C| - 1 \quad \text{για όλους τους κύκλους} c \in 2, ..., n \tag{13}$$

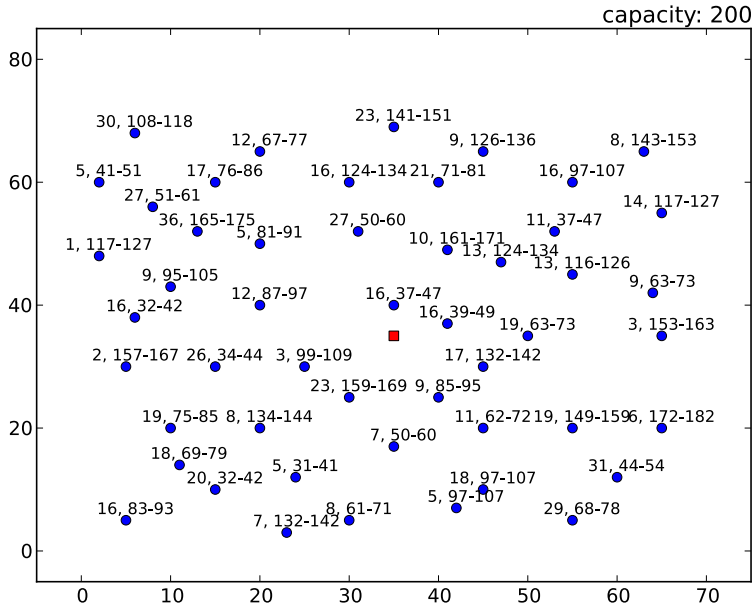
Η συνάρτηση 8 είναι η αντικειμενική συνάρτηση κόστους του προβλήματος και σκοπός είναι η ελαχιστοποίηση της. Η εξίσωση 9 δηλώνει πως η κάθε πιθανή ακμή στο γράφημα θα παίρνει την τιμή 1 αν εισαχθεί στη διαδρομή και 0 αν διαφορετικά. Οι επόμενες εξισώσεις 10,11,12 μας δείχνουν πως κάθε κόμβος πρέπει να επισκέπτεται ακριβώς μια φορά. Το όχημα δηλαδή πρέπει να εισέλθει και να εξέλθει ακριβώς μια φορά. Στην παρακάτω εικόνα **1.4.3** δίνεται μια σχηματική απεικόνιση ενός στιγμιοτύπου TSP με τις πρωτεύουσες όλων των πολιτειών των ΗΠΑ και μια εφικτή λύση του προβλήματος (Πηγή εικόνας: `https://physics.aps.org/synopsis-for/10.1103/PhysRevA.95.032323`).

Σχήμα **1.4.3** Στιγμιότυπο ενός TSP και η αναπαράσταση μια εφικτής λύσης

1.5 Το πρόβλημα της δρομολόγησης οχημάτων με χρονικά παράθυρα

Το πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα αποτελεί ουσιαστικά μια εκδοχή του προβλήματος δρομολόγησης οχημάτων περιορισμένης χωρητικότητας. Οι περιορισμοί χωρητικότητας εξακολουθούν να ισχύουν όμως αυτή την παραλλαγή ισχύει και ο επιπλέον περιορισμός σύμφωνα με τον οποίο κάθε πελάτης πρέπει να εξυπηρετηθεί μέσα σε ένα συγκεκριμένο και δεδομένο από την αρχή χρονικό διάστημα (a_i, b_i). Τα επιπλέον δεδομένα που δίνονται για την επίλυση του προβλήματος είναι η χρονική στιγμή που ξεκινούν τα οχήματα από την κεντρική αποθήκη, ο χρόνος ταξιδιού από τον κόμβο i στον κόμβο j, ο χρόνος εξυπηρέτησης για κάθε πελάτη και φυσικά τα χρονικά περιθώρια που έχουμε στη διάθεσή μας για την εξυπηρέτηση του. Αντικειμενικός στόχος του προβλήματος είναι η ελαχιστοποίηση της διαδρομής των οχημάτων και του συνολικού χρόνου μεταφοράς, καθώς και η εξυπηρέτηση της ζήτησης των πελατών μέσα στον οριοθετημένο χρόνο (Solomon 1987).

Η εφικτότητα μιας λύσης εξαρτάται μόνο από το εάν ο πελάτης εξυπηρετείται τη χρονική στιγμή που έχει ζητηθεί και όχι αργότερα. Δηλαδή εάν ένα όχημα φτάσει πολύ αργά τότε η λύση δεν είναι εφικτή αφού το χρονικό παράθυρο έχει κλείσει. Στην περίπτωση όμως που φτάσει νωρίς τότε μπορεί απλά να περιμένει εκεί μέχρι να ανοίξει το χρονικό παράθυρο και να εξυπηρετήσει τον πελάτη. Στο σχήμα **1.5.4** έχουμε το στιγμιότυπο ενός VRPTW προβλήματος (Πηγή εικόνας: `http://find-santa.eu/presentations/vrptwms-intro/#(6)`).



Σχήμα **1.5.4** Στιγμιότυπο ενός VRPTW

Εκφράζοντας μαθηματικά τις παραπάνω προτάσεις ζητάμε την ελαχιστοποίηση της ποσότητας

$$c = \sum_{ij} c_{ij} \sum_{v} x_{ijv} \tag{14}$$

$$x_{i,j}^{v} = \begin{cases} 1 & \text{αν οι κόμβοι } i, j \text{ είναι συνεχόμενοι στη διαδρομή του οχήματος } v \\ 0 & \text{διαφορετικά} \end{cases} \tag{15}$$

$$\sum_{k \in E} \sum_{j \in V} x_{ijv} = 1 \quad \forall j \in N \tag{16}$$

$$\sum_{j \in V-0} x_{0jv} = 1 \quad \forall j \in N \quad v \in k \tag{17}$$

$$\sum_{j \in V-(j)} x_{ijv} - \sum_{j \in V-(j)} x_{jiv} = 0 \quad \forall j \in N \quad v \in k \tag{18}$$

$$\sum_{j \in V-(n+1)} x_{i(n+1)v} = 1 \quad \forall v \in k \tag{19}$$

$$x_{ijv}(w_{iv} + s_i + t_{ji} + w_{jv}) \leq 0 \quad \forall v \in K \quad \forall (i,j) \in K \tag{20}$$

$$a_i \sum_{j \in V} x_{ijv} \leq w_{iv} \leq b_i \sum_{j \in V} x_{ijv} \quad \forall i \in N \quad \forall v \in k \tag{21}$$

$$E \leq w_{iv} \leq L \quad \forall i \in (0, n+1) \quad \forall v \in K \tag{22}$$

$$\sum_{i \in N} d_i \sum_{j \in V} x_{ijv} \leq C \quad \forall v \in K \tag{23}$$

$$0 \leq x_{ijv} \quad \forall v \in K \quad \forall (i,j) \in A \tag{24}$$

Η αντικειμενική συνάρτηση εκφράζει το συνολικό κόστος. Με τους περιορισμούς 17,18,19 χαρακτηρίζουμε τη ροή της διαδρομής που κάνει το όχημα ν. Οι περιορισμοί 20,22 και 23 ελέγχουν αν είναι εφικτή η τοποθέτηση ενός πελάτη σε ένα κύκλο με βάση τα χρονικά διαστήματα και τη χωρητικότητα του οχήματος. Με τη χρήση του περιορισμού 21 θέτουμε το w_{iv} ίσο με μηδέν αν ένα όχημα δεν επισκέπτεται τους πελάτες i και j στη συγκεκριμένη διαδρομή.

1.6 Το πρόβλημα της δρομολόγησης οχημάτων με πολλαπλές αποθήκες

Στο συγκεκριμένο είδος προβλήματος Multi Depot VRP - MDVRP η εξυπηρέτηση των πελατών γίνεται από περισσότερες από μια αποθήκες. Αν οι πελάτες είναι τοποθετημένοι σε σύνολα κοντά στις αποθήκες, τότε το πρόβλημα διανομής πρέπει να μοντελοποιηθεί ως ενα σύνολο από ξεχωριστά και πλήρως ανεξάρτητα VRP. Σε ένα MDVRP κάθε αποθήκη μπορεί ναν εξυπηρετήσει ενα συγκεκριμένο σύνολο πελατών. Ένας καθορισμένος αριθμός οχημάτων βρίσκεται σε κάθε αποθήκη ξεχωριστά, ενώ κάθε όχημα που προέρχεται από μία αποθήκη, εξυπηρετεί τους πελάτες που έχουν ανατεθεί στην αποθήκη αυτή, και επιστρέφει στην ίδια αποθήκη από την οποία προήλθε (Cordeau, Gendreau, and Laporte 1997).

Αρκετά συχνά παρουσιάζεται και η περίπτωση όπου ένα όχημα ξεκινάει από μια αποθήκη και τερματίζει σε μια άλλη, είτε ενδιάμεσα σταματάει σε κάποια άλλη αποθήκη για να φορτώσει, για παράδειγμα, επιπλέον προιόντα και στη συνέχεια να συνεχίσει την διαδρομή του (Crevier, Cordeau, and Laporte 2007). Ο στόχος του προβλήματος είναι να εξυπηρετηθούν όλοι οι πελάτες, ενώ παράλληλα να ελαχιστοποιείται ο αριθμός χρήσης οχημάτων καθώς και η διανυόμενη απόσταση. Η μαθηματική μοντελοποίηση του προβλήματος ακολουθεί παρακάτω.

$$c = \sum_{i=0}^{d} \sum_{j=0}^{n} c_{ij} \sum_{k=1}^{K} x_{ijk} + \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} \sum_{k=1}^{K} x_{ijk} \tag{25}$$

$$\sum_{k=1}^{K} y_{ik} = 1 \quad \forall i = 1, 2, ..., n \tag{26}$$

$$\sum_{i=1}^{d}\sum_{k=1}^{K} y_{ik} = K \tag{27}$$

$$\sum_{j=1}^{n} x_{ijk} = \sum_{j=1}^{n} x_{jik} = y_{ik} \quad \forall i = 1, 2, ..., n \quad \forall k = 1, 2, ..., K \tag{28}$$

$$\sum_{k=1}^{K} x_{ijk} = \sum_{k=1}^{K} x_{bik} \quad \forall i = 1, 2, ..., d \quad \forall j, b \in V \tag{29}$$

$$\sum_{i \in S} d_i y_{ik} \leq C, \quad S \subseteq V \quad \forall k = 1, 2, ..., K \tag{30}$$

$$\sum_{j \in S}\sum_{j \in S} c_{ij} x_{ijk} = D, \quad S \subseteq V \quad \forall k = 1, 2, ..., K \tag{31}$$

$$\sum_{i \in S_1}\sum_{j \notin S_1} x_{ijk} \geq y_{hk} \quad \forall S_1 \in V, h \in S_1 \quad \forall k = 1, 2, ..., K \tag{32}$$

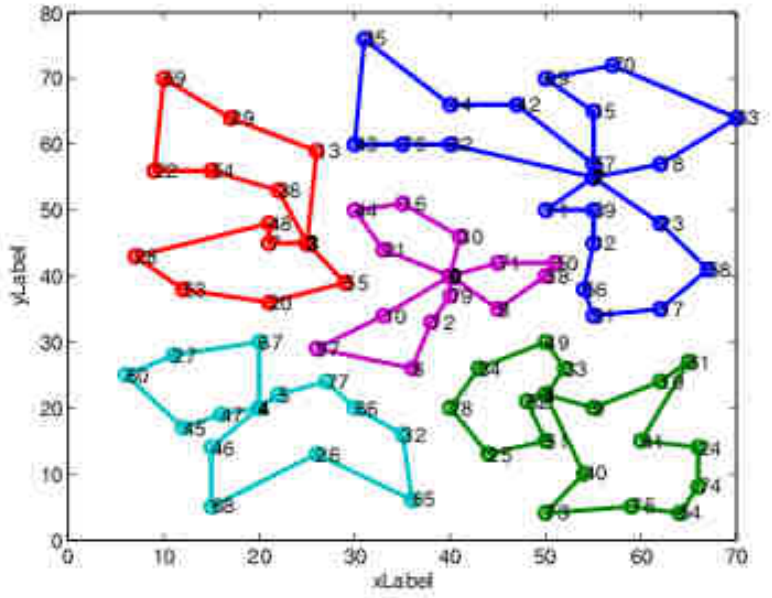
$$y_{ik} \in \{0, 1\}, \quad \forall i = 1, 2, ..., n \quad \forall k = 1, 2, ..., K \tag{33}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i = 1, 2, ..., n \quad \forall j = 1, 2, ..., n \quad \forall k = 1, 2, ..., K \tag{34}$$

Η συνάρτηση 25 είναι η προς ελαχιστοποίηση συνάρτηση η οποία αποτελείται από δύο αθροίσματα. Το πρώτο εκφράζει το κόστος μεταφοράς των προϊόντων όταν το όχημα πραγματοποιεί την πρώτη του διαδρομή από την αποθήκη προς κάποιο πελάτη ενώ το δεύτερο εκφράζει το κόστος για την μεταφορά των προϊόντων μεταξύ πελατών κατά την διάρκεια μια διαδρομής.

Όσον αφορά τους περιορισμούς, η σχέση 26 διασφαλίζει ότι ο πελάτης θα δέχεται εξυπηρέτηση μία φορά. Η σχέση 27 δείχνει ότι όλος ο στόλος οχημάτων πρέπει να χρησιμοποιηθεί για την επίλυση του προβλήματος. Η σχέση 28 δείχνει οτι ένα όχημα πηγαίνει σε έναν πελάτη και στη

συνέχεια φεύγει από αυτόν. Η σχέση 29 εξασφαλίζει ότι κάθε ένα όχημα k που φεύγει από την αποθήκη i προς τον πελάτη στον κόμβο j θα επιστρέψει και πάλι στην αποθήκη i μετά τον τελευταίο πελάτη b που θα εξυπηρετήσει. Ο Περιορισμός 30 είναι περιορισμός που αφορά την χωρητικότητα του οχήματος ενώ ο 31 θέτει όρια χιλιομετρικά ή χρονικά στην απόσταση που μπορεί να διανύσει ένα όχημα κατά την διάρκεια μιας διαδρομής. Τέλος η σχέση 32 εξασφαλίζει την συνέχεια της διαδρομής του οχήματος k. Στο σχήμα **1.6.5** που ακολουθεί έχουμε το στιγμιότυπο ενός MDVRP προβλήματος και η αναπαράσταση της λύσης του (Πηγή εικόνας: `http://immiao.github.io/2014/09/10/mutiple-depot-vehicle-routing-problem/`).



Σχήμα **1.6.5** Στιγμιότυπο ενός MDVRP

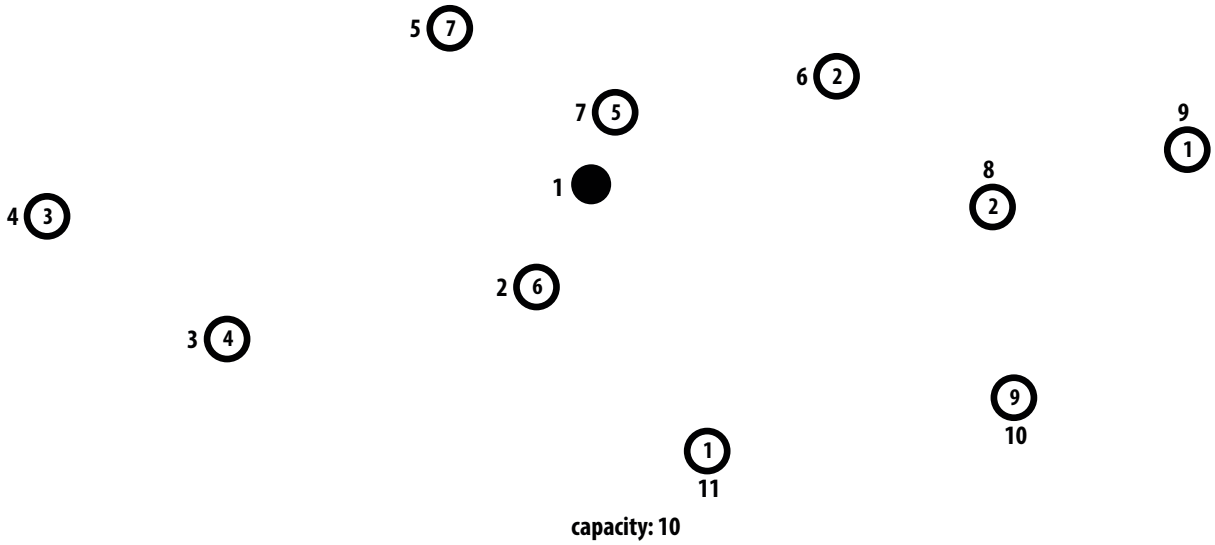
ΚΕΦΑΛΑΙΟ 2

Κατασκευή αρχικής λύσης

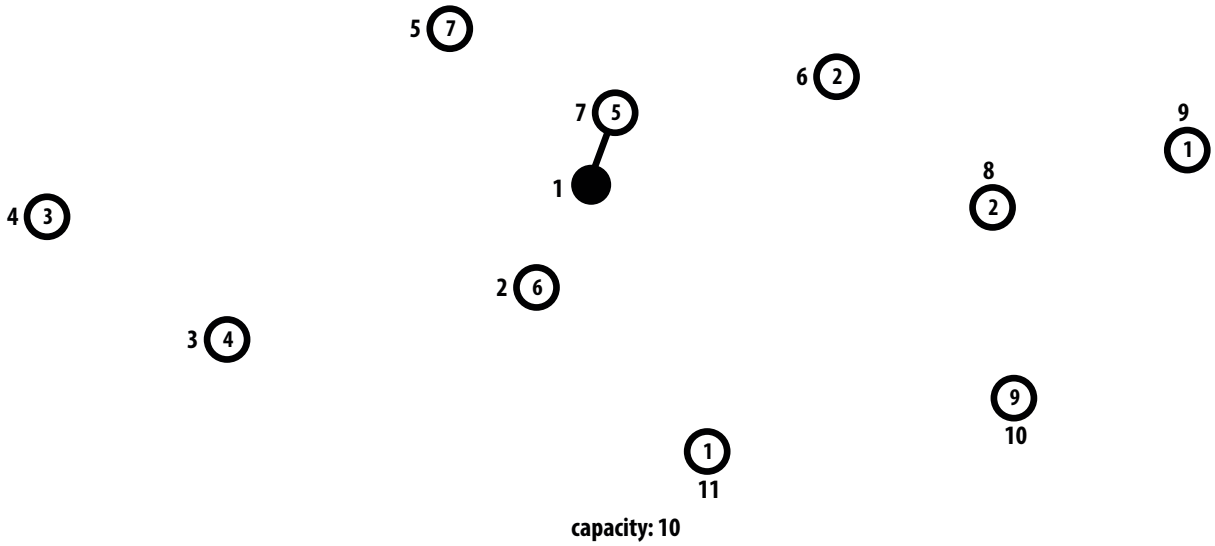
2.1 Η μέθοδος του εγγύτερου γείτονα

Η μέθοδος του εγγύτερου γείτονα είναι η πιο απλή στη σύλληψη μέθοδος με την οποία μπορούμε να κατασκευάσουμε ένα αρχικό σύνολο διαδρομών για το CVRP αλλά και για πολλά άλλα προβλήματα συνδυαστικής βελτιστοποίησης. Πρόκειται για μία άπληστη μέθοδο δηλαδή μια τεχνική η οποία επιλέγει σε κάθε βήμα την φαινομενικά καλύτερη επιλογή και με αυτό τον τρόπο επισκέπτεται τους κόμβους ζήτησης. Οι άπληστες μέθοδοι συνήθως δίνουν ποιοτικά φτωχές λύσεις στα υπολογιστικά δύσκολα προβλήματα όπως το CVRP όμως αποτελούν ένα καλό σημείο εκκίνησης για την βελτιστοποίηση που θα ακολουθήσει έτσι ώστε να φτάσουμε όσο το δυνατον πιο κοντά στην πραγματικά βέλτιστη λύση του προβλήματος (Joshi and Kaur 2015).

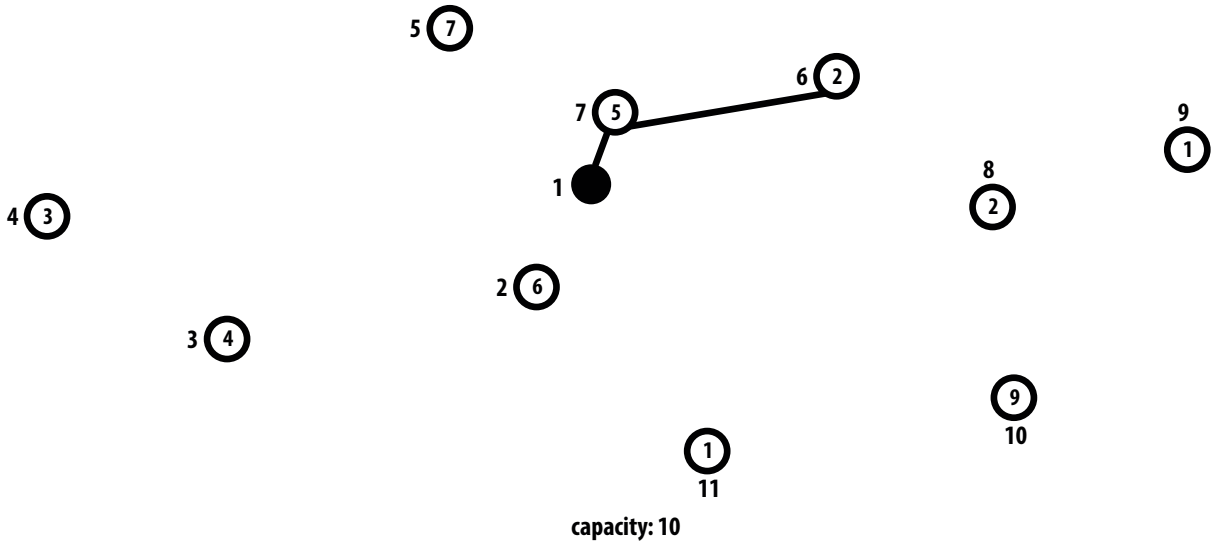
Ο αλγόριθμος ξεκινά με δεδομένα τον πίνακα που δείχνει τις αποστάσεις μεταξύ των κόμβων, τη λίστα που περιέχει την ζήτηση του κάθε κόμβου και φυσικά με την χωρητικότητα των οχημάτων. Ξεκινάμε αρχικοποιώντας μια κενή λίστα στην οποία θα προσθέσουμε κόμβους προς επίσκεψη και μια μεταβλητή που δείχνει τη συνολική ζήτηση της διαδρομής. Στη συνέχεια καλείται η συνάρτηση που βρίσκει τον επόμενο κόμβο προς επίσκεψη. Η συνάρτηση αυτή ψάχνει ποιος είναι ο κοντινότερος κόμβος ζήτησης στην κεντρική αποθήκη (depot) και τον προσθέτει στη λίστα. Η ζήτηση του κόμβου προστίθεται στην μεταβλητή αυτή και η συνάρτηση εύρεσης του επόμενου προς επίσκεψη κόμβου καλέιται ξανά από τον κόμβο στον οποίο βρισκόμαστε. Κάθε φορά που η συνάρτηση κάνει αναζήτηση για τον επόμενο κόμβο ελέγχει αν η υπολειπόμενη χωρητικότητα του οχήματος αρκεί για να καλύψει τη ζήτηση του κόμβου. Έαν δεν επαρκεί η χωρητικότητα ελέγχει τον αμέσως κοντινότερο κόμβο. Η διαδικασία επαναλαμβάνεται μέχρι να μην υπάρχει κόμβος που να μπορεί να εξυπηρετηθεί από το συγκεκριμένο όχημα. Η διαδικασία ξεκινά από την αρχή έχοντας διαγράψει όλους τους κόμβους που εχουν εξυπηρετηθεί από το προηγούμενο όχημα και επαναλαμβάνεται μέχρι να εξυπηρετηθούν όλοι οι κόμβοι ζήτησης. Στις παρακάτω εικόνες **2.1.1** - **2.1.7** φαίνεται βήμα προς βήμα η κατασκευή των διαδρομών από τον αλγόριθμο του εγγύτερου γείτονα.



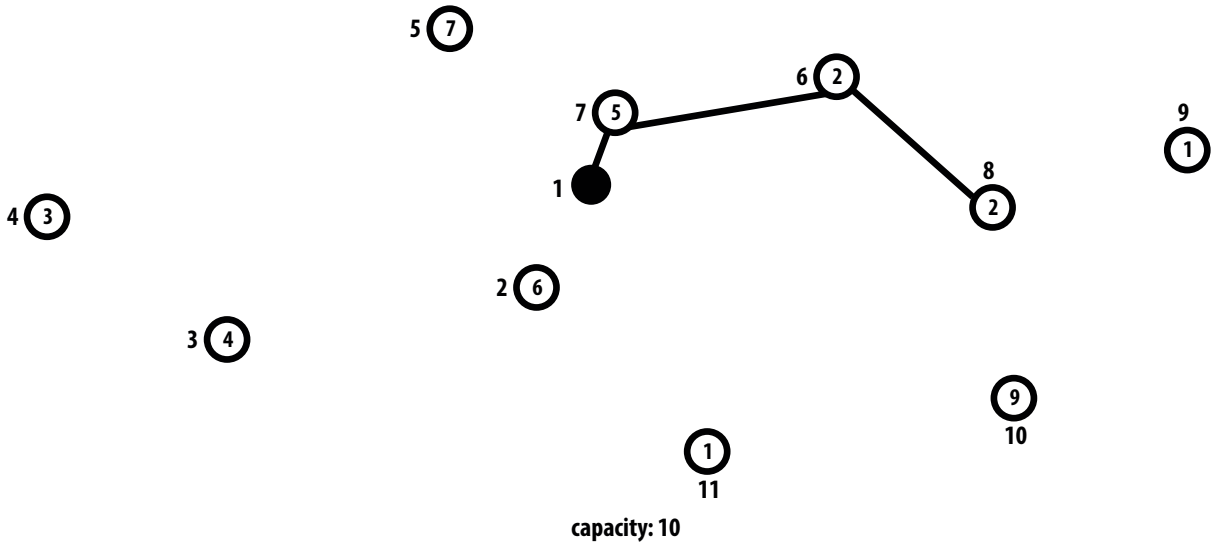
Σχήμα **2.1.1** Βασικό σχήμα ενός στιγμιοτύπου CVRP



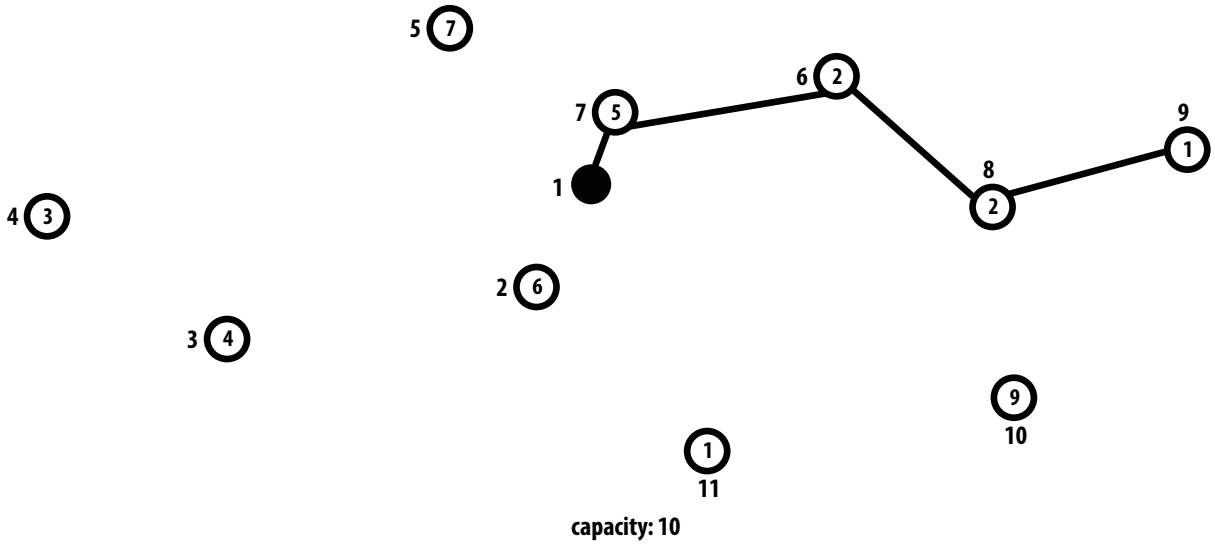
Σχήμα **2.1.2** Πρώτο βήμα του αλγορίθμου nearest neighbor



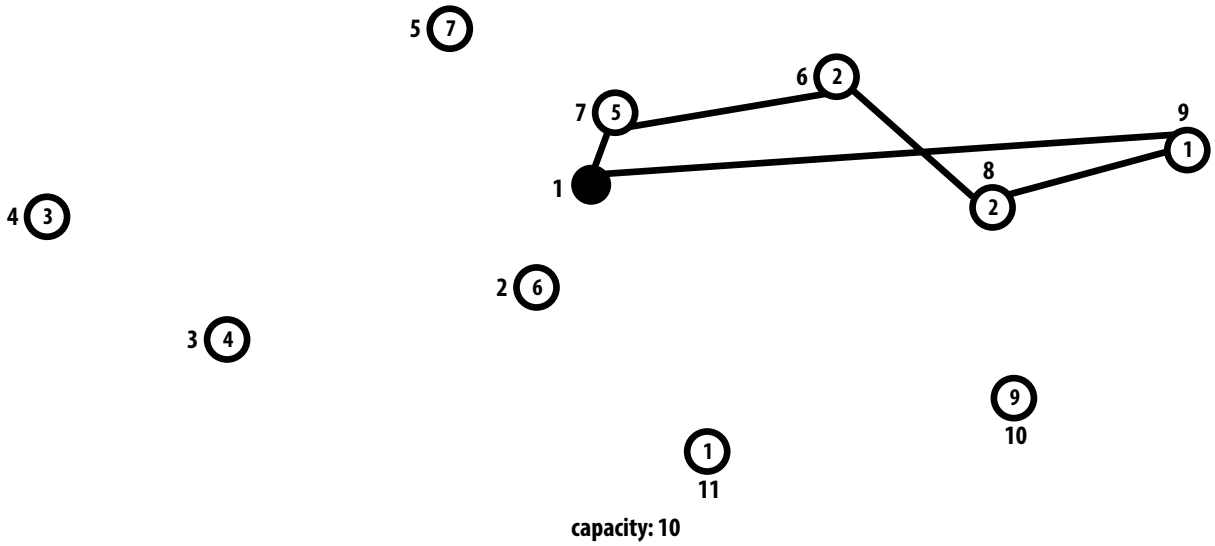
Σχήμα **2.1.3** Δεύτερο βήμα του αλγορίθμου nearest neighbor



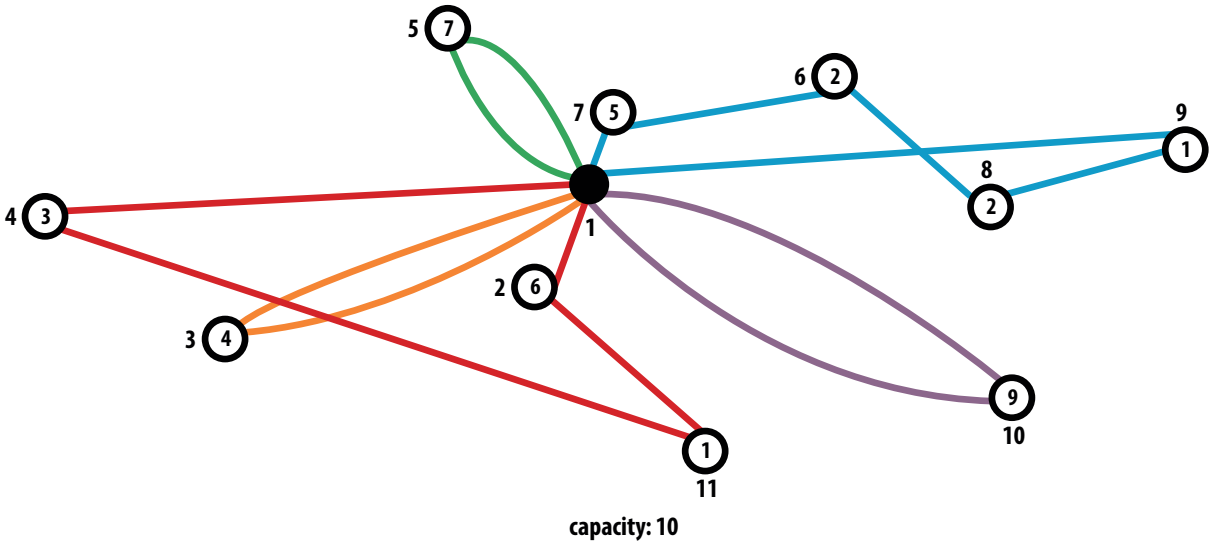
Σχήμα **2.1.4** Τρίτο βήμα του αλγορίθμου nearest neighbor



Σχήμα **2.1.5** Τέταρτο βήμα του αλγορίθμου nearest neighbor



Σχήμα **2.1.6** Πέμπτο βήμα του αλγορίθμου nearest neighbor



Σχήμα **2.1.7** Τελική δρομολόγηση με nearest neighbor

Παρακάτω παρατίθεται ο ψευδοκώδικας 1 πάνω στον οποίο βασιστηκέ η υλοποίηση του αλγορίθμου στη γλώσσα προγραμματισμού Python.

Algorithm 1 Αλγόριθμος Nearest Neighbor

Require: $DISTANCEMATRIX, DEMAND, CAPACITY$

procedure NEAREST NEIGHBOR($DISTANCEMATRIX, DEMAND, CAPACITY$)
    routing = [ ]
    available = [2,3,...,N]
    while *available* $\neq \emptyset$ **do**
        current = 1
        route = [1]
        find nextstep(current)
        while *nextstep* $\neq -1$ **do**
            append nextstep to route
            current = nextstep
            remove current from available
            find nextstep(current)
        end while
        append route to routing
    end while
end procedure

2.2 Η μέθοδος εξοικονόμησης των Clarke και Wright

Ο αλγόριθμος που προτάθηκε το 1964 από τους Clarke και Wright (Clarke and Wright 1964) είναι ο πιό διαδεδομένος αλγόριθμος για την κατασκευής μιας αρχικής λύσης για το cvrp πρόβλημα. Η δημοφιλία του αλγορίθμου αυτού οφείλεται στην απλή δομή του καθώς επίσης και στην καλή ποιότητα των αποτελεσμάτων του. Πρόκειται για έναν άπληστο αλγόριθμο, δηλαδή σε κάθε βήμα

κάνει την φαινομενικά καλύτερη επιλογή με βάση κάποια κριτήρια που έχουμε ορίσει. Αν και άπληστος τα αποτελέσματα που μας δίνει ο αλγόριθμος των Clarke και Wright έχουν στα πιο πολλά στιγμιότυπα, σχετικά μικρή απόκλιση από τις βέλτιστες λύσεις των προβλημάτων. Ο συγκεκριμένος αλγόριθμος αποτελεί τη βάση πολλών σύγχρονων και πιο περίπλοκων αλγορίθμων για την επίλυση πληθώρας προβλημάτων που σχετίζονται με την δρομολόγηση οχημάτων.

Η κεντρική ιδέα του αλγορίθμου είναι ο συντελεστής αποταμίευσης σύμφωνα με τον οποίο ταξινομούνται οι πιθανές ακμές και κατασκευάζονται βήμα βήμα οι διαδρομές της λύσης. Ο συντελεστής αποταμίευσης στην ουσία μας δείχνει κατά πόσο μας συμφέρει η τοποθέτηση μιας συγκεκριμένης ακμής στην υπό κατασκευή διαδρομή. Για την εξυπηρέτηση των κόμβων ζήτησης έχουμε διαθέσιμα οχήματα τα οποία έχουν μια δεδομένη χωρητικότητα. Ο στόχος είναι φυσικά να ανατεθεί ο κάθε κόμβος ζήτησης (πελάτης) σε ένα όχημα και η αποτελεσματική δρομολόγηση των οχημάτων, έτσι ώστε να είναι όσο το δυνατόν χαμηλότερο το συνολικό κόστος. Ο συντελεστής αποταμίευσης της ακμής που συνδέει δύο κόμβους i και j υπολογίζεται από τον τύπο $S_{ij} = D_{i0} + D_{j0} - D_{ij}$ όπου D_{i0} και D_{j0} είναι η αποστάσεις από την απόθηκη των κόμβων i και j αντίστοιχα ενώ , D_{ij} είναι η απόσταση του κόμβου i από τον κόμβο j.

Ο τρόπος με τον οποίο ορίζεται ο συντελεστής αποταμίευσης μας οδηγεί στο συμπέρασμα πως όσο πιο κοντά βρίσκονται δύο κόμβοι και όσο πιο μακριά από την αποθήκη τόσο υψηλότερος θα είναι ο συντελεστής αποταμίευσης της συγκεκριμένης ακμής και συνεπώς τόσο πιθανότερη η τοποθέτηση της στη διαδρομή ενός οχήματος. Αν δηλαδή δύο κόμβοι είναι πολύ κοντά μεταξύ τους και πολύ μακρυά από την αποθήκη τότε το πιθανότερο είναι να τοποθετηθούν σε μία διαδρομή ο ένας μετά τον άλλο (Lysgaard 1997).

Παρακάτω παρατίθεται ο ψευδοκώδικας 2 πάνω στον οποίο βασιστηκέ η υλοποίηση του αλγορίθμου στη γλώσσα προγραμματισμού Python.

Algorithm 2 Αλγόριθμος Clarke Wright

Require: $DISTANCE\ MATRIX, DEMAND, CAPACITY$

procedure CLARKE WRIGHT($DISTANCE\ MATRIX, DEMAND, CAPACITY$)
    $routing \leftarrow [\,]$
    Create a list with all the feasible edges
    Calculate savings for all feasible edges
    $edges \leftarrow (i, j, saving)$
    Sort the edges list by savings in descenting order
    while $edges \neq \emptyset$ **do**
        $route \leftarrow [\,]$
        $Not\ complete \leftarrow True$
        while $Not\ complete$ **do**
            Append the edge that is in the top of the edges list to route and delete it from the edges list
            Find the next edge that can be placed in the route starting from the ton of the list
            if *There is an edge that can be placed* **then**
                Append the edge that is in the top of the edges list to route and delete it from the edges list
                Delete all edges that have at least one vertice that belong in the middle of the route
            else
                Append route in to routes
                $not\ complete \leftarrow False$
            end if
        end while
    end while
end procedure

Θα ακολουθήσει ένα μικρό παράδειγμα για την καλύτερη κατανοήση της λειτουργίας του αλγορίθμου. Έστω ένα συμμετρικό πρόβλημα με 7 πελάτες. Η αρίθμηση ξεκινά από το 1 το οποίο το αναθέτουμε στην κεντρική αποθήκη και φτάνει μέχρι το 8. Παρακάτω δίνονται ο πίνακας αποστάσεων καθώς και ο πίνακας που μας δείχνει τις απαιτήσεις κάθε κόμβου ζήτησης. Η χωρητικότητα των οχημάτων θεωρούμε ότι είναι 20.

Πίνακας **2.1** Πίνακας αποστάσεων παραδείγματος

-	1	2	3	4	5	6	7	8
1	0	13	23	21	17	20	18	25
2	0	0	25	27	23	26	21	15
3	0	0	0	14	22	18	30	16
4	0	0	0	0	30	20	19	28
5	0	0	0	0	0	14	15	9
6	0	0	0	0	0	0	12	14
7	0	0	0	0	0	0	0	16
8	0	0	0	0	0	0	0	0

Τώρα υπολογίζεται ο συντελεστής εξοικονόμησης για κάθε εφικτή ακμή του προβλήματος και

Πίνακας **2.2** Πίνακας ζήτησης παραδείγματος

Κόμβος	Ζήτηση
1	0
2	10
3	7
4	2
5	9
6	8
7	5
8	6

δημιουργείται η ταξινομημένη λιστα με τις ακμές.

Πίνακας **2.3** Πίνακας συντελεστών εξοικονόμησης

-	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	11	7	7	7	10	23
3	0	0	0	30	18	25	11	32
4	0	0	0	0	8	21	20	18
5	0	0	0	0	0	23	20	33
6	0	0	0	0	0	0	26	31
7	0	0	0	0	0	0	0	27
8	0	0	0	0	0	0	0	0

Ξεκινάμε βάζοντας την ακμή με τον μεγαλύτερο συντελεστή εξοικονόμησης. Αύτη είναι η (5-8) με συντελεστή 33. Οι κόμβοι 5 και 8 έχουν συνολική ζήτηση $9 + 6 = 15$. Στη συνέχεια ψάχνουμε την επόμενη ακμή της οποίας ένα άκρο να είναι το 5 ή το 8. Βρίσκουμε την ακμή (3-8) η οποία έχει συντελεστή 32 όμως δεν μπορεί να τοποθετηθεί στην υπάρχουσα διαδρομή αφού η ζήτηση του κόμβου 3 είναι 7 και έτσι θα έχουμε συνολική ζήτηση $9+6+7 = 22$ που υπεβαίνει την χωρητικότητα του οχήματος. Για τον ίδιο λόγο απορρίπτουμε και την αμέσως οικονομικότερη ακμή που είναι η (6-8). Συνεχίζοντας την αναζήτηση βρίσκουμε την ακμή (8-7) η οποία μπορεί να τοποθετηθεί στη διαδρομή. Η συνολική χωρητικότητα είναι $9+6+5 = 20$ άρα σταματάμε την κατασκευή της παρούσας

διαδρομής αφού φτάσαμε στο μέγιστο της χωρητικότητας του οχήματος. Έτσι κατασκευάστηκε η διαδρομή (5-8-7) με συνολική ζήτηση 20 και συνολικό κόστος $d(1,5)+d(5,8)+d(8,7)+d(7,1) = 17+9+16+18 = 60$. Τώρα πρέπει να διαγραφούν όλες οι ακμές που έχουν ένα τουλάχιστον άκρο στη διαδρομή που ήδη κατασκευάστηκε.

Στη συνέχεια ξεκινάμε τη διαδικασία από την αρχή επιλέγοντας πάλι την οικονομικότερη ακμή που είναι η (3-4) με συντελεστή 30 και συνολική ζήτηση 7+2 = 9. Η επόμενη ακμή που τοποθετείται είναι η (6-3) με συντελεστή 25 και τώρα η συνολική ζήτηση είναι $7 + 2 + 8 = 17$. Η κατασκευή της διαδρομής τελειώνει αφού ο κόμβος 2 που απομένει έχει χωρητικότητα 10 και δεν μπορεί να τοποθετηθεί στην παρούσα διαδρομή. Το κόστος της ανέρχεται σε $d(1,6) + d(6,3) + d(3,4) + d(4,1) = 20 + 18 + 14 + 21 = 73$ Τώρα πρέπει να διαγραφούν όλες οι ακμές που έχουν ένα τουλάχιστον άκρο στη διαδρομή (6-3-4) που μόλις κατασκευάστηκε. Παρατηρούμε ότι δεν έχει μείνει καμία εφικτή ακμή, όμως ο κόμβος 2 δεν έχει εξυπηρετηθεί ακόμα. Σε αυτή την περίπτωση κατασκευάζεται μια διαδρομή που θα εξυπηρετεί μόνο τον κόμβο 2 με κόστος $d(1,2) + d(2,1) = 13 + 13 = 26$

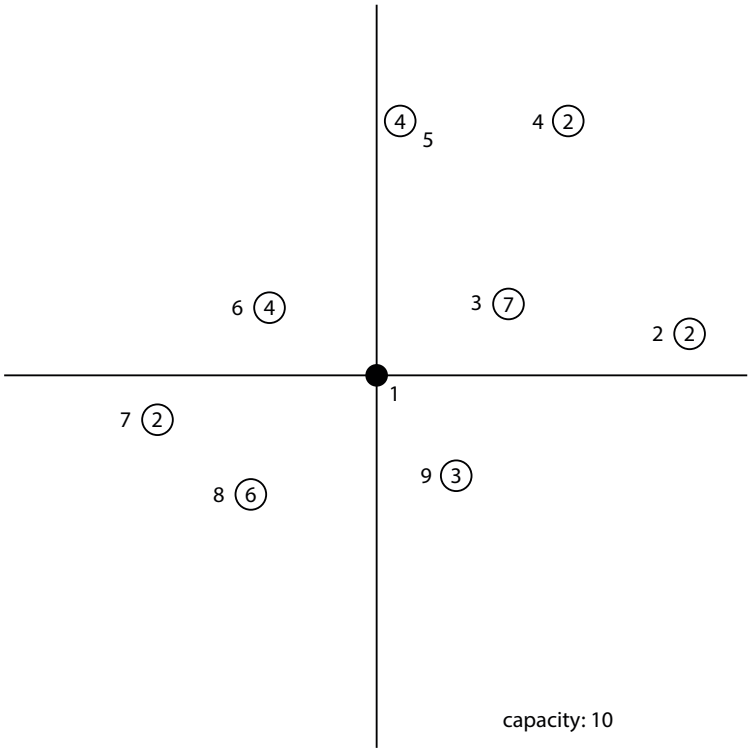
Έτσι για αυτό το στιγμιότυπο ακολουθώντας τα βήματα του αλγορίθμου Clarke Wright κατασκευάστηκαν οι διαδρομές (5-8-7),(6-3-4),(2) με συνολικό κόστος $60 + 73 + 26 = 159$.

2.3 Η μέθοδος σάρωσης (Sweep)

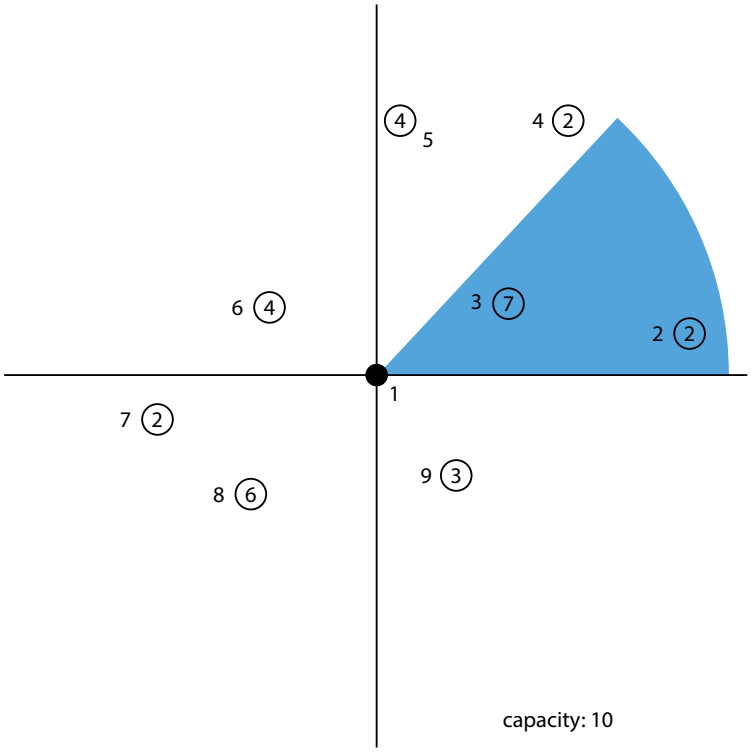
Μια από τις πιο δημοφιλείς και αποτελεσματικές μεθόδους κατασκευής για το CVRP είναι ο αλγόριθμος sweep ή αλλιώς αλγόριθμος σάρωσης που προτάθηκε από τους Gillette & Miller το 1974 (Gillett and Miller 1974). Ο αλγόριθμος αυτός εστιάζει αρχικά στην συσταδοποίηση των κόμβων δηλαδή στο να χωριστούν οι κόμβοι ζήτησης σε ομάδες που μπορούν να αποτελέσουν ξεχωριστές διαδρομές. Αυτό επιτυγχάνεται χρησιμοποιόντας έναν περιστρεφόμενο ημιάξονα ο οποίος σαρώνει το επίπεδο και αναθέτει τους κόμβους ζήτησης σε διαδρομές με τη σειρά που σαρώθηκαν και φυσικά ικανοποιόντας τους περιορισμούς χωρητικότητας που επιβάλλει το πρόβλημα. Για να πετύχουμε το επιθυμητό αποτέλεσμα προγραμματιστικά κάνουμε μια παράλληλη μεταφορά του στιγμιοτύπου που δουλεύουμε στο επίπεδο έτσι ώστε η αποθήκη να ταυτίζεται με την αρχή των αξόνων. Στη συνέχεια υπολογίζεται η γωνία ϑ που ορίζει τις πολικές συντεταγμένες κάθε κόμβου (Laporte and Semet 2002). Πρακτικά η ζητούμενη γωνία ϑ ενός κόμβου είναι η γωνία που πρέπει να περιστραφεί κατά τη θετική φορά ο θετικός ημιάξονας x έτσι ώστε να συναντήσει τον κόμβο αυτόν. Προφανώς η γωνία ϑ κινείται στο διάστημα [0,2π) και υπολογίζεται άμεσα από τον τύπο ϑ= atan2(y,x) όπου (x,y) είναι οι καρτεσιανές συντεταγμένες του κόμβου.

Στη συνέχεια η λίστα των κόμβων ζήτησης ταξινομούνται με βάση τη σειρά με την οποία σαρώθηκαν δηλαδή με βάση την γωνία ϑ που υπολογίστηκε προηγουμένως. Έτσι ξεκινάμε απο την κορυφή της λίστας και εντάσουμε τους κόμβους με τη σειρά σε μία διαδρομή μέχρις ότου με τον

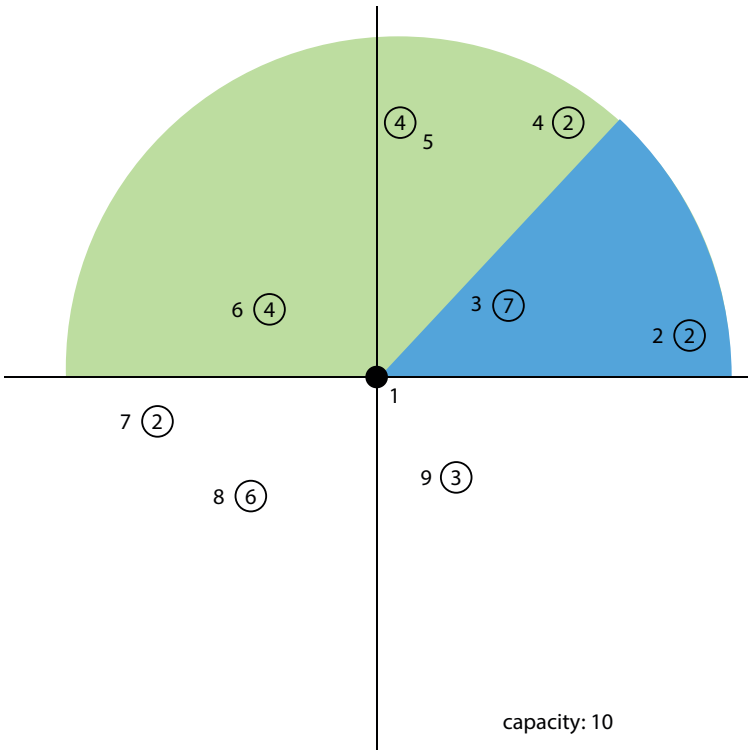
περιορισμό να μην ξεπερνουμε την χωρητικότητα του οχήματος. Όταν η δεδομένη χωρητικότητα ξεπεραστεί τότε αρχικοποιούμε μια νέα διαδρομή. Επαναλαμβάνουμε μέχρι όλοι οι κόμβοι να έχουν ενταχθεί σε μια και μόνο διαδρομή. Με αυτό τον τρόπο έχουμε πετύχει την συσταδοποίηση όπου πλεόν γνωρίζουμε πόσα οχήματα θα μας χρειαστούν και ποιούς κόμβους θα εξυπηρετήσει το καθένα. Έπειτα εφαρμόζεται μία μέθοδος βελτιστοποίησης για να να γίνει αποτελεσματικά η δρομολόγηση στην κάθε συστάδα η οποία πλεόν αντιμετωπίζεται ως ένα απλό TSP πρόβλημα αφού έχει δημιουργηθεί έτσι ώστε να μην ξεπερνά τη χωρητικότητα του οχήματος. Μπορούν να χρησιμοποιηθούν πολλές διαφορετικές μέθοδοι βελτιστοποίησης της διαδρομής όπως swap ,relocate, 2-opt, 3-opt, or-opt οι οποίες θα αναλυθούν στο επόμενο κεφάλαιο. Ενδεικτικά η μέθοδος που χρησιμοποιήθηκε για την υλοποίηση του αλγορίθμου στην παρούσα εργασία ήταν η 2-opt. Στις εικόνες **2.3.8-2.3.12** παρουσιάζεται η συσταδοποίηση βήμα προς βήμα.



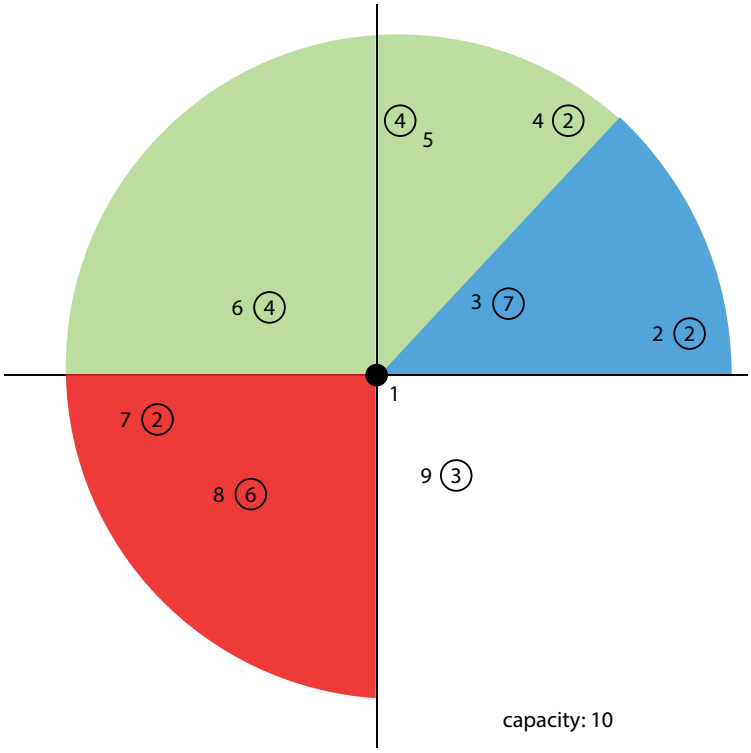
Σχήμα **2.3.8** Αρχικό στιγμιότυπο προβλήματος CVRP



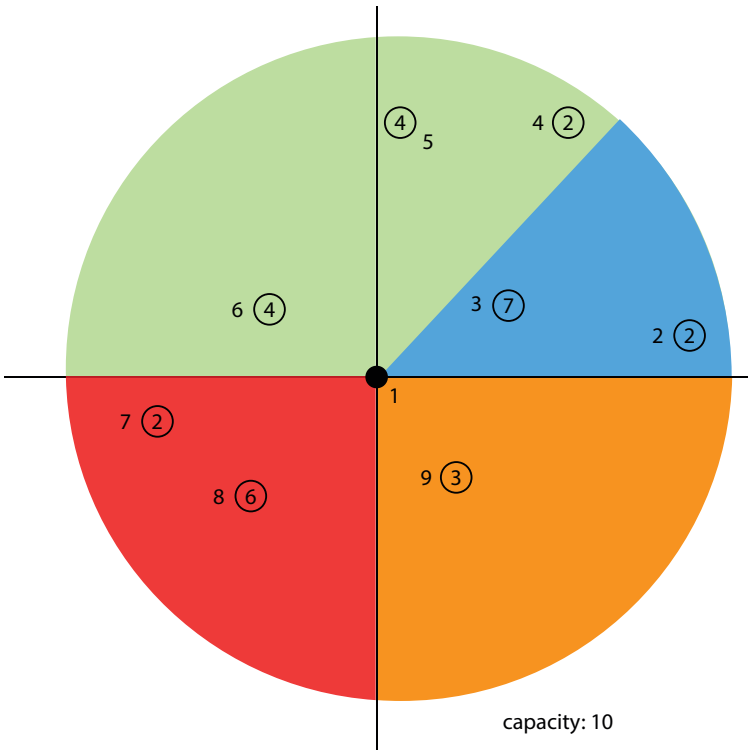
Σχήμα **2.3.9** Πρώτο βήμα του αλγορίθμου sweep



Σχήμα **2.3.10** Δεύτερο βήμα του αλγορίθμου sweep



Σχήμα **2.3.11** Τρίτο βήμα του αλγορίθμου sweep



Σχήμα **2.3.12** Τελική συσταδοποίηση με τον αλγόριθμο sweep

Παρακάτω παρατίθεται ο ψευδοκώδικας πάνω στον οποίο βασιστηκέ η υλοποίηση του αλγορίθμου στη γλώσσα προγραμματισμού Python.

Algorithm 3 Αλγόριθμος Sweep

Require: $COORDINATES, DEMAND, CAPACITY$

procedure SWEEP ALGORITHM($COORDINATES, DEMAND, CAPACITY$)← routing ← [ ]
    compute angle for every demand point
    order the demand point list by angle in ascending order
    $route \leftarrow [\,]$
    $i \leftarrow 1$
    while $i \leqslant N$ **do**
        if $Total\ demand(route) + Demand(sorted\ by\ angle[i]) \leqslant capacity$ **then**
            append sortedbyangle[i] to route
            $i \leftarrow i + 1$
        else
            append route to routes
            $route \leftarrow [\,]$
        end if
    end while
    for route in routing **do**
        $improvement \leftarrow true$
        $cost \leftarrow COST(route)$
        while $improvement \leftarrow true$ **do**
            $route1 \leftarrow 2opt(route)$
            if $COST(route1) \geqslant cost$ **then**
                $improvement \leftarrow false$
            else
                $route \leftarrow route1$
            end if
        end while
    end for
    =0

ΚΕΦΑΛΑΙΟ 3

Τοπική αναζήτηση

3.1 Εισαγωγή

Στην επιστήμη της πληροφορικής η τοπική αναζήτηση είναι μια ευρετική μέθοδος που χρησιμοποιείται για την επίλυση υπολογιστικά δυσκολών προβλημάτων βελτιστοποίησης. Συνήθως εφαρμόζεται σε προβλήματα στα οποία αναζητείται μια λύση που να μεγιστοποιεί ή να ελαχιστοποιεί την αντικειμενική συνάρτηση του προβλήματος, ανάμεσα σε ένα αρκετά μεγάλο πλήθος πιθανών λύσεων. Οι αλγόριθμοι τοπικής αναζήτησης ψάχνουν τη βέλτιστη λύση εφαρμόζοντας έναν τελεστή αναζήτησης μέχρι να βρεθεί η καλύτερη δυνατή λύση ή να τελειώσει το χρονικό περιθώριο αναζήτησης που είχε οριστεί απο την αρχή του αλγορίθμου. Οι αλγόριθμοι τοπικής αναζήτησης χρησιμοποιούνται ευρέως σε πληθώρα δυσκολών υπολογιστικών προβημάτων της πληροφορικής, των μαθηματικών, της επιχειρησιακής έρευνας και της βιοπληροφορικής.

Η ιδέα της τοπικής αναζήτησης προέρχεται από την βασική ανθρώπινη λογική της αναζήτησης μιας λύσης σε ένα πρόβλημα, προσπαθώντας να κάνει επιτρεπτές αλλαγές σε μια υπάρχουσα εφικτή λύση με σκοπό την περαιτέρω βελτιστοποίηση. Αυτό επιτυχγάνεται με τρόπο επαναληπτικό ψάχνοντας σε κάθε επανάληψη κάποιο εφικτό βήμα που θα βελτιώσει την υπάρχουσα λύση. Οι μέθοδοι τοπικής αναζήτησης είναι ευρετικές μέθοδοι συνεπώς βασει ορισμού δεν εγγυώνται ότι θα βρουν το ολικό βέλτιστο και ούτε ότι θα καταφέρουν να βελτιώσουν έστω και ελάχιστα την τρέχουσα λύση, επιστρέφοντας ένα αποτέλεσμα με καλύτερη τιμή αντικειμενικής συνάρτησης. Για αυτό το λόγο είναι απαραίτητο να χρησιμοποιηθούν πολλές διαφορετικές μέθοδοι τοπικής αναζήτησης προκειμένου να μεγιστοποιηθεί η πιθανότητα εύρεσης μιας καλύτερης λύσης (Lourenço, Martin, and Stützle 2003).

Οι μέθοδοι τοπικής αναζήτησης παίρνουν σαν όρισμα μια εφικτή λύση του προβλήματος και ένα κριτήριο τερματισμού. Ορίζεται μια συγκεκριμένη διαδικασία αναζήτησης και εκτελείται μέχρι να ικανοποιηθεί το κριτήριο τερματισμού. Το κριτήριο τερματισμού μπορεί να είναι χρονικό, δηλαδή η αναζήτηση να εκτελείται μέχρι να φτάσει ένα δεδομένο χρονικό όριο, ή να εκτελείται μέχρι να βρει μια οποιαδήποτε βελτίωση, ή να εκτελείται εξαντλητικά μέχρι να εξετάσει κάθε πιθανό βήμα βάσει

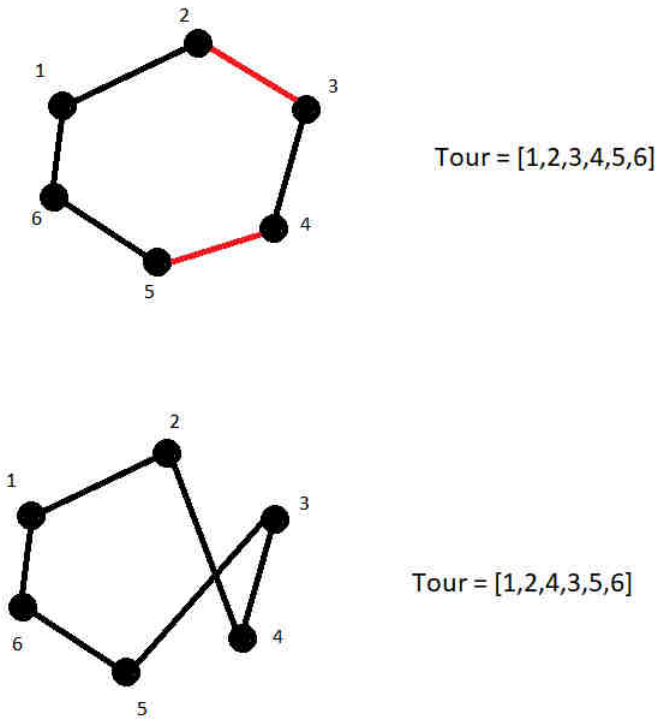
της δεδομένης διαδικασίας και τελικά να επιλέξει το βέλτιστο με κριτήριο την ελαχιστοποίηση (ή την μεγιστοποίηση) της τιμής της αντικειμενικής συνάρτησης.

Θα εξεταστούν διάφορες δομές γειτονιάς για το πρόβλημα CVRP με σκοπό την αναζήτηση σε βάθος για ακόμα πιο ποιοτικές λύσεις. Οι τελεστές αναζήτησης μπορούν να μας προσφέρουν μια πληθώρα βελτιώσεων η οποίες προκύπτουν αιτιοκρατικά και εξαρτώνται από τις παραμέτρους τις οποίες έχουμε θέσει εμείς στον αλγόριθμο υλοποίησης. Οι επιλογές αυτές έχουν να κάνουν με τον χρόνο εκτέλεσης ή με το αν αναζητούμε την καλύτερη δυνατή βελτίωση ή απλά μια οποιαδήποτε βελτίωση.

3.2 2-opt

Μια από τις κλασικότερες δομές γειτονιάς είναι η 2-opt η οποία προτάθηκε αρχικά από τον Croes το 1958 (Croes 1958). Στην εργασία του ορίζει την τεχνική ως μια βελτιστοποιημένη λύση για το πρόβλημα του ταξιδιώτη πωλήσεων τόσο για συμμετρικές όσο και ασύμμετρες περιπτώσεις (οι οποίες απαιτούν περισσότερους υπολογισμούς). Δεν υπάρχει καμία εγγύηση ότι αυτή η τεχνική θα βρει μια ολικά βέλτιστη απάντηση και έτσι χακτηρίζεται ως ευρετική. Αυτό που πραγματικά κάνει είναι να βελτιώσει σταδιακά μια αρχικά δοθείσα και εφικτή λύση (τοπική αναζήτηση) έως ότου φθάσει σε μια τοπικά βέλτιστη λύση και δεν μπορούν να γίνουν άλλες βελτιώσεις. Οι βελτιώσεις γίνονται χρησιμοποιώντας αυτό που ονομάζει «αναστροφές».(Englert, Röglin, and Vöcking 2007)

Η μέθοδος 2-Opt είναι ίσως η πιο βασική και ευρέως χρησιμοποιούμενη τοπική ευρετική αναζήτηση για το ΤΣΠ. Αυτός ο ευρετικός αλγόριθμος επιτυγχάνει πολύ καλά αποτελέσματα σε ευκλείδειες περιπτώσεις ¨πραγματικού κόσμου¨, τόσο σε σχέση με το χρόνο εκτέλεσης όσο και τον λόγο προσέγγισης. Υπάρχουν πολυάριθμες πειραματικές μελέτες σχετικά με την απόδοση της 2-Opt. Παρόλο που η σύγκλιση προς τη βέλτιστη λύση δεν είναι σε καμία περίπτωση βέβαιη, αποδίδει πολύ ποιοτικά αποτελέσματα σε προβλήματα VRP,CVRP καιTSP. Αποτελεί μια από τις βασικότερες δομές γειτονιάς που χρησιμοποιείται στις περισσότερες μεθευρετικές μεθόδους όπως η προσομοιωμένη ανόπτηση και η αναζήτηση μεταβλητής γειτονιάς. Όσον αφορά την πολυπλοκότητα της μεθόδου έχουν γίνει έρευνες που καταλήγουν στο συμπέρασμα πως η 2-Opt έχει πολυωνυμική πολυπλοκότητα χρόνου κατα μέσο όρο $O(n^2)$. Στο σχήμα **3.2.1** φαίνεται η εφαρμογή της μεθόδου σε μια διαδρομή 6 κόμβων.



Σχήμα **3.2.1** Οπτική αναπαράσταση της βελτιστοποίησης με 2-opt

Ο αλγόριθμος ξεκινά με μία δεδομένη εφικτή λύση του προβλήματος και εφαρμόζει τον τελεστή αλλαγής που φαίνεται στο σχήμα για όλα τα πιθανά ζευγάρια ακμών. Τα ζευγάρια ακμών που επιλέγονται για αφαίρεση δεν πρέπει να είναι συνεχόμενα στην δεδομένη διαδρομή διότι έτσι απομονώνεται ένας κόμβος και δεν υπάρχει δυνατότητα διαφορετικής επανασύνδεσης του με την υπόλοιπη διαδρομή. Αφού επιλεχθεί ένα ζευγάρι ακμών αφαιρείται από τη διαδρομή. Έτσι απομονώνεται ένα κομμάτι της διαδρομής το οποίο επανασυνδέεται με φορά αντίθετη από αυτή που είχε πρίν την αφαίρεση.

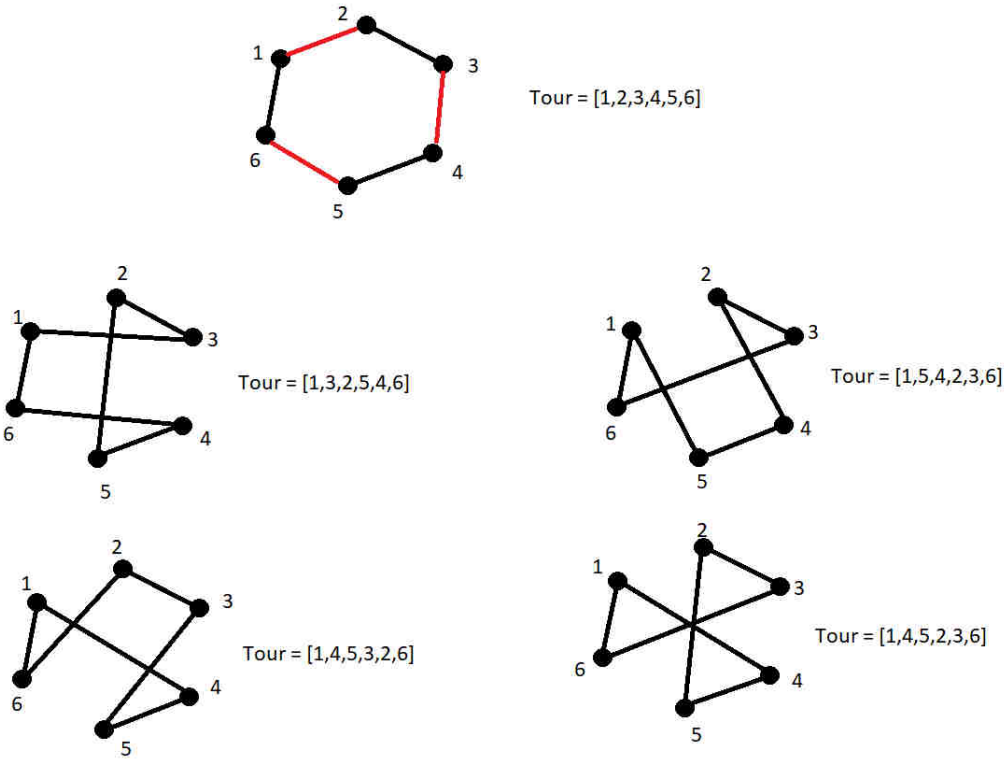
Algorithm 4 2-Opt Algorithm

Require: $Tour, DistanceMatrix$
procedure 2-Opt Algorithm($Tour$)
  for $i = 1$ to $Tour(length)$ **do**
    for $j = i + 2$ to $Tour(length)$ **do**
      $NewTour \leftarrow Tour[1 : i] + Tour[j : i] + Tour[j : n]$
      if $Cost(NewTour) < Cost(Tour)$ **then**
        $Tour \leftarrow NewTour$
      end if
    end for
  end for
  Return Tour
end procedure=0

3.3 3-opt

Η βελτιστοποίηση, 3-opt είναι ένας απλός αλγόριθμος τοπικής αναζήτησης για την επίλυση του προβλήματος του περιοδεύοντος πωλητή αλλά μπορεί να έχει χρήση και σε προβλήματα της κατηγορίας CVRP (Golden, Wasil, Kelly, and Chao 1998). Σε αντιστοιχία με την προηγούμενη μέθοδο 2-opt μπορούμε να εφαρμόσουμε και την 3-opt. Αντί να αφαιρέσουμε 2 ακμές από την δεδομένη διαδρομή τώρα θα αφαιρέσουμε 3 ακμές και θα συνδέσουμε τις υποδιαδρομές με όλους τους πιθανούς τρόπους χωρίς να συμπεριλάβουμε τις συγκεκριμένες ακμές που αφαιρέθηκαν. Στη συνέχεια θα γίνει αξιολόγηση των λύσεων που προκύπτουν από την εφαρμογή του τελεστή και θα επιλεγεί η καλύτερη βάσει της αντικειμενικής συνάρτησης. Η διαδικασία φυσικά επαναλαμβάνεται μέχρι να μην υπάρχει άλλη βελτίωση (Psaraftis 1983).

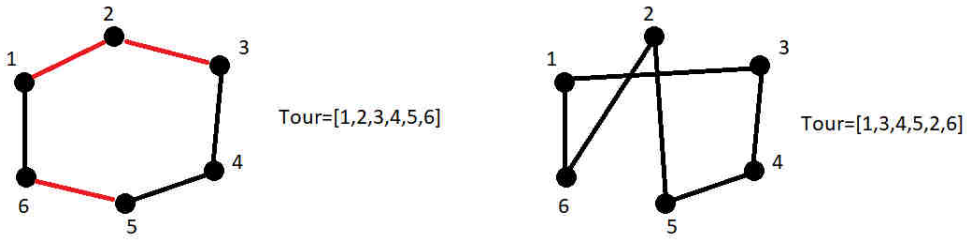
Στην εικόνα **3.3.2** επιλέχθηκαν για διαγραφή οι ακμές (1,2),(3,4),(5,6). Τα κομμάτια της διαδρομής που απομένουν μπορούν να επανασυνδεθούν με 4 διαφορετικούς τρόπους σε αυτή την περίπτωση. Συγκεκριμένα οι 4 πιθανές τριάδες ακμών που μπορούν να κάνουν την επανασύνδεση είναι οι (1,3),(2,5),(6,4) , (1,5),(2,4),(6,3) , (5,3),(2,6),(1,4) , (6,3),(2,5),(1,4). Από αυτές εξετάζουμε ποια είναι η πιο συμφέρουσα και την επιλέγουμε.



Σχήμα **3.3.2**: Οπτική αναπαράσταση του 3-opt όταν οι ακμές που επιλέγονται δεν είναι συνεχόμενες

Όταν οι ακμές που επιλέγονται είναι και οι 3 συνεχόμενες τότε δεν είναι εφικτή η επανασύνδεση

των τμημάτων της διαδρομής με διαφορετικό τρόπο, ενώ όταν είναι συνεχόμενες μόνο οι 2 από τις 3 τότε η επανασύνδεση γίνεται με έναν μόνο τρόπο. Στο σχήμα **3.3.3** επιλέχθηκαν οι ακμές (1,2),(2,3),(5,6) για αφαίρεση. Η επανασύνδεση σε αυτή την περίπτωση γίνεται δημιουργώντας τις ακμές (1,3),(2,5),(6,2).



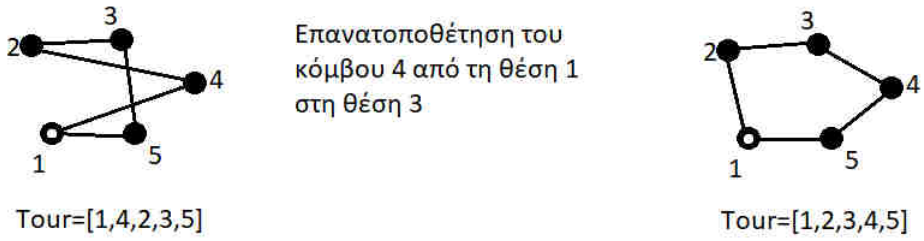
Σχήμα **3.3.3**: Οπτική αναπαράσταση του 3-opt όταν οι 2 ακμές που επιλέγονται είναι συνεχόμενες

Algorithm 5 3-Opt Algorithm

Require: $Tour, DistanceMatrix$
procedure 3-Opt Algorithm($Tour$)
  for $i = 1$ to $Tour(length)$ **do**
    for $j = i + 1$ to $Tour(length)$ **do**
      for $k = j + 1$ to $Tour(length)$ **do**
        if i,j,k not consecutive **then**
          Create new tours T_1,T_2,T_3,T_4
          Newtour ← $cheapestof$ T_1,T_2,T_3,T_4
          if $Cost(NewTour) < Cost(Tour)$ **then**
            $Tour \leftarrow NewTour$
          end if
        end if
      end for
    end for
  end for
  Return Tour
end procedure

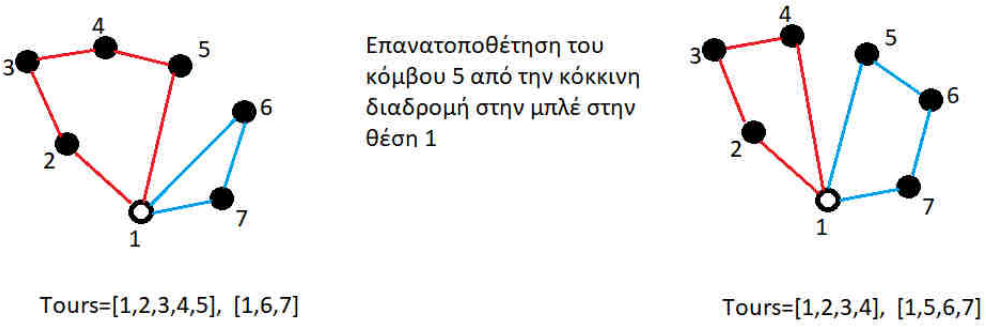
3.4 Επανατοποθέτηση (Relocate)

Η μέθοδος επανατοποθέτησης επιχειρεί να μεταφέρει έναν κόμβο από τη θέση στην οποία βρίσκεται μέσα σε μια διαδρομή σε μία άλλη θέση. Ο κόμβος διαγράφεται από τη θέση που είναι και μαζί του διαγράφονται και οι ακμές που τον ενώνουν με τους γειτονικούς του κόμβους ενώ οι 2 γειτονικοί του κόμβοι συνδέονται με μία νέα ακμή. Στη συνέχεια επιλέγεται ανάμεσα σε ποιους κόμβους θα επανατοποθετηθεί και αφαιρείται η ακμή η ακμή που τους συνδέει ενώ προστίθενται στη διαδρομή οι νέες ακμές που συνδέουν τον κόμβο που μετακινήθηκε με τους νέους του γείτονες. Η διαδικασία φαίνεται στο σχήμα **3.4.4**.



Σχήμα **3.4.4** Επανατοποθέτηση κόμβου μέσα στην ίδια διαδρομή

Όταν θέλουμε οχι απλά να τοποθετήσουμε σε άλλη θέση έναν κόμβο αλλά να τον εντάξουμε σε μία άλλη διαδρομή, πρέπει απαραίτητα να γίνει έλεγχος εφικτότητας επανατοποθέτησης, διότι είναι πιθανό να μην χωράει στην διαδρομή λόγω της περιορισμένης χωρητικότητας. Εάν για παράδειγμα οι κόμβοι μιας διαδρομής έχουν συνολική ζήτηση 90 και το όχημα έχει χωρητικότητα 100 τότε είναι αδύνατον να τοποθετηθεί στη διαδρομή ένας κόμβος με ζήτηση 15 γιατί έτσι θα παραβιάζεται ο περιορισμός χωρητικότητας.



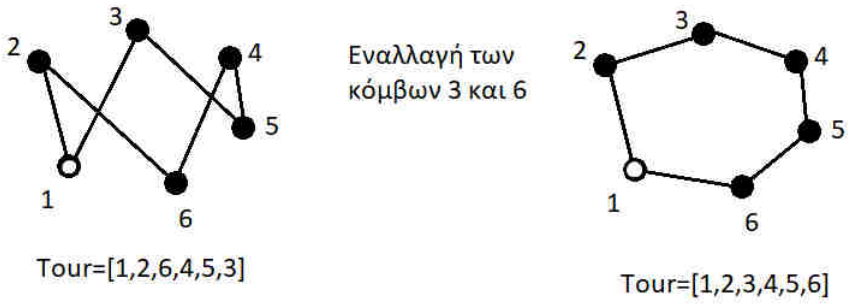
Σχήμα **3.4.5** Επανατοποθέτηση κόμβου σε διαφορετική διαδρομή

Η μέθοδος της επανατοποθέτησης μπορεί να επεκταθεί με το να επιχειρήσουμε να επανατοποθετήσουμε ένα ολόκληρο κομμάτι μιας διαδρομής. Δηλαδή μπορεί να επιλεχθεί ένας αριθμός κόμβων που είναι συνεχόμενοι σε μία διαδρομή και να αποκοπεί αυτούσιος και να τοποθετηθεί σε ένα άλλο σημείο της ίδιας ή διαφορετικής διαδρομής. Για παράδειγμα στη διαδρομή 2,4,6,3,7,5,9 μπορούμε να αποκόψουμε το τμήμα 6,3,7 και να το επανατοποθετήσουμε όπως είναι σε ένα άλλο σημείο της διαδρομής. Επιλέγουμε την προτελευταία θέση και έτσι προκύπτει η διαδρομή 2,4,5,6,3,7,9.

3.5 Ανταλλαγή (Swap)

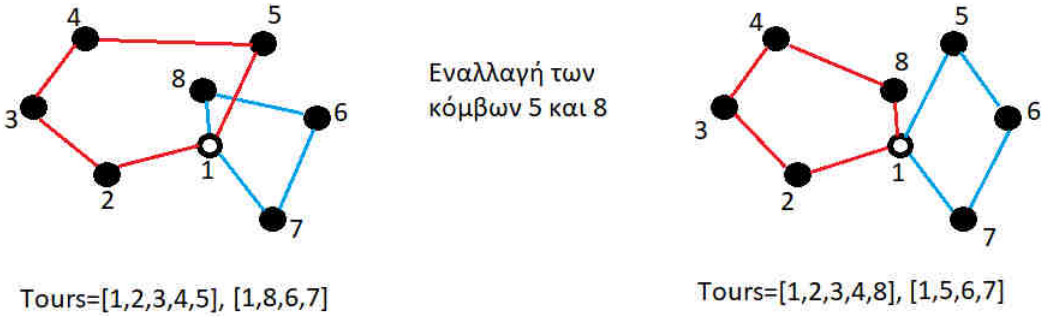
Η μέθοδος ανταλλαγής επιχειρεί να ανταλλάξει τη θέση δύο κόμβων σε μια δεδομένη δρομολόγηση. Ο κόμβοι διαγράφονται από τις θέσεις που είναι και μαζί τους διαγράφονται και οι ακμές

που τους ενώνουν με τους γειτονικούς τους κόμβους. Στη συνέχεια επανατοποθετούνται ο ένας στη θέση του άλλου και προστίθενται οι κατάλληλες νέες ακμές που συνδέουν τους κόμβους που αντάλλαξαν θέσεις με τους νέους τους γείτονες. Η διαδικασία φαίνεται στο σχήμα **3.5.6**.



Σχήμα **3.5.6** Εναλλαγή κόμβων μέσα στην ίδια διαδρομή

Φυσικά όταν επιχειρούμε την ανταλλαγή κόμβων που ανήκουν σε διαφορετικές διαδρομές τότε πρέπει να γίνει έλεγχος εφικτότητας της ανταλλαγής αφού αν δεν τηρούνται οι περιορισμοί χωρητικότητας η ανταλλαγή δεν μπορεί να πραγματοποιηθεί.



Σχήμα **3.5.7** Εναλλαγή κόμβων από διαφορετικές διαδρομές

Όπως και η μέθοδος της επανατοποθέτησης, έτσι και η μέθοδος της ανταλλαγής μπορεί να επεκταθεί με το να επιχειρήσουμε να ανταλλάξουμε ολόκληρα κομμάτια μιας διαδρομής. Δηλαδή μπορούν να επιλεχθούν δύο σύνολα κόμβων που είναι συνεχόμενοι σε μία διαδρομή και να γίνει ανταλλαγή των συνόλων αυτών. Βέβαια τα σύνολα αυτά μπορούν να ανηκουν στην ίδια η σε διαφορετική διαδρομή. Για παράδειγμα στη διαδρομή 2,4,6,3,7,5,9 μπορούμε να ανταλλάξουμε τα τμήματα 4,6,3 και 7,5. Έτσι προκύπτει η διαδρομή 2,7,5,4,6,3,9.

3.6 Αλγόριθμοι πρώτης καθόδου και καλύτερης καθόδου

Όλες οι τεχνικές τοπικής αναζήτησης που περιγράφηκαν παραπάνω αποτελούν στην ουσία ένα σύνολο κανόνων για την κάθε μία από αυτες. Αυτό το σύνολο κανόνων μας δείχνει ποιες είναι οι πιθανές κινήσεις όταν έχουμε μια αρχική λύση x. Για παράδειγμα η γειτονιά ανταλλαγής είναι όλες οι πιθανές ανταλλαγές κόμβων. Ο αλγόριθμος ψάχνει να βρει μέσα σε αυτό το σύνολο κανόνων μια πιθανή βελτίωση της αρχικής λύσης x. Φυσικά εάν δεν βρεθεί βελτίωση ο αλγόριθμος σταματάει αφού έχει εγκλωβιστεί. Υπάρχουν γενικά δύο βασικοί τρόποι για να γίνει η αναζήτηση ανεξαρτήτως γειτονιάς. Πρόκειται για τους αλγορίθμους πρώτης καθόδου και καλύτερης καθόδου.

Ο αλγόριθμος πρώτης καθόδου λαμβάνει ως είσοδο την γειτονιά (δηλαδή το σύνολο κανόνων της) και ψάχνει μέχρι να βρει μία βελτίωση. Όταν τη βρει σταματάει και επιστρέφει την νέα λύση κάνοντας την κίνηση που βρήκε ότι προσφέρει βελτίωση μειώνοντας (ή αυξάνοντας αν έχουμε πρόβλημα μεγιστοποίησης) την τιμή της αντικειμενικής συνάρτησης. Στον ψευδοκώδικα 6 που ακολουθεί περιγράφεται η διαδικασία.

Algorithm 6 First Improvement

Require: *Tour* x, *Neighborhood* $N(x)$
1: **procedure** First Improvement Algorithm(*Tour* x, *Neighborhood* $N(x)$)
2: **for** x' in $N(x)$ **do**
3: **if** $f(x') < f(x)$ **then**
4: Return x'
5: **end if**
6: **end for**
7: Return x
8: **end procedure**

Η δεύτερη στρατηγική που μπορεί να ακολουθηθεί είναι ο αλγόριθμος καλύτερης κατάβασης κατά τον οποίο η γειτονιά εξερευνείται εξαντλητικά για να βρεθεί η καλύτερη δυνατή κίνηση που μπορεί να προσφέρει βελτίωση. Είναι εμφανές ότι η μέθοδος αυτή απαιτεί περισσότερο χρόνο αφού θα πρέπει να εξετάσει ενδελεχώς ολόκληρο το σύνολο των γειτονικών λύσεων όσο μεγάλο και αν είναι αυτό, όμως εγγυάται μια καλύτερης ποιότητας βελτίωση στην τιμή της αντικειμενικής συνάρτησης. Στον ψευδοκώδικα 7 που ακολουθεί περιγράφεται η διαδικασία.

Algorithm 7 Best Improvement

Require: $Tour\ x, Neighborhood\ N(x)$
1: **procedure** BEST IMPROVEMENT ALGORITHM($Tour\ x, Neighborhood\ N(x)$)
2: $best \leftarrow x$
3: **for** x' in $N(x)$ **do**
4: **if** $f(x') < f(x)$ **then**
5: $best \leftarrow x'$
6: **end if**
7: **end for**
8: Return best
9: **end procedure**

Και οι δύο αλγόριθμοι εκτελούνται πολλές φορές και όχι μόνο μία. Όταν βρεθεί η βελτίωση που αναζητούμε η διαδικασία ξεκινά πάλι από τη αρχή έχοντας για είσοδο τη νέα λύση που δώθηκε στην προηγούμενη εκτέλεση. Οι επαναλήψεις συνεχίζονται μέχρι να μην προσφέρεται καμία βελτίωση.

Σύμφωνα με υπολογιστικές μελέτες που έχουν διεξαχθεί (Hansen and Mladenović 2006, Ochoa, Verel, and Tomassini 2010) η καλύτερη κάθοδος δεν είναι πάντα η καλύτερη δυνατή προσέγγιση. Οι έρευνες έδειξαν ότι όταν σε κάποιο πρόβλημα ξεκινάμε με μια αρχική λύση που προήλθε από κάποια τυχαία γεννήτρια λύσεων τότε είναι προτιμότερο να επιχειρείται βελτιστοποίηση με τον αλγόριθμο πρώτης κατάβασης. Όταν όμως ξεκινάμε δημιουργώντας μια λύση με κάποιον αλγόριθμο κατασκευής έστω και αν αυτός ο αλγόριθμος είναι άπληστος και δεν μας δίνει ιδιαίτερα ποιοτικά αποτελέσματα τότε ο αλγόριθμος καλύτερης κατάβασης ενδείκνυται για την βελτιστοποίηση της λύσης αυτής. Το συμπέρασμα αυτό προκύπτει από τις υπολογιστικές μελέτες που έχουν γίνει σε αυτό το ερευνητικό πεδίο και δεν υπάρχει κάποια αυστηρή μαθηματική απόδειξη για να υποστηρίξει τον εν λόγω ισχυρισμό.

ΚΕΦΑΛΑΙΟ 4

Μεθευρετικές μέθοδοι

4.1 Εισαγωγή

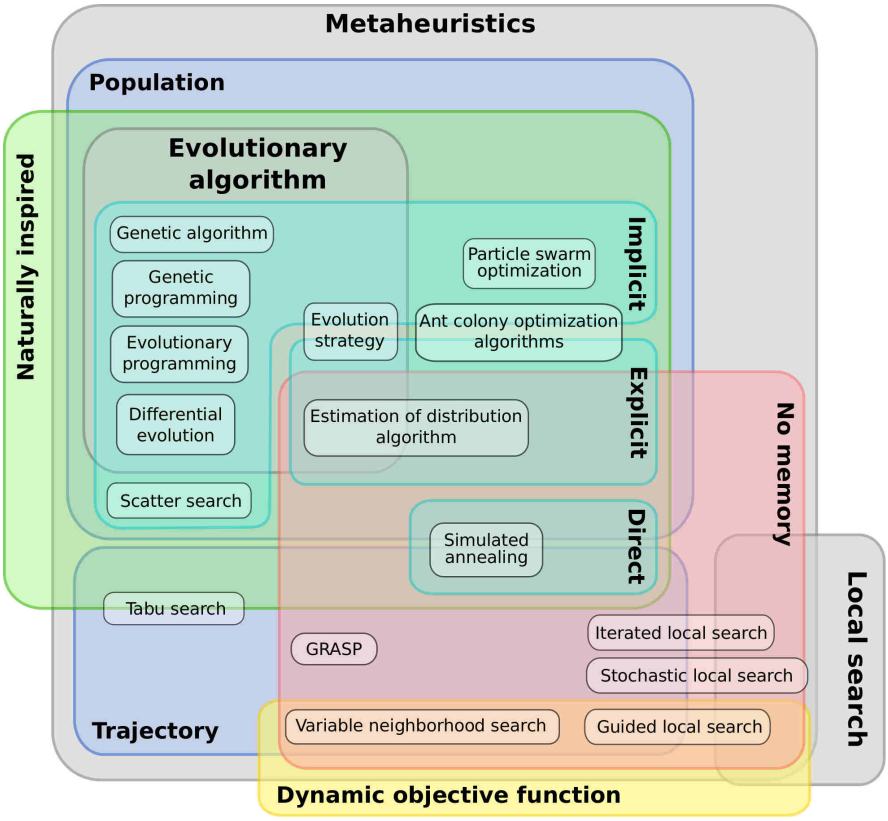
Στην επιστήμη των υπολογιστών, της τεχνητής νοημοσύνης και της μαθηματικής βελτιστοποίησης οι ευρετικές είναι τεχνικές κατάλληλα σχεδιασμένες για την επίλυση υπολογιστικά δύσκολων προβλημάτων. Σε τέτοιου είδους προβλήματα οι κλασικοί ακριβείς αλγόριθμοι είναι πολύ αργοί λόγω του μεγάλου πλήθους υπολογισμών που απαιτούνται για την εύρεση της βέλτιστης λύσης του προβλήματος. Έτσι οι ακριβείς αλγόριθμοι μπορούν να αντικατασταθούν απο ευρετικές μεθόδους οι οποίες επιχειρούν να φτάσουν όσο το δυνατόν πιο κοντά στη βέλτιστη λύση θυσιάζοντας την ακρίβεια για την ταχύτητα. Μια ευρετική μέθοδος αξιολογεί εναλλακτικές λύσεις σε κάθε βήμα ενός αλγορίθμου αναζήτησης. Σε πολλά προβλήματα η βέλτιστη λύση είναι δυνατόν να υπολογιστεί απο ευρετικές μεθοδολογίες ενώ σε άλλα η εύρεση της είναι πιο πολύπλοκη επιτρέποντας μας μόνο να την προσεγγίσουμε σε ένα λογικό χρονικό περιθώριο (Pearl 1984).

Ο στόχος μια ευρετικής μεθόδου είναι να λύσει ένα απαιτητικό υπολογιστικά πρόβλημα σε σύντομο χρονικό διάστημα. Η λύση που θα επιστρέψει ένας ευρετικός αλγόριθμος πρέπει να λύνει το πρόβλημα όμως δεν μπορεί να εγγυηθεί ότι θα βρει το ολικό βέλτιστο του προβλήματος. Είναι πολύ πιθανό η λύση αυτή να μην είναι η καλύτερη δυνατή από το σύνολο όλων των πιθανών λύσεων, ή απλά να είναι μια προσέγγιση της βέλτιστης λύσης. Παρόλα αυτά, αυτό που καθιστά πολύτιμη μια λύση που υπολογίστηκε με τη βοήθεια ευρετικών μεθοδολογιών, είναι η ταχύτητα με την οποία υπολογίστηκε. Οι ευρετικές τεχνικές είναι κατά κανόνα πολύ πιο γρήγορες σε εκτέλεση από τους συνηθισμένους ακριβείς αλγορίθμους.

Μια από τις κυριότερες κατηγορίες ευρετικών μεθόδων είναι οι ευρετικοί αλγόριθμοι αναζήτησης για την εύρεση του ολικού βέλτιστου στο ζητούμενο πρόβλημα ή μιας λύσης που να βρίσκεται όσο το δυνατόν πιο κόντα στο ολίκο βέλτιστο το οποίο αναζητούμε.

Ένας από τους όρους που συναντάμε συχνά μελετώντας για τις ευρετικές μεθόδους είναι οι μεθευρετικές μεθοδολογίες. Οι μεθευρετικές μεθοδολογίες αποτελούν γενικά πλαίσια εφαρμογής των κλασικών ευρετικών μεθοδολογιών τα οποία μπορούν εύκολα να προσαρμοστούν αλγοριθμικά

έτσι ώστε να έχουν τη δυνατότητα να εφαρμοστούν σε κάθε πρόβλημα. Πρόκειται για γενικού τύπου μεθοδολογίες οι οποίες έχουν τεράστιο πεδίο εφαρμογής και πληθώρα τρόπων με τους οποίους μπορούν να προσεγγίσουν ένα πρόβλημα. Χαρακτηριστικά παραδείγματα μεθευρετικών αλγορίθμων είναι η προσομοιωμένη ανόπτηση (simulated annealing), η αναζήτηση ταμπού (taboo search), η αναζήτηση μεταβαλλόμενης γειτνίασης (variable neighborhood search - VNS), οι γενετικοί αλγόριθμοι (genetic algorithms - GM) και διάφοροι αλγόριθμοι εμπνευσμένοι από τη φύση όπως ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών (ant colony optimization - ACO) και η βελτιστοποίηση σμήνους σωματιδίων (Particle Swarm Optimization - PSO). Η αναζήτηση μεταβαλλόμενης γειτνίασης είναι ο αλγόριθμος που επιλέχθηκε για την υλοποίηση στο πλαίσιο της παρούσας διπλωματικής εργασίας και θα μελετηθεί ξεχωριστά και αναλυτικότερα στο επόμενο κεφάλαιο. Στο παρακάτω σχήμα κατηγοριοποιούνται οι δημοφιλείς μεθευρετικοί αλγόριθμοι με βάση κάποια βασικά χαρακτηριστικά. Τα χαρακτηριστικά αυτά είναι το αν βασίζονται σε μια αρχική λύση ή σε ένα ολόκληρο σύνολο λύσεων ενός προβλήματος όπως οι γενετικοί, αν είναι εμπνευσμένοι από τη φύση όπως η ACO και η simulated annealing, αν βασίζονται στην τοπική αναζήτηση και άλλα. Στο παρακάτω σχήμα **4.1.1** βλέπουμε μια οπτική αναπαράσταση της κατηγοριοποίησης διαφόρων μεθευρετικών. (Πηγή εικόνας: `https://en.wikipedia.org/wiki/Metaheuristic`)



Σχήμα **4.1.1** Κατηγοριοποίηση μεθευρετικών μεθόδων

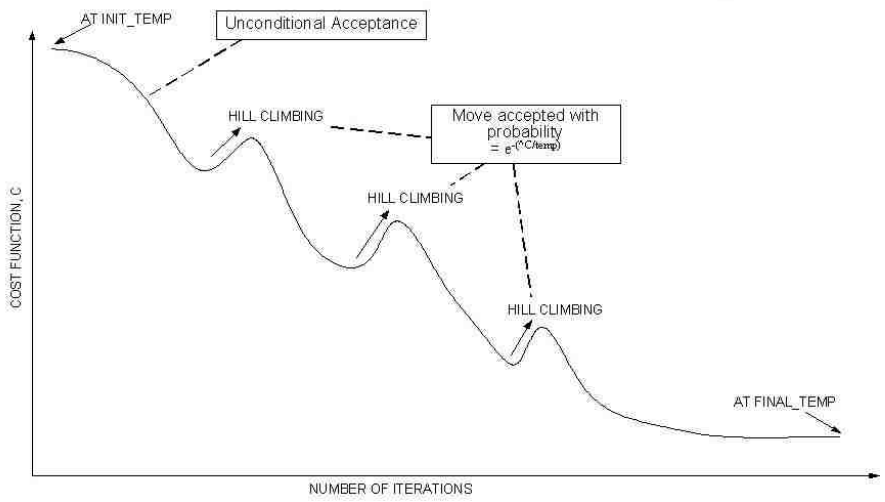
4.2 Προσομοιωμένη ανόπτηση

Σε πολλές ευρετικές μεθόδους (όπως η αναρρίχηση λόφων) υπάρχει ο κίνδυνος ο αλγόριθμος να εγκλωβιστεί σε κάποιο τοπικό βέλτιστο και έτσι να μην είναι δυνατό να βρεθεί μια ενδεχόμενα καλύτερη λύση. Από την άλλη μεριά όμως δεν επιθυμούμε να επιλέγεται εντελώς τυχαία το αν θα κρατήσουμε μια λύση. Μια μεθοδολογία που επιχειρεί να ξεπεράσει αυτά τα εμπόδια και να ισορροπήσει μεταξύ των δυο αυτών άκρων είναι η προσομοιωμένη ανόπτηση. Η μεθοδολογία αυτή αποτελεί ένα παράδειγμα χρήσης μεθόδων στης στατιστικής μηχανικής στον τομέα της βελτιστοποίησης μεγάλων και πολύπλοκων υπολογιστικών προβλημάτων. Παρουσιάστηκε το 1983 από τους (Kirkpatrick, Gelatt, and Vecchi 1983) και είναι εμπνευσμένη από την τεχνική της ανόπτησης των μετάλλων.

Η ανόπτηση των μετάλλων είναι μία διαδικασία που χρησιμοποιείται στην επιστήμη της μεταλλουργίας με σκοπό να μαλακώσουμε ή να σκληρύνουμε μέταλλα θερμαίνοντάς τα σε πολύ υψηλές θερμοκρασίες και στη συνέχεια ψύχοντας τα σταδιακά. Η διαδικασία αυτή έχει σαν αποτέλεσμα το υλικό να στερεοποιηθεί σε μία κρυσταλλική κατάσταση χαμηλής ενέργειας (Aarts and Korst 1988). Όπως αναφέρεται στο υπάρχει μια σχέση μεταξύ της επιστήμης της στατιστικής μηχανικής και της υπολογιστικής βελτιστοποίησης σε προβλήματα πολλών μεταβλητών.

Η γενική ιδέα της προσομοιωμένης ανόπτησης είναι ότι ξεκινάμε με κάποια αρχική λύση και σε κάθε επανάληψη επιλέγεται μια τυχαία κίνηση. Σε περίπτωση που αυτή η κίνηση οδηγεί σε βελτίωση τότε την κρατάμε. Εάν όχι πάλι μπορεί να γίνει αποδεκτή με βάση κάποια πιθανότητα για να αποφύγουμε τον εγκλωβισμό σε τοπικό βέλτιστο. Η πιθανότητα αυτή εξαρτάται από δύο παράγοντες. Ο ένας είναι το πόσο χειροτέρεψε η επιλογή αυτή την προηγούμενη κατάσταση και ο άλλος είναι η παράμετρος θερμοκρασίας T.

Η παράμετρος θερμοκρασίας T είναι μια μεταβλητή η οποία στην αρχή παίρνει μεγάλες τιμές και στη συνέχεια μειώνεται όσο περνάνε οι επαναλήψεις.Έτσι όταν το T μικρύνει αρκετά μετά από κάποιο αριθμό επαναλήψεων τότε θα γίνεται όλο και πιο δύσκολο να κρατήσουμε μια λύση που είναι χειρότερη από την αρχική. Αποδεικνύεται πως για αρκετά μικρές αρχικές τιμές του T η μέθοδος της προσομοιωμένης ανόπτησης τείνει να συμπεριφέρεται σαν αναρρίχηση λόφων όπου μία λύση γίνεται αποδεκτή μόνο αν είναι καλύτερη από την προηγούμενη, και έτσι είναι πολύ πιθανός ο εγκλωβισμός σε κάποιο τοπικό βέλτιστο. Αντίθετα για μεγάλες αρχικές τιμές του T η μέθοδος συμπεριφέρεται σαν μια απλή τυχαία αναζήτηση. Η παράμετρος θερμοκρασίας T μειώνεται σε κάθε επανάληψη του αλγορίθμου. Υπάρχουν πολλές διαφορετικές συναρτήσεις που έχουν χρησιμοποιηθεί για την αποτελεσματική μείωση του T (Van Laarhoven and Aarts 1987). Ενδεικτικά η μείωση μπορεί να γίνεται γραμμικά $T(t) = T_0 - b * t$, λογαριθμικά $T(t) = \frac{T_0}{logt}$ ή εκθετικά $T(t) = T_0 * a^t$. (Πηγή εικόνας: `https://www.slideshare.net/idforjoydutta/simulated-annealing-24528483`)



Σχήμα **4.2.2** Σύγκλιση της προσομοιωμένης ανόπτησης

Σε γενικές γραμμές η σωστή επιλογή της αρχικής τιμής της παραμέτρου T αλλά και η επιλογής της κατάλληλης συνάρτησης μείωσης του T είναι μια διαδικασία που απαιτεί πολλούς πειραματισμούς. Εξαρτάται μόνο από το πρόβλημα και από τις διάφορες παραμέτρους αυτού. Όταν εφαρμοστούν όμως οι σωστές τιμές για την παραμετροποίηση του αλγορίθμου πάνω σε ένα συγκεκριμένο πρόβλημα, αποδεικνύεται ότι η μέθοδος της προσομοιωμένης ανόπτησης μπορεί να υπολογίσει ακόμα και το ολικό βέλτιστο του προβλήματος.

Algorithm 8 Αλγόριθμος Simulated Annealing

Require: $MaxTemp, MaxIterations$
procedure SIMULATED ANNEALING($MaxTemp, MaxIterations$)
  $Scurrent \leftarrow CreateInitialSolution(ProblemSize)$
  $Sbest \leftarrow Scurrent$
  for $i = 1$ to $MaxIterations$ **do**
    $Si \leftarrow CreateNeighborSolution(Scurrent)$
    $CurrentTemp \leftarrow CalculateTemperature(i, MaxTemp)$
    if $Cost(Si) < Cost(Scurrent)$ **then**
      $Scurrent \leftarrow Si$
      if $Cost(Si) < Cost(Sbest)$ **then**
        $Sbest \leftarrow Si$
      end if
    else
      if $Random(0,1) < AcceptProbability(Cost(Si), CurrentTemp)$ **then**
        $Scurrent \leftarrow Si$
      end if
    end if
  end forreturn Sbest
end procedure

Η μέθοδος της προσομοιωμένης ανόπτησης αν και μετρά περισσότερα από 35 χρόνια από τότε που παρουσιάστηκε για πρώτη φορά συνεχίζει ακόμα να κεντρίζει το ενδιαφέρον των ερευνητών που

με διάφορες παραλλαγές προσπαθούν να πετύχουν καλύτερα αποτελέσματα και μικρότερους χρόνους εκτέλεσης. Ενδεικτικά διερευνώνται παράλληλες υλοποιήσεις (Wang, Mu, Zhao, and Sutherland 2015) αλλά και πιο σύγχρονες επεκτάσεις της μεθόδου (Vincent, Redi, Hidayat, and Wibowo 2017).

4.3 Αναζήτηση Tabu

Η αναζήτηση Tabu, που δημιουργήθηκε από τον Fred W. Glover το 1986 και επισημοποιήθηκε το 1989 (Glover 1989), είναι μια μεθευρετική μέθοδος αναζήτησης που χρησιμοποιεί μεθόδους τοπικής αναζήτησης με σκοπό τη μαθηματική βελτιστοποίηση.

Οι τοπικές αναζητήσεις σε μια γειτονιά, δηλαδή σε ένα σύνολο γειτονικών λύσεων, παίρνουν σαν είσοδο μια λύση του προβλήματος και αναζητούν στους άμεσους γείτονές του (δηλαδή λύσεις που είναι παρόμοιες και μπορούμε να φτάσουμε σε αυτές εφαρμόζοντας κάποιον συγκεκριμένο τελεστή αναζήτησης) ελπίζοντας να βρεθεί μια βελτιωμένη λύση της ήδη υπάρχουσας λύσης. Είναι συνηθισμένο φαινόμενο οι μέθοδοι τοπικής αναζήτησης να εγκλωβίζονται σε τοπικά βέλτιστα από τα οποία είναι δύσκολο να ξεφύγουν διότι οι πιθανά καλύτερες ποιοτικά λύσεις βρίσκονται αρκετά μακριά.

Η αναζήτηση Tabu ενισχύει την απόδοση της απλής τοπικής αναζήτησης χαλαρώνοντας τον βασικό της κανόνα ο οποίος είναι να μην γίνονται αποδεκτές χειρότερες λύσεις (de Werra and Hertz 1989). Κατ άρχάς, σε κάθε βήμα μπορούν να γίνουν αποδεκτές κινήσεις χειροτέρευσης εάν δεν υπάρχει διαθέσιμη βελτίωση (όπως όταν η αναζήτηση είναι κολλημένη σε αυστηρό τοπικό βέλτιστο). Επιπλέον, οι απαγορεύσεις (από εκεί προέρχεται ο όρος Tabu) εισάγονται για να αποφευχθεί η περίπτωση να γυρίσουμε πίσω στις χειρότερες λύσεις που είχαν προηγουμένως επισκεφθεί αλλά δεν απέδωσαν στη συνέχεια κάποια βελτίωση της κατάστασης.

Η εφαρμογή της αναζήτησης Tabu χρησιμοποιεί δομές μνήμης που περιγράφουν τις λύσεις επισκέψεων ή σύνολα κανόνων που παρέχονται από το χρήστη. Αν μια εφικτή λύση έχει ήδη εξεταστεί από τον αλγόριθμο προηγουμένως εντός ενός συγκεκριμένου χρονικού διαστήματος ή εάν έχει παραβιάσει έναν κανόνα τότε εισάγεται στη λίστα Tabu (απαγορευμένη), έτσι ώστε ο αλγόριθμος να μην επιστρέψει ξανα σε αυτή τη λύση.

Η αναζήτηση Tabu χρησιμοποιεί την τοπική αναζήτηση για να μετακινηθεί από μια πιθανή λύση x σε μια ενδεχομέχως καλύτερη ποιοτικά λύση x' στη γειτονιά του x, μέχρι έχει ικανοποιηθεί κάποιο κριτήριο διακοπής (χρονικό όριο ή όριο ποιότητας λύσης). Οι διαδικασίες τοπικής αναζήτησης συχνά σταματούν σε περιοχές με χαμηλή βαθμολογία (τοπικά ελάχιστα) ή σε περιοχές όπου σημειώνεται οροπέδιο (τοπικά μέγιστα) ανάλογα με τη φύση του προβλήματος, επειδή δεν μπορούν να εντοπίσουν κάποια βελτίωση που να βρίσκεται σχετικά κοντά. Για να αποδράσει από αυτές τις παγίδες και να εξερευνήσει νέες περιοχές του χώρου αναζήτησης που ενδεχομένως θα μείνουν ανεξερεύνητες από

άλλες διαδικασίες τοπικής αναζήτησης, η αναζήτηση Tabu διερευνά διεξοδικά τη γειτονιά κάθε εφικτής λύσης καθώς λαμβάνει χώρα η αναζήτηση. Οι λύσεις που γίνονται δεκτές στη νέα γειτονιά, καθορίζονται από τη χρήση δομών μνήμης, $N^*(x)$. Χρησιμοποιώντας αυτές τις δομές μνήμης, η αναζήτηση εξελίσσεται μεταβαίνοντας από την τρέχουσα λύση x σε μια βελτιωμένη λύση x' στο $N^*(x)$.

Αυτές οι δομές μνήμης σχηματίζουν αυτό που είναι γνωστό ως λίστα Tabu, ένα σύνολο κανόνων και απαγορευμένες λύσεις που χρησιμοποιούνται για να φιλτράρουν ποιες λύσεις θα γίνουν δεκτές στη γειτονιά $N^*(x)$ που πρέπει να διερευνηθεί από την αναζήτηση, και ποιες θα απορριφθούν. Στην απλούστερη μορφή της, μια λίστα Tabu είναι λίστα από εφικτές λύσεις που έχουμε ξαναεπισκεφθεί στο πρόσφατο παρελθόν (λιγότερο από n πριν από τις επαναλήψεις, όπου n είναι ο αριθμός των προηγούμενων λύσεων που πρέπει να αποθηκευτούν). Συνηθέστερα, μια λίστα Tabu αποτελείται από λύσεις που έχουν αλλάξει από τη διαδικασία της μετάβασης από τη μια λύση στην άλλη. Είναι βολικό, για ευκολία στην περιγραφή, να κατανοήσουμε μια λύση που θα κωδικοποιείται και θα εκπροσωπείται από τέτοια χαρακτηριστικά.

Algorithm 9 Αλγόριθμος αναζήτησης Tabu

Require: $DISTANCE\ MATRIX, DEMAND, CAPACITY$
procedure TABU SEARCH($DISTANCE\ MATRIX, DEMAND, CAPACITY$)
  $sBest \leftarrow s0$
  $bestCandidate \leftarrow s0$
  $tabuList \leftarrow [\,]$
  $tabuList.push(s0)$
  while not stoppingCondition() **do**
    $sNeighborhood \leftarrow getNeighbors(bestCandidate)$
    for sCandidate in sNeighborhood **do**
      if (tabuList does not contain sCandidate) and cost(sCandidate) > cost(bestCandidate) **then**
        $bestCandidate \leftarrow sCandidate$
      end if
    end for
    if $fitness(bestCandidate) > fitness(sBest)$ **then**
      $sBest \leftarrow bestCandidate$
    end if
    $tabuList.push(bestCandidate)$
    if tabuList.size > maxTabuSize **then**
      $tabuList.removeFirst()$
    end if
  end while
  return $sBest$
end procedure

Ξεκινώντας ο αλγόριθμος αρχικοποιεί τη διαδικασία δημιουργώντας μια κενή λίστα Tabu. Στη συνέχεια καλεί κάποια ευρετική μέθοδο κατασκευής αρχικής λύσης εάν αυτό είναι δυνατό ή αλλιώς δημιουργεί μια αρχική λύση με σχεδόν τυχαίους κανόνες. Έπειτα θέτει την αρχική αυτή λύση ώς την καλύτερη που έχει μέχρι στιγμής βρεθεί και την προσθέτει στη λίστα Tabu έτσι ώστε να μην επιστρέψει σε αυτήν. Η κεντρική δομή επανάληψης της μεθόδου ξεκινά την έρευνα σε γειτονικές

περιοχές μέχρι να συναντήσει κάποιο κριτήριο τερματισμού όπως το έχει ορίσει ο χρήστης (Hertz and de Werra 1990, Gendreau, Hertz, and Laporte 1994). Κατά τη διάρκεια της διαδικασίας της έρευνας για τη βέλτιστη λύση προστιθενται συνέχεια λύσεις στη λίστα Tabu και έτσι ο αλγόριθμος έχει ώς στόχο να φτάσει αρκετά βαθιά στις γειτονιές αυτές έτσι ώστε να εντοπίσει αν είναι δυνατόν το ολικό βέλτιστο. Ο ψευδοκώδικας 9 περιγράφει τη βασικής μορφής μιας αναζήτησης Tabu.

Η αναζήτηση Tabu αν και είναι από τις πρώτες μεθευρετικές που διατυπώθηκαν παραμένει και σήμερα μια από τις πιο δημοφιλείς μέθόδους για την επίλυση προβλημάτων δρομολόγησης. Συνδυάζοντας στοιχεία από άλλες μεθευρετικές και με την κατάλληλη επιλογή ρυθμίσεων και παραμέτρων μπορεί να αποδόσει εξαιρετικά αποτελέσματα (Sicilia, Quemada, Royo, and Escuín 2016, Lai, Demirag, and Leung 2016) και να προσαρμοστεί σε πολλά είδη προβλημάτων δρομολόγησης.

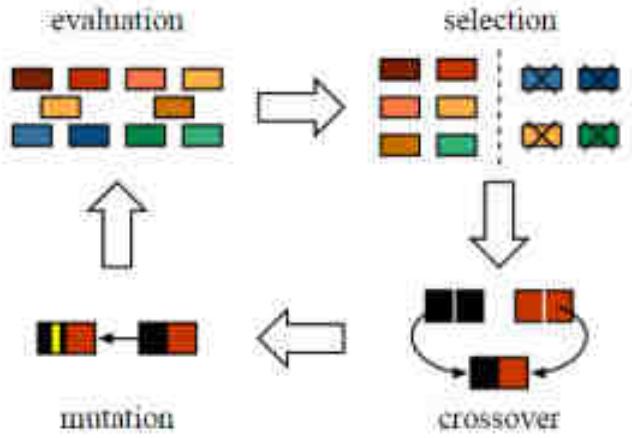
4.4 Ο γενετικός αλγόριθμος

Στην επιστήμη των υπολογιστών και την επιχειρησιακή έρευνα, ο γενετικός αλγόριθμος είναι μια μεθευρετική μέθοδος εμπνευσμένη από την φυσική επιλογή που συμβαίνει στη φύση και οδηγεί στην εξέλιξη όλων των ζωντανών οργανισμών και ανήκει σε μια μεγάλη κλάση αλγορίθμων που ονομάζονται εξελικτικοί αλγόριθμοι. Τέτοιου είδους αλγόριθμοι χρησιμοποιούνται ευρέως για την εύρεση λύσεων σε κάθε είδους υπολογιστικά δύσκολων προβλημάτων βελτιστοποίησης και αναζήτησης. Βασίζεται σε τελεστές επιλογής, μετάλλαξης και συνδυασμού όπως ακριβώς παρατηρείται και στη φύση σύμφωνα με την θεωρία της εξέλιξης του Δαρβίνου (Glover and Kochenberger 2006).

Σε αντίθεση με άλλες μεθευρετικές οι γενετικοί αλγόριθμοι χρησιμοποιούν ένα σύνολο λύσεων αντί για μία αρχική λύση την οποία βέλτιώνουν. Το σύνολο αυτό λύσεων ονομάζεται πληθυσμός υποψήφιων λύσεων και προκύπτει είτε από ευρετικές μεθόδους κατασκευής είτε, αν το επιτρέπει το πρόβλημα, τύχαια. Αυτό το σύνολο παίρνει σαν είσοδο ο γενετικός αλγόριθμος και προσπαθεί να το εξελίξει, συγκλίνοντας προς την καλύτερη δυνατή λύση. Η σύγκλιση αυτή επιτυγχάνεται με την παραγωγή νέων γενεών οι οποίες είναι ποιοτικά καλύτερες από τις προηγούμενες. Η καταλληλότητα μια λύσης συνήθως αναπαριστάται από την τιμή της αντικειμενικής συνάρτησης την οποία έχουμε θέσει εμείς. Στην περίπτωση της δρομολόγησης οχημάτων η αντικειμενική συνάρτηση είναι το κόστος. Όσο χαμηλότερο είναι το κόστος τόσο πιο κατάλληλη είναι μια υποψήφια λύση. Αφού δημιουργηθεί ο αρχικός πληθυσμός λύσεων, στον οποίο είναι συνηθισμένο να έχουμε εκατοντάδες ή και χιλιάδες υποψήφιους, τότε ελέγχεται ποιες από αυτές τις λύσεις είναι οι καλύτερες. Όσο καλύτερη είναι η τιμή της αντικειμενικής συνάρτησης μιας λύσης τόσο πιο πιθανό είναι να επιλεχθεί η συγκεκριμένη λύση, μέσω μιας συνάρτησης τυχαίας επιλογής, για να συμμετέχει στην παραγωγή της νέας γενιάς. Το επόμενο βήμα αφού συγκεντρωθεί το υποσύνολο λύσεων που θα παράγουν τις νέες λύσεις, είναι η εφαρμογή των γενετικών τελεστών της διασταύρωσης (crossover) και της μετάλλαξης (mutation).

Ο τελεστής της διασταύρωσης είναι η διαδικασία κατά την οποία δύο λύσεις συνδυάζονται για να παράγουν έναν απόγονο. Η διαστάυρωση μπορεί να γίνει με διάφορους τρόπους επιλέγοντας στοιχεία και από τους δύο γονείς για την δημιουργία της νέας λύσης. Ανάλογα με το πρόβλημα η αναπαράσταση των λύσεων μπορεί να είναι λίστες από bit ή λίστες πραγματικών αριθμών ή ακόμη και ολόκληρα δέντρα ή γράφοι. Σε διαφορετικές κατηγορίες προβλημάτων χρησιμοποιούνται και διαφορετικές υλοποιήσεις του τελεστή crossover. Οι πιο συνηθισμένες από αυτές περιλαμβάνουν τον διαχωρισμό των γονέων σε κομμάτια και στη συνέχεια τον συνδυασμό αυτών των κομματιών της αναπαράστασης τους για την δημιουργία του απογόνου. Συνήθως οι γόνεις διαχωρίζονται σε δύο ή σε τρία κομμάτια single point crossover, two point crossover και η νέα λύση προκύπτει από τον συνδυασμό των κομματιών αυτών.

Η μετάλλαξη είναι ένας γενετικός τελεστής που χρησιμοποιείται για τη διατήρηση της γενετικής ποικιλότητας από μια γενιά ενός πληθυσμού στην επόμενη. Κατά τη μετάλλαξη, η λύση μπορεί να αλλάξει εξ ολοκλήρου από την προηγούμενη λύση. Ως εκ τούτου, ο γενετικός αλγόριθμος μπορεί να επιτύχει μια καλύτερη λύση χρησιμοποιώντας τη μετάλλαξη. Η μετάλλαξη λαμβάνει χώρα κατά τη διάρκεια της εξέλιξης σύμφωνα με μια πιθανότητα μετάλλαξης που μπορεί να οριστεί από τον χρήστη. Αυτή η πιθανότητα πρέπει να είναι χαμηλή. Αν η ρύθμιση είναι πολύ υψηλή, η αναζήτηση θα μετατραπεί σε μια πρωτόγονη τυχαία αναζήτηση. Στην εικόνα **4.4.3** δίνεται μια σχηματική αναπαράσταση της λειτουργίας των τελεστών διασταύρωσης και μετάλλαξης (Πηγή εικόνας: `http://www.jade-cheng.com/au/coalhmm/optimization/`).



Σχήμα **4.4.3** Γραφική αναπαράσταση του γενετικού αλγορίθμου

Το κλασικό παράδειγμα ενός χειριστή μετάλλαξης περιλαμβάνει μια πιθανότητα ότι ένα αυθαίρετο bit σε μια γενετική ακολουθία θα αλλάξει από την αρχική του κατάσταση. Μια κοινή μέθοδος εφαρμογής του χειριστή μετάλλαξης περιλαμβάνει τη δημιουργία μιας τυχαίας μεταβλητής για κάθε bit σε μια ακολουθία. Αυτή η τυχαία μεταβλητή δηλώνει εάν ένα συγκεκριμένο κομμάτι θα τροπο-

ποιηθεί ή όχι. Αυτή η διαδικασία μετάλλαξης, με βάση τη βιολογική σημειακή μετάλλαξη, ονομάζεται μετάλλαξη μονού σημείου. Άλλοι τύποι είναι η αντιστροφή και η μετάλλαξη με κινητά σημεία. Όταν η γονιδιακή κωδικοποίηση είναι περιοριστική όπως στα προβλήματα μετάθεσης, οι μεταλλάξεις είναι ssaps, αναστροφές και ανακατασκευές (Eiben, Raue, and Ruttkay 1994).

Ο αλγόριθμος ξεκινά παράγοντας έναν αρκετά μεγάλο πληθυσμό υποψήφιων λύσεων του προβλήματος. Στο επόμενο βήμα γίνεται η αξιολόγηση κάθε λύσης από την τιμή της αντικειμενικής συνάρτησης. Κάθε λύση επιλέγεται με μια πιθανότητα ανάλογη της καταλληλότητας της για αναπαραγωγή. Αφού επιλεχθούν οι καταλληλότεροι υποψήφιοι τότε αρχίζει η διαδικασία του crossover έτσι ώστε να παράγουμε έναν νέο πληθυσμό λύσεων ίσο σε πλήθος με τον προηγούμενο. Υποψηφιες λύσεις από το δείγμα που επιλέχθηκε διασταυρώνονται μεταξύ τους με τυχαίο τρόπο και έτσι προκύπτει ο νέος πληθυσμός. Τώρα κάθε μέλος του νέου πληθυσμού παθαίνει μια μετάλλαξη με μία πιθανότητα που εξαρτάται από παράγοντες που θα θέσει ο χρήστης. Κατά τη μετάλλαξη αλλάζει κάποιο στοιχείο της λύσης χρησιμοποιώντας κάποιον από τους τελεστές βελτιστοποίησης με τυχαίο τρόπο. Έτσι έχοντας έναν νέο πληθυσμό η διαδικασία επαναλαμβάνεται μέχρι να συναντήσουμε τα κριτήρια τερματισμού τα οποία συνήθως είναι ο χρόνος εκτέλεσης, ο αριθμός των γενεών που παράχθηκαν ή αν ο αλγόριθμος έχει βρεί μια λύση που να είναι μέσα στα αποδεκτά όρια σφάλματος που έχει θέσει ο χρήστης.

Algorithm 10 Γενετικός Αλγόριθμος

Require: *StoppingCriteria*
procedure GENETIC ALGORITHM(*StoppingCriteria*)
  Initialize new population
  while StoppingCriteria not met **do**
    for $i = 1$ to $Size(population)$ **do**
      $x \leftarrow RandomSelect(population, FitnessFunction)$
      $y \leftarrow RandomSelect(population, FitnessFunction)$
      $child \leftarrow CrossOver(x, y)$
      mutate child with a small random probability
      add child to new population
    end for
    $population \leftarrow NewPopulation$
  end while
end procedure

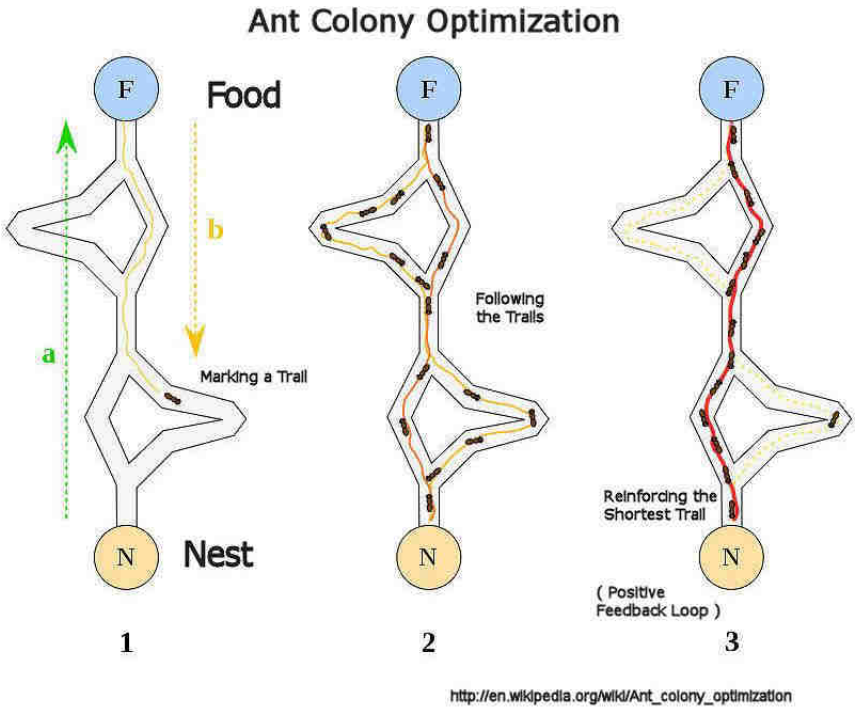
4.5 Αλγόριθμος βελτιστοποίησης αποικίας των μυρμηγκιών

Στην επιστήμη των υπολογιστών και την επιχειρησιακή έρευνα, ο αλγόριθμος βελτιστοποίησης αποικιών μυρμηγκιών (Ant colony optimization - ACO) είναι μια πιθανοκρατική τεχνική για την επίλυση υπολογιστικών προβλημάτων που μπορούν να περιοριστούν στην εύρεση καλών διαδρομών μέσα σε γράφους (Dorigo and Birattari 2010). Είναι μια τεχνική εμπνευσμένη από τη συμπεριφορά των πραγματικών μυρμηγκιών. Η επικοινωνία των μυρμηγκιών με φερομόνες είναι συχνά το

κυρίαρχο πρότυπο που χρησιμοποιείται. Οι συνδυασμοί τεχνητών μυρμηγκιών και οι αλγόριθμοι τοπικής αναζήτησης έχουν γίνει μια μέθοδος επιλογής για πολλές εργασίες βελτιστοποίησης που περιλαμβάνουν κάποιο είδος γραφήματος, π.χ. δρομολόγηση οχημάτων και δρομολόγηση διαδικτύου.

Η βελτιστοποίηση των αποικιών των μυρμηγκιών είναι μια κατηγορία αλγορίθμων βελτιστοποίησης που έχουν σχεδιαστεί με βάση τη δράση και τη λειτουργία μιας πραγματικής αποικίας μυρμηγκιών. Τα τεχνητά «μυρμήγκια» (πράκτορες προσομοίωσης) εντοπίζουν τις βέλτιστες λύσεις με το να κινούνται σε ένα χώρο που περιλαμβάνει όλες τις πιθανές λύσεις. Τα πραγματικά μυρμήγκια αφήνουν φερομόνες στο περασμά τους και έτσι επιτυγχάνουν την μεταξύ τους επικοινωνία και καθοδήγηση ενώ εξερευνούν το περιβάλλον τους. Τα προσομοιωμένα «μυρμήγκια» καταγράφουν ομοίως τις θέσεις τους και την ποιότητα των λύσεών τους, έτσι ώστε σε μεταγενέστερες προσομοιώσεις επαναλήψεις περισσότερα μυρμήγκια να βρουν ακόμα καλύτερες λύσεις (Dorigo and Stützle 2003).

Στον φυσικό κόσμο, κάποια είδη μυρμηγκιών αρχικά περιπλανιούνται τυχαία και όταν βρίσκουν τροφή επιστρέφουν πίσω στην αποικία τους, αφήνοντας φερομόνης στα μονοπάτια που πέρασαν. Αν τα άλλα μυρμήγκια βρουν ίχνη φερορμόνης, είναι πιθανό να μην συνεχίσουν να ταξιδεύουν τυχαία, αλλά να ακολουθήσουν το ίχνος, να επιστρέψουν και να το ενισχύσουν αν βρουν τελικά τροφή. (Πηγή εικόνας: `https://www.sciencedirect.com/science/article/pii/S0142061515005840`).



Σχήμα **4.5.4** Βελτιστοποίηση αποικίας μυρμηγκιών

Με την πάροδο του χρόνου, όμως, το μονοπάτι των φερομονών αρχίζει να εξατμίζεται, μειώνοντας έτσι την πιθανότητα κάποιο μυρμήγκι να το ακολουθήσει. Έτσι αν μια διαδρομή είναι μεγάλη,

μη αποδοτική απαιτεί πολύ χρόνο τότε το ίχνος της φερορμόνης εξατμίζεται και είναι λιγότερο πιθανό κάποιο άλλο μυρμήγκι να ακολουθήσει αυτή τη διαδρομή. Μια σύντομη διαδρομή, για λόγους σύγκρισης, γίνεται πιο ελκυστική διότι το μυρμήγκι που θα την ακολουθήσει θα βρει πιο εύκολα την τροφή και θα επιστρέψει στην αποικία σε μικρό χρονικό διάστημα αφήνοντας ξανά φερομόνη στο μονοπάτι. Έτσι η πυκνότητα φερομόνης γίνεται υψηλότερη σε μικρότερες διαδρομές παρά σε μακρύτερες διαδρομές. Η εξάτμιση της φερομόνης έχει επίσης το πλεονέκτημα της αποφυγής της σύγκλισης προς μια τοπικά βέλτιστη λύση. Αν δεν υπήρχε καθόλου εξάτμιση, τα μονοπάτια που επιλέχθηκαν από τα πρώτα μυρμήγκια θα τείνουν να είναι υπερβολικά ελκυστικά για τα επόμενα. Σε αυτή την περίπτωση, η εξερεύνηση του χώρου της λύσης θα είναι περιορισμένη. Η επίδραση της εξάτμισης των φερομονών στα πραγματικά συστήματα μυρμηγκιών είναι ασαφής, αλλά είναι πολύ σημαντική στα τεχνητά συστήματα (Blum 2005)

Ο αλγόριθμος ξεκινά με την τοποθέτηση όλων των μυρμηγκιών στην έναρξη. Στη συνέχεια κάθε μυρμήγκι χτίζει μια εφικτή λύση επιλέγοντας σε κάθε βήμα ποια θα είναι η επόμενη ακμή από την οποία θα περάσει. Κάθε φορά που περνάει από μια ακμή αφήνει μια ποσότητα φερομόνης. Αφού κατασκευαστούν οι αρχικές Η επιλογή γίνεται εντελώς τυχαία στην αρχή και στη συνέχεια λαμβάνοντας υπόψην την ποσότητα φερομόνης που υπάρχει σε κάθε ακμή, γίνεται πιο ελκυστική (δηλαδή και πιο πιθανή) η επιλογή ακμής με πιο μεγάλες ποσότητες φερομόνης. Σε κάθε βήμα γίνεται ενημέρωση της υπάρχουσας φερομόνης σε κάθε ακμή του γραφήματος. Προχωρώντας, παρατηρούμε σύγκλιση προς τη βέλτιστη λύση ενώ λόγω της εξάτμισης της φερομόνης αποφεύγεται ο εγκλωβισμός σε τοπικά βέλτιστα.

Algorithm 11 Αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών

Require: *Stopping Criteria*
procedure ANT COLONY OTIMIZATION(*Stopping Criteria*)
    Position each ant in a starting node
    while Stopping Criteria not met **do**
        repeat
            for each ant **do**
                choose next node by applying the transition rule
                apply step by step pheromone update
            end for
        until every ant has built a solution
        Update best solution
        apply offline pheromone update
    end while
end procedure

Ο αλγόριθμος βελτιστοποίησης αποικιών μυρμηγκιών αποτελεί μια από τις σημαντικότερες τεχνικές με τις οποίες αντιμετωπίζονται τα προβλήματα δρομολόγησης. Υπάρχει μια τεράστια πληθώρα παραλλαγών και τροποποιήσεων της αρχικής μεθόδου έτσι ώστε να παίρνουμε όλο και καλύτερα αποτελέσματα και αποτελέι ένα πεδίο έρευνας που σίγουρα θα απασχολήσει πολλούς ερευνητές και

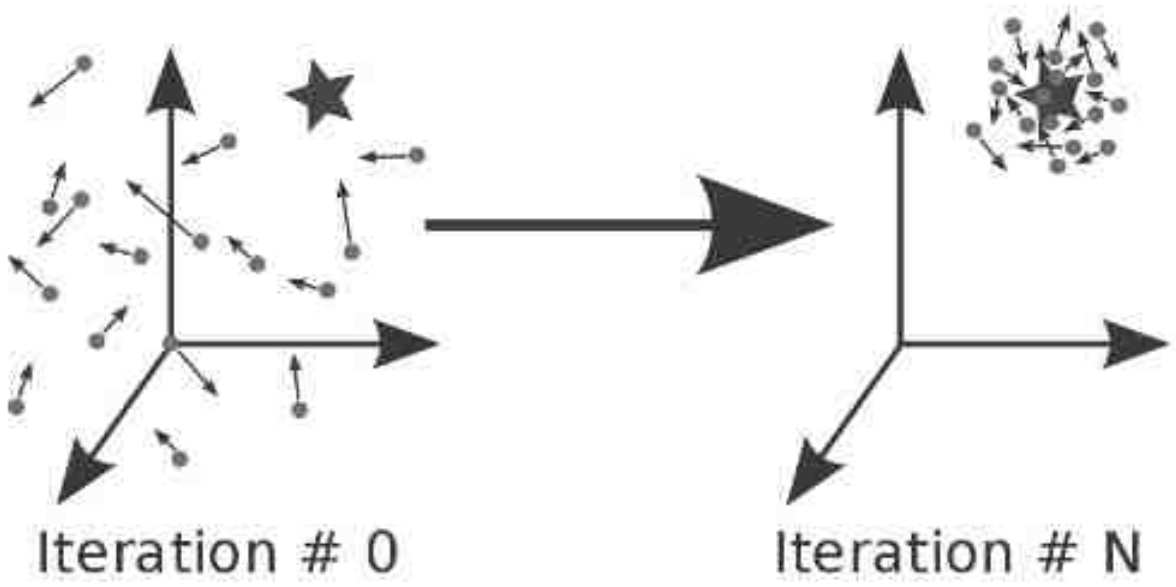
στο μέλλον. Ενδεικτικά έχουν εξεταστεί παράλληλες υλοποιήσεις του αλγορίθμου (Ting and Chen 2013, Doerner, Hartl, Kiechle, Lucka, and Reimann 2004) αλλά και βελτιωμένες εκδοχές του αρχικού αλγορίθμου με εξαιρετικά αποτελέσματα (Yu, Yang, and Yao 2009).

4.6 Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων

Η PSO αποδίδεται αρχικά στους (Kennedy 2010) και αρχικά προορίζεται για την προσομοίωση της κοινωνικής συμπεριφοράς ως σχηματοποιημένη αναπαράσταση της κίνησης των οργανισμών σε ένα σμήνος πουλιών ή κοπάδι ψαριών. Ο αλγόριθμος απλοποιήθηκε και παρατηρήθηκε ότι επιτελούσε βελτιστοποίηση. Το βιβλίο των Kennedy, Eberhart (Kennedy 2006) περιγράφει πολλές φιλοσοφικές πτυχές της PSO και της ευφυΐας σμήνους.

Η PSO είναι μια μεθευρετική μέθοδος δεδομένου ότι κάνει ελάχιστες ή καθόλου υποθέσεις σχετικά με το πρόβλημα που έχει να βελτιστοποιήσει και μπορεί να κάνει αναζήτηση ολικών βέλτιστων σε πολύ μεγάλους χώρους υποψήφιων λύσεων. Ωστόσο, δεν υπάρχουν ποτέ μεθευρετικές όπως το η PSO που εγγυώνται την εύρεση της βέλτιστης λύσης. Επίσης, η PSO δεν χρησιμοποιεί τη διαβάθμιση του προβλήματος που πρέπει να βελτιστοποιηθεί, πράγμα που σημαίνει ότι δεν απαιτεί το πρόβλημα βελτιστοποίησης να είναι διαφοροποιήσιμο όπως απαιτείται από τις κλασσικές μεθόδους βελτιστοποίησης.

Μια βασική παραλλαγή του αλγόριθμου PSO δουλεύει έχοντας έναν πληθυσμό (σμήνος) υποψήφιων λύσεων (σωματίδια). Αυτά τα σωματίδια μετακινούνται στο χώρο αναζήτησης σύμφωνα με λίγους απλούς κανόνες. Οι κινήσεις των σωματιδίων καθοδηγούνται από τη δική τους πιο γνωστή θέση στο χώρο αναζήτησης καθώς και από την πιο γνωστή θέση ολόκληρου του σμήνους. Όταν ανακαλύπτονται βελτιωμένες θέσεις αυτές θα έρθουν έπειτα για να καθοδηγήσουν τις κινήσεις του σμήνους. Η διαδικασία επαναλαμβάνεται και με αυτόν τον τρόπο αναμένεται, αλλά δεν είναι εγγυημένο ότι θα βρεθεί τελικά μια ικανοποιητική λύση (Shi et al. 2001). Στην εικόνα **4.6.5** που ακολουθεί αναπαριστάται σχηματικά η συμπεριφορά των σωματιδίων με την πάροδο των επαναλήψεων (Πηγή εικόνας: `https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html`).



Σχήμα **4.6.5** Βελτιστοποίηση σμήνους σωματιδίων

Τυπικά, ας είναι $f : \mathbb{R}^n \to \mathbb{R}$ η συνάρτηση κόστους που πρέπει να ελαχιστοποιηθεί. Η συνάρτηση παίρνει μια υποψήφια λύση σαν όρισμα με τη μορφή ενός διανύσματος πραγματικών αριθμών και παράγει έναν πραγματικό αριθμό ως έξοδο που υποδεικνύει την αντικειμενική τιμή συνάρτησης της δεδομένης υποψήφιας λύσης. Η κλίση της f δεν είναι γνωστή. Ο στόχος είναι να βρούμε μια λύση a για την οποία $f(a) \leq f(b)$ για όλα τα b στον χώρο αναζήτησης, που θα σημαίνει a είναι το ολικό ελάχιστο (Eberhart and Kennedy 1995).

Στον αλγόριθμο 12 που ακολουθεί το S είναι ο αριθμός των σωματιδίων, κάθε ένα από τα οποία έχει μια συγκεκριμένη θέση και μια συγκεκριμένη ταχύτητα που δίνονται ώς τα διανύσματα x_i και v_i αντίστοιχα. Το p_i είναι η καλύτερη θέση στην οποία έχει βρεθεί το σωματίδιο i ενώ το g είναι η καλύτερη θέση που έχει ανακαλυφθεί από όλο το σμήνος. Οι τιμές b_{lo} και b_{up} αναπαριστούν το χαμηλότερο και το υψηλότερο όριο στον χώρο αναζήτησης. Το κριτήριο τερματισμού μπορεί να είναι ο αριθμός των επαναλήψεων ή μια λύση όπου πιθανώς να ικανοποιούνται τα κριτήρια αποδεκτού σφάλματος της αντικειμενικής συνάρτησης. Οι παράμετροι ω, φ_p και φ_g επιλέγονται από τον ερευνητή και είναι καθοριστικές για τη συμπεριφορά και την αποτελεσματικότητα της μεθόδου.

Algorithm 12 Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων

Require: *Stopping Criteria*

procedure PARTICLE SWARM OPTIMIZATION(*Stopping Criteria*)
  for each particle i = 1, ..., S **do**
    Initialize the particle's position with a uniformly distributed random vector: $x_i \sim \mathrm{U}(\mathrm{b}_{lo}, b_{up})$
    Initialize the particle's best known position to its initial position
    $p_i \leftarrow x_i$
    if $f(p_i) < f(g)$ **then**
      Update the swarm's best known position:
      $g \leftarrow p_i$
      Initialize the particle's velocity: $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$
    end if
    while Stopping Criteria not met **do**
      for each particle i = 1, ..., S **do**
        for each dimension d = 1, ..., n **do**
          Pick random numbers: r_p, $r_g \sim U(0, 1)$
          Update the particle's velocity:
          $v_{i,d} \leftarrow \omega * v_{i,d} + \varphi_p r_p(p_{i,d} - x_{i,d}) + \varphi_g r_g(g_d - x_{i,d})$
        end for
        Update the particle's position:
        $x_i \leftarrow x_i + v_i$
        if $f(x_i) < f(p_i)$ **then**
          Update the particle's best known position:
          $p_i \leftarrow x_i$
          if $f(p_i) < f(g)$ **then**
            Update the swarm's best known position:
            $g \leftarrow p_i$
          end if
        end if
      end for
    end while

ΚΕΦΑΛΑΙΟ 5

Αναζήτηση μεταβαλλόμενης γειτνίασης

5.1 Εισαγωγή

Η αναζήτηση μεταβαλλόμενης γειτνίασης (VNS) που προτάθηκε από τους (Mladenović and Hansen 1997, Hansen and Mladenović 2003) είναι μια μεθερευρετική μέθοδος για την επίλυση πληθώρας προβλημάτων συνδυαστικής βελτιστοποίησης και εύρεσης του ολικού βέλτιστου. Εξετάζει τις μακρινές περιοχές (ή αλλίως γειτονιές) της τρέχουσας ισχύουσας λύσης και μετακινείται από εκεί σε μία καινούργια λύση εάν και μόνο εάν παρουσιαστεί βελτίωση της αντικειμενικής συνάρτησης. Η τοπικές μέθοδοι αναζήτησης εφαρμόζονται επανειλημμένα για να βρεθούν λύσεις στη γειτονιά πλησιάζοντας το τοπικό βέλτιστο. Η VNS σχεδιάστηκε για την προσέγγιση λύσεων διακριτών και συνεχών προβλημάτων βελτιστοποίησης και σύμφωνα με αυτά, στοχεύει στην επίλυση γραμμικών προβλημάτων, προβλημάτων ακεραίου προγράμματισμού, προβλημάτων μεικτών ακέραιων αριθμών, μη γραμμικών προβλημάτων κλπ. Η VNS αλλάζει συστηματικά τη γειτονιά σε δύο φάσεις. Πρώτον εκτελείται κάθοδος για να βρεθεί μια τοπική βέλτιστη και τελικά μια φάση διαδοχικών διαταραχής για να ξεφύγει από την αντίστοιχη κοιλάδα και να καταφέρει να εξερευνήσει πιο απομακρυσμένες περιοχές. Το VNS βασίζεται στις ακόλουθες αντιλήψεις (Hansen and Mladenović 1999):

- Το τοπικό ελάχιστο σε σχέση με μια δομή γειτονιάς δεν είναι αναγκαστικά τοπικό ελάχιστο για μια άλλη δομή γειτονιάς.
- Ένα παγκόσμιο ελάχιστο είναι ένα τοπικό ελάχιστο σε σχέση με όλες τις πιθανές δομές γειτονιάς.
- Για πολλά προβλήματα, τα τοπικά ελάχιστα σε σχέση με μία ή περισσότερες γειτονιές είναι σχετικά κοντά μεταξύ τους.

Η τοπική αναζήτηση όπως είδαμε και σε προηγούμενο κεφάλαιο χρησιμοποιεί ευρετικούς αλγορίθμους όπως ο 2-opt. Ο στόχος είναι να βρεί μια βελτίωση της υπάρχουσας λύσης μέσα από ένα σύνολο κανόνων που ορίζουν την εκάστοτε γειτονιά. Έτσι προκύπτει το σύνολο επιτρεπτών

βημάτων από τα οποία επιλέγουμε την λύση που παρουσιάζει την μεγαλύτερη βελτίωση σε σχέση με την υπάρχουσα. Εάν δεν βρεθεί βελτίωση αυτό σημαίνει ότι ο αλγόριθμος έχει εγκλωβιστεί σε ένα τοπικό βέλτιστο (στην περίπτωση της δρομολόγησης οχημάτων πρόκειται για τοπικό ελάχιστο) και έτσι δεν μπορεί να κάνει κανένα βήμα. Σκοπός της μεθόδου VNS είναι ο απεγκλωβισμός από τέτοια τοπικά βέλτιστα (Kytöjoki, Nuortio, Bräysy, and Gendreau 2007).

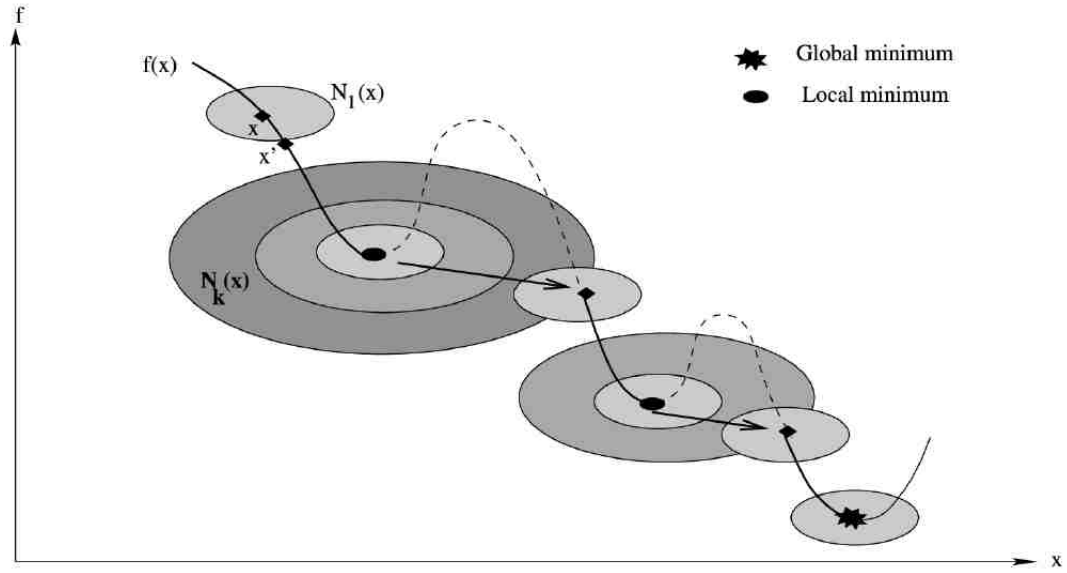
Σε αντίθεση με πολλές άλλες μεθερευρετικές, οι βασικές εκδοχές της VNS και οι επεκτάσεις της είναι απλές και απαιτούν λίγες ή και καθόλου παραμέτρους. Ως εκ τούτου, εκτός από την παροχή πολύ καλών λύσεων, συχνά με απλούστερους τρόπους από άλλες μεθόδους, η VNS μας δείχνει που οφείλεται η πολύ καλή της απόδοση, η οποία, με τη σειρά της, μπορεί να οδηγήσει σε αποδοτικότερες και πιο εξελιγμένες υλοποιήσεις. Τα πεδία εφαρμογών της VNS αυξάνονται ραγδαία σε αριθμό και αφορούν πολλούς τομείς όπως ακέραιος προγραμματισμός, δρομολόγηση οχημάτων, σχεδιασμός δικτύου, χωροθέτηση, τεχνητή νοημοσύνη, μηχανική, προβλήματα συγκέντρωσης, βιολογία, σχεδιασμός τηλεπικοινωνιών.

5.2 Basic Variable Neighborhood Search - BVNS

Η βασική εκδοχή της αναζήτησης μεταβαλλόμενης γειτνίασης BVNS λαμβάνει ώς είσοδο μια αρχική λύση x τον αριθμό διαθέσιμων γειτονιών k_{max} καθώς επίσης και το κριτήριο τερματισμού που συνήθως σε αυτό το είδος αλγορίθμων είναι ένα χρονικό όριο που επιστρέπουμε στον αλγόριθμο έτσι ώστε να μας επιστρέψει την καλύτερη βελτίωση που έχει βρεθέι μέσα σε αυτό το χρονικό περιθώριο. Το k παίρνει αρχικά την τιμή 1 που σημαίνει ότι ξεκινάμε από την πρώτη γειτονιά που έχουμε ορίσει. Στη συνέχεια λαμβάνει χώρα η ανατάραξη (shaking) κατα την οποία γίνονται κάποια τυχαία, όμως σε κάθε περίπτωση επιτρεπτά βήματα, έτσι ώστε να ξεφύγουμε από το τοπικό βέλτιστο και να εξερευνήσουμε και άλλες περιοχές λύσεων. Εκεί ο αλγόριθμος επιχειρεί να εφαρμόσει την μέθοδο καλύτερης βελτίωσης με βάση την γειτονιά k στην οποία βρισκόμαστε. Εάν δεν βρεθεί βελτίωση το k αυξάνεται κατά 1 δηλαδή πηγαίνουμε στην επόμενη διαθέσιμη γειτονιά ξεκινώντας από την αρχή κάνοντας διατάραξη και βελτιστοποίηση με βάση αυτή τη γειτονιά. Εάν βρεθεί βελτίωση τότε το k ξαναγίνεται 1 δηλαδή επιστρέφουμε στην πρώτη γειτονιά και ξεκινάμε από την αρχή. Όλα αυτά εκτελούνται μέχρι να φτάσουμε το χρονικό όριο που έχει τεθεί από την αρχή και επιστρέφεται η καλύτερη λύση που έχει βρεθεί μέχρι εκείνο το σημείο. Η BVNS περιγράφεται στον αλγόριθμο 13 και μία οπτική αναπράσταση της λειτουργίας της δίνεται στην εικόνα **5.2.1**. (Πηγή εικόνας: `https://commons.wikimedia.org/wiki/File:Basic_Variable_Neighborhood_Search.png`)

Algorithm 13 Basic VNS

Require: $Tour\ x, k_{max}, Stopping\ Criteria$
procedure BVNS($Tour\ x, k_{max}$)
  repeat
    $k \leftarrow 1$
    repeat
      $x' \leftarrow Shake(x, k)$
      $x'' \leftarrow Best\ Improvement(x, N_k(x))$
      if $f(x'') < f(x)$ **then**
        $x \leftarrow x''$
        $k \leftarrow 1$
      else
        $k \leftarrow k + 1$
      end if
    until $k = k_{max}$
  until Stopping Criteria
  Return x
end procedure=0



Σχήμα **5.2.1** Σχηματική αναπαράσταση λειτουργίας της BVNS

5.3 Variable Neighborhood Descent - VND

Η κάθοδος μεταβαλλόμενης γειτνίασης (VND) αποτελεί μια μέθοδο βελτιστοποίησης που συνδυάζει αιτιοκρατικά πολλές γειτονίες με σκοπό την εύρεση καλύτερων λύσεων στο πρόβλημά μας. Όπως και η BVNS ξεκινά εξετάζοντας βελτιστοποίηση στην πρώτη γειτονιά και αν δεν βρεθεί κάποια βελτίωση τότε πηγαίνει στην επόμενη. Εάν βρεί βελτίωση σε κάποια γειτονιά τότε κάνει το βήμα που απαιτείται για την βελτίωση και επιστρέφει στην πρώτη γειτονιά. Η διαφορά με την BVNS είναι πως δεν παρεμβάλλεται η διαδικασία ανατάραξης και έτσι η VND ως πλήρως αιτιοκρατική με-

θοδολογία σταματά όταν ικανοποιηθεί το κριτήριο τερματισμού που σε αυτή την περίπτωση δεν είναι ο χρόνος εκτέλεσης αλλά η μη εύρεση κάποιας βελτίωσης (Hansen, Mladenović, Brimberg, and Pérez 2019). Η VND περιγράφεται στον αλγόριθμο 14.

Algorithm 14 VND

Require: *Tour* x, k_{max}, *Stopping Criteria*
1: **procedure** VND(*Tour* x, k_{max})
2: **repeat**
3: $k \leftarrow 1$
4: **repeat**
5: $x' \leftarrow Best\ Improvement(x, N_k(x))$
6: **if** $f(x') < f(x)$ **then**
7: $x \leftarrow x'$
8: $k \leftarrow 1$
9: **else**
10: $k \leftarrow k + 1$
11: **end if**
12: **until** $k = k_{max}$
13: **until** No improvement was found
14: Return x
15: **end procedure**

5.4 General Variable Neighborhood Search - GVNS

Η γενική αναζήτηση μεταβαλλόμενης γειτνίασης (GVNS) είναι η μέθοδος που προκύπτει αν στην BVNS ανικαταστήσουμε το βήμα βελτιστοποίσης Best Improvement με τον αλγόριθμο της καθόδου μεταβαλλόμενης γειτνίασης (VND). Έτσι θα έχουμε πιο αποτελεσματική αλλά ίσως και πιο αργή βελτιστοποίηση. Σε κάθε επανάληψη η τρέχουσα υποψήφια λύση βελτιστοποιείται όχι με τη χρήση μιας γειτονιάς αλλά με την VND η οποία κάνει πιο διεξοδική βελτιστοποίηση εξετάζοντας έναν ικανοποιητικό αριθμό γειτονιών πριν σταματήσει. Στον αλγόριθμο 15 περιγράφεται η μέθοδος της GVNS (Hansen and Mladenović 2001).

Algorithm 15 General VNS

Require: *Tour* x, k_{max}, *Stopping Criteria*
1: **procedure** GVNS(*Tour* x, k_{max})
2: **repeat**
3: $x' \leftarrow Shake(x, k)$
4: $x'' \leftarrow VND(x, k_{max})$
5: **if** $f(x'') < f(x)$ **then**
6: $x \leftarrow x''$
7: $k \leftarrow 1$
8: **else**
9: $k \leftarrow k + 1$
10: **end if**
11: **until** Stopping Criteria
12: Return x
13: **end procedure**

5.5 Reduced Variable Neighborhood Search - RVNS

Η μειωμένη αναζήτηση μεταβαλλόμενης γειτνίασης είναι μια παραλλαγή που είναι πιο δημοφιλής όταν έχουμε να αντιμετωπίσουμε πολύ μεγάλης διάστασης προβλήματα (Hansen and Mladenović 2002). Σε τέτοιου μεγέθους προβλήματα η φάση της βελτιστοποίησης είναι βέβαιο ότι θα μας κοστίσει πολύ σε υπολογιστικό χρόνο και δεν θα καταφέρουμε να εξευρευνήσουμε έναν ικανοποιητικό αριθμό περιοχών του συνολικού χώρου των εφικτών λύσεων, λόγω των περιορισμών χρόνου που θέτουμε στις διάφορες παραλλαγές των αλγορίθμων τύπου VNS. Για αυτό το λόγο έχουμε την RVNS από την οποία λείπει η φάση της βελτιστοποίησης. Ξεκινά με μια αρχική λύση είτε τυχαία είτε με τη χρήση κάποιου ευρετικού αλγορίθμου κατασκευής. Στη συνέχεια όπως και στην BVNS γίνεται το shake αλλά απουσιάζει η βελτιστοποίηση με αποτέλεσμα να γίνονται αλλεπάλληλες τυχαίες μετακινήσεις. Σε κάθε επανάληψη του shake γίνεται ο απαραίτητοε έλεγχος για το αν βρέθηκε κάποια βελτίωση. Η διαδικασία περιγράφεται στον αλγόριθμο 16.

Algorithm 16 Reduced VNS

Require: *Tour* x, k_{max}, *Stopping Criteria*
1: **procedure** RVNS(*Tour* x, k_{max})
2: **repeat**
3: $k \leftarrow 1$
4: **repeat**
5: $x' \leftarrow Shake(x, k)$
6: **if** $f(x') < f(x)$ **then**
7: $x \leftarrow x'$
8: $k \leftarrow 1$
9: **else**
10: $k \leftarrow k + 1$
11: **end if**
12: **until** $k = k_{max}$
13: **until** Stopping Criteria
14: Return x
15: **end procedure**

5.6 Skewed Variable Neighborhood Search - RVNS

Στόχος της συγκεκριμένης παραλλαγής είναι η εξερεύνηση πολύ μακρινών περιοχών μέσα στο χώρο των εφικτών λύσεων. Αυτό επιτυγχάνεται μετρώντας την απόσταση δύο υποψήφιων λύσεων με μια συνάρτηση μετρησης $r(x, x')$ και με τη βοήθεια της παραμέτρου a που με την κατάλληλη επιλογή τιμής θα επιτρέψει την μεταπήδηση σε άλλες πιο απομακρυσμένες περιοχές. Ειδικά στα προβλήματα δρομολόγησης οχημάτων στα οποία ξεκινάμε την εξερεύνηση έχοντας μια ποιοτική αρχική λύση και όχι κάποια τυχαία, η συγκεκριμένη μέθοδος ίσως να μην αποδώσει ιδιαίτερα καλά αποτελέσματα αφού η βέλτιστη λύση του προβλήματος δεν θα βρίσκεται σε κάποια πολύ απομακρυσμένη περιοχή αλλά θα έχει αρκετά κοινά χαρακτηριστικά με την αρχική μας λύση.

Algorithm 17 Skewed VNS

Require: $Tour\ x, k_{max}, Stopping\ Criteria$

procedure SVNS($Tour\ x, k_{max}$)
  $best \leftarrow x$
  repeat
    $k \leftarrow 1$
    repeat
      $x' \leftarrow Shake(x, k)$
      $x'' \leftarrow Best\ Improvement(x', N_k(x'))$
      if $f(x'') - a * r(x, x'') < f(x)$ **then**
        $x \leftarrow x''$
        $k \leftarrow 1$
      else
        $k \leftarrow k + 1$
      end if
    until $k = k_{max}$
    $best \leftarrow Keep\ Best(best, x)$
    $x \leftarrow best$
  until Stopping Criteria
  Return x
end procedure

ΚΕΦΑΛΑΙΟ 6

Μορφή αρχείων CVRP και υλοποίηση αλγόριθμου βασισμένου στην GVNS

6.1 Μορφή αρχείων CVRP

Όλες οι καταχωρήσεις σε αυτή την ενότητα είναι της μορφής <keyword>: <value>, όπου <keyword> υποδηλώνει αλφαριθμητικά και η λέξη-κλειδί <value> δηλώνει αλφαριθμητικά ή αριθμητικά δεδομένα. Οι όροι <string>, <Integer> και <real> υποδηλώνει μορφή σειράς χαρακτήρων, ακέραια ή πραγματικά δεδομένα, αντίστοιχα. Η σειρά των προδιαγραφών των λέξεων κλειδιά στο αρχείο δεδομένων είναι αυθαίρετη, αλλά πρέπει να είναι συνεπής, δηλαδή, κάθε φορά που έχει οριστεί μια λέξη κλειδί, όλες οι απαραίτητες πληροφορίες για τη σωστή ερμηνεία των λέξεων κλειδιά πρέπει να είναι γνωστές. Παρακάτω δίνεται μια λίστα με όλες τις διαθέσιμες λέξεις κλειδιά του προβλήματος CVRP.

- NAME: <string> Προσδιορίζει το αρχείο δεδομένων.
- TYPE: <string> Καθορίζει τον τύπο των δεδομένων.
- COMMENT: <string> Πρόσθετα σχόλια
- DIMENTION: <Integer> Ο συνολικός αριθμός των κόμβων και των αποθηκών.
- CAPACITY: <Integer> Καθορίζει την χωρητικότητα των οχημάτων.
- EDGE WEIGHT TYPE: <string> Καθορίζει τη συνάρτηση υπολογισμού της απόστασης.
- EDGE WEIGHT FORMAT: <string> Περιγράφει τη μορφή των βαρών των ακμών, εάν δοθούν ρητά.
- EDGE DATA FORMAT: <string> Περιγράφει τη μορφή των ακμών ενός γραφήματος, εάν η γραφική παράσταση δίνεται ρητά.

- NODE COORD TYPE: <string> Καθορίζει τις συντεταγμένες του κάθε κόμου και μπορούν να χρησιμοποιηθούν είτε για γραφική απεικόνιση είτε για υπολογισμούς αποστάσεων.
- DISPLAY DATA TYPE: <string> Καθορίζει ρητώς πώς μπορεί να ληφθεί μία γραφική απεικόνιση των κόμβων.
- DEMAND SECTION: <Integer> Καθορίζει την ζήτησης του κάθε κόμβου.
- DEPOT SECTION: <Integer> Καθορίζει ποιοι κόμβοι είναι αποθήκες άρα και αφετηρίες.
- EOF: Τερματίζει τα δεδομένα εισόδου.

Ακολουθεί ένα παράδειγμα από το σύνολο των στιγμιοτύπων που εξετάστηκαν:

NAME : E-n22-k4
COMMENT : (Christophides and Eilon, Min no of trucks: 4, Optimal value: 375)
TYPE : CVRP
DIMENSION : 22
EDGE WEIGHT TYPE : EUC 2D
CAPACITY : 6000
NODE COORD SECTION
1 145 215
2 151 264
3 159 261
4 130 254
5 128 252
6 163 247
7 146 246
8 161 242
9 142 239
10 163 236
11 148 232
12 128 231
13 156 217
14 129 214
15 146 208
16 164 208
17 141 206
18 147 193
19 164 193

20 129 189
21 155 185
22 139 182
DEMAND SECTION
1 0
2 1100
3 700
4 800
5 1400
6 2100
7 400
8 800
9 100
10 500
11 600
12 1200
13 1300
14 1300
15 300
16 900
17 2100
18 1000
19 900
20 2500
21 1800
22 700
DEPOT SECTION
1
-1
EOF

Πρόκειται για το συγμιότυπο E-n22-k4 από το σύνολο των Christophides and Eilon. Το συγκεκριμένο είναι τύπου CVRP με συνολικό αριθμό κόμβων 22. Οι αποστάσεις των κόμβων υπολογίζονται με την συνηθισμένη ευκλείδεια απόσταση και η χωρητικότητα κάθε οχήματος είναι 6000. Στη συνέχεια στο NODE COORD SECTION δίνονται οι συντεταγμένες κάθε κόμβου ενώ στο DEMAND SECTION δίνονται οι απαιτήσεις των κόμβων. Τέλος στο DEPOT SECTION

προσδιορίζεται ποιοί κόμβοι είναι αποθήκες. Σε αυτής της μορφής τα προβλήματα η αποθήκη είναι μια (ο κόμβος 1) ενώ σε άλλου είδους παρόμοια προβλήματα που έχουν πιο πολλές αποθήκες καταγράφονται σε αυτό το σημείο. Το -1 και το EOF σηματοδοτούν το τέλος των δεδομένων του αρχείου.

6.2 Υλοποίηση αλγόριθμου βασισμένου στην GVNS

Η προτεινόμενη υλοποίηση που πραγματοποιήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας περιλαμβάνει αρχικά τις κατάλληλες συναρτήσεις για να διαβάσει τα στιγμιότυπα και να μετατρέψει σε πίνακες τα δεδομένα που δίνονται στα αρχεία μορφής .vrp. Οι συναρτήσεις αυτές διαβάζουν τις συντεταγμένες των σημείων και κατασκευάζουν τον πίνακα αποστάσεων. Η συνάρτηση απόστασης είναι η συνηθισμένη ευκλείδεια απόσταση. Ο πίνακας αποστάσεων που παράγεται είναι τετράγωνος και συμμετρικός διάστασης n*n όπου το στοιχείο που βρίσκεται στην γραμμή i και στη στήλη j δείχνει την απόσταση των κόμβων i και j. Επίσης διαβάζεται και μετατρέπεται σε πίνακα το τμήμα που δηλώνει τη ζήτηση του κάθε κόμβου. Επιπλεόν διαβάζονται σαν απλές μεταβλητές η χωρητικότητα των οχημάτων, η διάσταση του προβλήματος κάθως και η βέλτιστη τιμή της αντικειμένικης συνάρτησης κόστους. Η διάσταση του προβλήματος, η χωρητικότητα των οχημάτων, η ζήτηση κάθε κόμβου, ο πίνακας αποστάσεων καθώς και οι βέλτιστη τιμή εμφανίζονται στην οθόνη μόλις επιλεγεί από το χρήστη το στιγμιότυπο.

Στο επόμενο βήμα της υλοποίησης αφού επιλέξει το στιγμιότυπο το οποίο θέλουμε να εξετάσει, ο χρήστης καλείται να επιλέξει και μία μέθοδο κατασκευής αρχικής λύσης. Οι μέθοδοι που υλοποιήθηκαν είναι ο αλγόριθμος του εγγύτερου γείτονα, ο αλγόριθμος σάρωσης και ο αλγόριθμος εξοικονόμησης. Η λύση που υπολογίζεται από την μέθοδο κατασκευής εμφανίζεται στην οθόνη και μαζί με την συνολική τιμή της αντικειμενικής συνάρτησης κόστους εμφανίζεται και το κόστος, αλλά και η συνολική ζήτηση κάθε διαδρομής ξεχωριστά.

Σε αυτό το σημείο ο χρήστης καλείται να επιλέξει το χρόνο για τον οποίο θα εκτελεστεί η μέθοδος GVNS που υλοποιήθηκε και στη συνέχεια ξεκινά η βελτιστοποίηση. Όταν τελειώσει το χρονικό περιθώριο που έδωσε ο χρήστης τότε ο αλγόριθμος σταματάει και τυπώνει στην οθόνη το καλύτερο αποτέλεσμα που βρέθηκε μέχρι εκείνο το χρονικό σημείο.

Η υλοποίηση της GVNS περιλαμβάνει, όπως περιγράφεται στον αλγόριθμο 15, ένα βήμα ανατάραξης shake και ένα βημα βελτιστοποίησης με τη μέθοδο κατάβασης (VND). Η VND υλοποιήθηκε έτσι ώστε να χρησιμοποιεί τις μεθόδους 2opt, relocate και swap που περιγράφονανται στις υποενότητες 3.2, 3.4 και 3.5 αντίστοιχα. Η επιλογή των μεθόδων αυτών έγινε μετά από πειραματισμούς και παρατηρήσεις όσον αφορά την ποιότητα των αποτελεσμάτων και χρησιμοποιούνται στην VND με τη σειρά που αναφέρθηκαν. Οι γειτονιές swap και relocate έχουν προγραμματιστεί ώστε να κάνουν ανταλλαγές και επανατοποθετήσεις μόνο μεταξύ διαφορετικών διαδρομών και όχι μεσα στην

ίδια διαδρομή, ενώ για την βελτιστοποίηση των διαδρομών ανεξάρτητα χρησιμοποιείται η 2opt.

Η διαδικασία του shaking είναι απαραίτητη όπως περιγράφεται παραπάνω έτσι ώστε να εξευρενηθεί σε ικανοποιητικό βαθμό η περιοχή των εφικτών λύσεων. Η υλοποίηση της μεθόδου που κάνει την ανατάραξη περιλαμβάνει έναν μετρητή ο οποίος ξεκινά από την τιμή 1. Όταν καλείται η συνάρτηση τότε πραγματοποιούνται τόσα τυχαία, αλλά παρόλα αυτά εφικτά βήματα από τις γειτονιές που προαναφέρθηκαν, όσα λέει ο μετρητής. Έπειτα η VND επιχειρεί να κάνει βελτιστοποίηση και ελέγχεται αν προέκυψε κάποια βελτίωση. Εάν δεν προκύψει βελτίωση τότε ο μετρητής ανατάραξης αυξάνεται κατά 1. Αυτό σημαίνει ότι όταν θα κληθεί ξανά η συνάρτηση shake θα γίνουν 2 τυχαία βήματα. Εάν όμως βρεθεί βελτίωση τότε ο μετρητής επανέρχεται στην τιμή 1.

Ο λόγος που χρησιμοποιείται η συγκεκριμένη τεχνική είναι ότι επειδή ξεκινάμε με μια ικανοποιητική λύση που προήλθε από καποιον αλγόριθμο κατασκευής. Γνωρίζουμε ότι λόγω της φύσης του προβλήματος η βέλτιστη δεν θα απέχει πολύ από την λύση που μας δίνει ο αλγόριθμος κατασκευής. Αυτό σημαίνει ότι οι δύο αυτές λύσεις θα έχουν πολλά κοινά χαρακτηριστικά. Έτσι η διαδικασία ανατάραξης πρέπει να γίνει σταδιακά έτσι ώστε να βεβαιωθούμε ότι θα εξερευνήσουμε γειτονικές περιοχές αλλά δεν θα ξεφύγουμε υπερβολικά από αυτές ψάχνοντας σε πολύ μακρινές περιοχές.

ΚΕΦΑΛΑΙΟ 7

Υπολογιστικά αποτελέσματα

Στο πλαίσιο της διπλωματικής υλοποιήθηκαν οι αλγόριθμοι κατασκευής εγγύτερου γείτονα (nearest neighbor), σάρωσης (sweep), καθώς επίσης και ο αλγόριθμος εξοικονόμησης (savings algorithm) των Clarke και Wright. Επίσης υλοποιήθηκαν οι μέθοδοι βελτιστοποίησης για τοπική αναζήτηση 2-opt, 3-opt, ανταλλαγής (Swap), επανατοποθέτησης (Relocate). Όλες οι μέθοδοι σχεδιάστηκαν τόσο για την καλύτερη δυνατή βελτίωση, όσο και για την πρώτη βελτίωση. Τέλος υλοποιήθηκε μια εκδοχή της GVNS η οποία χρησιμοποιεί τις προαναφερθείσες γειτονίες για τοπική αναζήτηση. Για την υλοποίηση χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python3. Τα πειράματα εκτελέστηκαν σε ένα laptop ACER Aspire V15 με τα ακόλουθα τεχνικά χαρακτηριστικά:

- Επεξεργαστής: Intel i5-6300HQ 2.3GHz with Turbo Boost up to 3.3GHz
- Μνήμη RAM: 8GB DDR4
- Λειτουργικό σύστημα: Windows 10 Education 64-bit

Τα πειράματα εκτελέστηκαν για μια πληθώρα στιγμιοτύπων από την TSP-Lib. Αυτά περιλαμβάνουν τα σετ A,B,P των Augerat, et al. καθώς και το σύνολο των Christofides & Eilon. Στις ενότητες 7.1, 7.2, 7.3 παρουσιάζονται τα αποτελέσματα για όλα τα στιγμιότυπα των συνόλων αυτών. Στους πίνακες **7.1**, **7.6**, **7.11** που ακολουθούν παρουσιάζονται κάποια στατιστικά στοιχεία για την μέθοδους κατασκευής του εγγύτερου γείτονα, σάρωσης και εξοικονόμησης αντίστοιχα, υπολογισμένα για κάθε σύνολο στιγμιοτύπων ξεχωριστά. Αυτά είναι η μέση τιμή του σφάλματος, η τυπική απόκλιση, η μέγιστη και η ελάχιστη τιμή σφάλματος.

Στούς πίνακες **7.2**, **7.3**, **7.4**, **7.5** παρατίθενται τα αποτελέσματα της μεθόδου κατασκευής του εγγύτερου γείτονα. Στούς πίνακες **7.7**, **7.8**, **7.9**, **7.10** παρατίθενται τα αποτελέσματα της μεθόδου σάρωσης. Στούς πίνακες **7.12**, **7.13**, **7.14**, **7.15** παρατίθενται τα αποτελέσματα της μεθόδου εξοικονόμησης.

Στην πρώτη στήλη δίνεται το όνομα του στιγμιοτύπου. Στην στήλη Customers δίνεται ο αριθμός των πελατών προς εξυπηρέτηση. Στις στήλες Vehicles και Optimal δίνονται ο αριθμός των

οχημάτων που απαιτούνται για την βέλτιστη δρομολόγηση και η βέλτιστη τιμή της αντικειμενικής συνάρτησης αντίστοιχα. Στις στήλες Vehicles και Value δίνονται ο αριθμός των οχημάτων της λύσης που δίνει ο αλγόριθμος κατασκευής και η τιμή της αντικειμενικής συνάρτησης αντίστοιχα. Στην στήλη Error δίνεται το σχετικό σφάλμα της λύσης. Στην τελευταία στήλη Time δίνεται ο χρόνος εκτέλεσης του αλγορίθμου σε δευτερόλεπτα.

7.1 Πίνακες υπολογιστικών αποτελεσμάτων για την μέθοδο του κοντινότερου γείτονα.

Πίνακας **7.1**: Στατιστικά στοιχεία της μεθόδου κοντινότερου γείτονα για τα set A,B,P,E

Set	Mean	Standard Deviation	Minimum	Maximum
E	38.45%	14.09%	17%	61%
A	39.11%	8.22%	22%	57%
B	41.69%	19.35%	12%	93%
P	33.34%	11.95%	10%	53%

Πίνακας **7.2**: Αποτελέσματα της μεθόδου κοντινότερου γείτονα για το set των Christofides & Eilon

Πρόβλημα	Customers	Vehicles	Optimal	NN Vehicles	NN Value	Error	Time
E-n22-k4	21	4	375	4	464	24%	0.002
E-n23-k3	22	3	569	3	760	34%	0.002
E-n30-k3	29	3	534	3	627	17%	0.002
E-n33-k4	32	4	835	4	1010	21%	0.002
E-n51-k5	50	5	521	5	728	40%	0.003
E-n76-k7	75	7	682	7	1098	61%	0.006
E-n76-k8	75	8	735	8	1081	47%	0.007
E-n76-k10	75	10	830	10	1221	47%	0.007
E-n76-k14	75	14	1021	15	1353	33%	0.008
E-n101-k8	100	8	815	8	1176	44%	0.010
E-n101-k14	100	14	1067	14	1658	55%	0.011

Πίνακας **7.3**: Αποτελέσματα της μεθόδου κοντινότερου γείτονα για το set A των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	NN Vehicles	NN Value	Error	Time
A-n32-k5	31	5	784	5	1145	46 %	0.002
A-n33-k5	32	5	661	5	977	48 %	0.003
A-n33-k6	32	6	742	6	1042	40 %	0.003
A-n34-k5	33	5	778	5	982	26 %	0.003
A-n36-k5	35	5	799	5	1077	35 %	0.003
A-n37-k5	36	5	699	5	948	42 %	0.003
A-n37-k6	36	6	949	6	1334	41 %	0.003
A-n38-k5	37	5	730	5	1084	48 %	0.002
A-n39-k5	38	5	822	5	1070	30 %	0.003
A-n39-k6	38	6	831	6	1145	38 %	0.003
A-n44-k6	43	6	937	6	1391	48 %	0.003
A-n45-k6	44	6	944	7	1485	57 %	0.003
A-n45-k7	44	7	1146	7	1428	25 %	0.003
A-n46-k7	45	7	914	7	1335	46 %	0.004
A-n48-k7	47	7	1073	7	1475	37 %	0.004
A-n53-k7	52	7	1010	7	1444	43 %	0.004
A-n54-k7	53	7	1167	7	1421	22 %	0.006
A-n55-k9	54	9	1073	9	1500	40 %	0.004
A-n60-k9	59	9	1354	9	1837	36 %	0.005
A-n61-k9	60	9	1034	9	1399	35 %	0.005
A-n62-k8	61	8	1288	8	1775	38 %	0.005
A-n63-k9	62	9	1616	9	2188	35 %	0.005
A-n63-k10	62	10	1314	10	1934	47%	0.006
A-n64-k9	63	9	1401	9	1964	40 %	0.006
A-n65-k9	64	9	1174	9	1749	49 %	0.006
A-n69-k9	68	9	1159	9	1513	31 %	0.006
A-n80-k10	79	10	1763	10	2348	33%	0.007

Πίνακας **7.4**: Αποτελέσματα της μεθόδου κοντινότερου γείτονα για το set B των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	NN Vehicles	NN Value	Error	Time
B-n31-k5	30	5	672	5	887	32%	0.002
B-n34-k5	33	5	788	5	886	12%	0.002
B-n35-k5	34	5	955	5	1304	36%	0.003
B-n38-k6	37	6	805	6	1268	57%	0.003
B-n39-k5	38	5	549	5	1058	93%	0.003
B-n41-k6	40	6	829	6	1080	30%	0.003
B-n43-k6	42	6	742	6	965	30%	0.004
B-n44-k7	43	7	909	7	1274	40%	0.003
B-n45-k5	44	5	751	5	1051	40%	0.003
B-n45-k6	44	6	678	7	1001	48%	0.004
B-n50-k7	49	7	741	7	1036	40%	0.004
B-n50-k8	49	8	1312	8	1527	16%	0.004
B-n51-k7	50	7	1032	7	1333	29%	0.004
B-n52-k7	51	7	747	7	1276	71%	0.005
B-n56-k7	55	7	707	7	1248	76%	0.005
B-n57-k7	56	7	1153	8	1866	62%	0.004
B-n57-k9	56	9	1598	9	1952	22%	0.005
B-n63-k10	62	10	1496	10	2002	34%	0.007
B-n64-k9	63	9	861	9	1272	48%	0.005
B-n66-k9	65	9	1316	9	1679	28%	0.005
B-n67-k10	66	10	1032	10	1427	38%	0.006
B-n68-k9	67	9	1272	9	1698	33%	0.005
B-n78-k10	77	10	1221	10	1758	44%	0.007

Πίνακας **7.5**: Αποτελέσματα της μεθόδου κοντινότερου γείτονα για το set P των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	NN Vehicles	NN Value	Error	Time
P-n16-k8	15	8	450	8	497	10%	0.003
P-n19-k2	18	2	212	2	250	18%	0.001
P-n20-k2	19	2	216	2	309	43%	0.001
P-n21-k2	20	2	211	2	257	22%	0.001
P-n22-k2	21	2	216	2	261	20%	0.001
P-n22-k8	21	8	603	9	801	33%	0.002
P-n23-k8	22	8	529	9	746	41%	0.003
P-n40-k5	39	5	458	5	682	49%	0.002
P-n45-k5	44	5	510	5	755	48%	0.003
P-n50-k7	49	7	554	7	794	43%	0.003
P-n50-k8	49	8	631	9	820	30%	0.005
P-n50-k10	49	10	696	10	907	30%	0.004
P-n51-k10	50	10	741	10	904	22%	0.004
P-n55-k7	54	7	568	7	754	33%	0.005
P-n55-k10	54	10	694	10	983	42%	0.006
P-n55-k15	54	15	989	16	1217	23%	0.007
P-n60-k10	59	10	744	10	1014	36%	0.005
P-n60-k15	59	15	968	15	1170	21%	0.006
P-n65-k10	64	10	792	10	1096	38%	0.007
P-n70-k10	69	10	827	10	1010	22%	0.006
P-n76-k4	75	4	593	4	833	40%	0.005
P-n76-k5	75	5	627	5	962	53%	0.006
P-n101-k4	100	4	681	4	1021	50%	0.008

7.2 Υπολογιστικά αποτελέσματα για την μέθοδο κατασκευής sweep.

Πίνακας **7.6** Στατιστικά στοιχεία της μεθόδου Sweep για τα set A,B,P,E

Set	Mean	Standard Deviation	Minimum	Maximum
E	10.81%	7.78%	4%	32%
A	18.40%	4.75%	8%	26%
B	17.39%	7.42%	5%	26%
P	12.04%	6.63%	5%	32%

Πίνακας **7.7** Αποτελέσματα της μεθόδου Sweep για το set των Christofides & Eilon

Πρόβλημα	Customers	Vehicles	Optimal	Sw Vehicles	Sw Value	Error	Time
E-n22-k4	21	4	375	4	397	6%	0.011
E-n23-k3	22	3	569	4	750	32%	0.007
E-n30-k3	29	3	534	3	554	4%	0.099
E-n33-k4	32	4	835	5	928	11%	0.071
E-n51-k5	50	5	521	6	601	15%	0.228
E-n76-k7	75	7	682	7	725	6%	0.529
E-n76-k8	75	8	735	8	799	9%	0.565
E-n76-k10	75	10	830	11	926	12%	0.236
E-n76-k14	75	14	1021	15	1147	12%	0.080
E-n101-k8	100	8	815	8	860	6%	1.659
E-n101-k14	100	14	1067	15	1234	16%	0.305

Πίνακας **7.8** Αποτελέσματα της μεθόδου sweep για το set A των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	Sw Vehicles	Sw Value	Error	Time
A-n32-k5	31	5	784	5	882	12%	0.063
A-n33-k5	32	5	661	5	788	19%	0.027
A-n33-k6	32	6	742	7	874	18%	0.020
A-n34-k5	33	5	778	6	867	11%	0.041
A-n36-k5	35	5	799	5	949	19%	0.049
A-n37-k5	36	5	699	5	833	25%	0.067
A-n37-k6	36	6	949	7	1131	19%	0.030
A-n38-k5	37	5	730	6	876	20%	0.059
A-n39-k5	38	5	822	6	1009	23%	0.052
A-n39-k6	38	6	831	6	989	19%	0.085
A-n44-k6	43	6	937	7	1164	24%	0.078
A-n45-k6	44	6	944	7	1118	18%	0.039
A-n45-k7	44	7	1146	7	1343	17%	0.076
A-n46-k7	45	7	914	7	1038	14%	0.084
A-n48-k7	47	7	1073	7	1155	8%	0.075
A-n53-k7	52	7	1010	8	1174	16%	0.056
A-n54-k7	53	7	1167	8	1364	17%	0.131
A-n55-k9	54	9	1073	9	1202	12%	0.066
A-n60-k9	59	9	1354	10	1683	24%	0.107
A-n61-k9	60	9	1034	10	1222	18%	0.061
A-n62-k8	61	8	1288	9	1619	26%	0.237
A-n63-k9	62	9	1616	10	1826	13%	0.114
A-n63-k10	62	10	1314	11	1554	18%	0.138
A-n64-k9	63	9	1401	10	1599	14%	0.158
A-n65-k9	64	9	1174	10	1367	16%	0.091
A-n69-k9	68	9	1159	10	1263	9%	0.149
A-n80-k10	79	10	1763	11	2137	21%	0.292

Πίνακας **7.9** Αποτελέσματα της μεθόδου Sweep για το set B των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	Sw Vehicles	Sw Value	Error	Time
B-n31-k5	30	5	672	5	801	19%	0.044
B-n34-k5	33	5	788	6	996	26%	0.023
B-n35-k5	34	5	955	5	1002	5 %	0.025
B-n38-k6	37	6	805	6	915	13%	0.053
B-n39-k5	38	5	549	5	611	11%	0.074
B-n41-k6	40	6	829	6	884	6 %	0.014
B-n43-k6	42	6	742	6	813	10%	0.050
B-n44-k7	43	7	909	8	1210	33%	0.051
B-n45-k5	44	5	751	6	905	20%	0.087
B-n45-k6	44	6	678	7	857	26%	0.057
B-n50-k7	49	7	741	7	785	6 %	0.078
B-n50-k8	49	8	1312	8	1516	15%	0.073
B-n51-k7	50	7	1032	8	1275	24%	0.041
B-n52-k7	51	7	747	7	923	24%	0.056
B-n56-k7	55	7	707	7	833	18%	0.112
B-n57-k7	56	7	1153	8	1446	25%	0.078
B-n57-k9	56	9	1598	9	1858	17%	0.065
B-n63-k10	62	10	1496	11	1730	16%	0.065
B-n64-k9	63	9	861	10	1027	19%	0.073
B-n66-k9	65	9	1316	10	1503	14%	0.112
B-n67-k10	66	10	1032	11	1271	23%	0.113
B-n68-k9	67	9	1272	9	1383	9 %	0.111
B-n78-k10	77	10	1221	11	1483	21%	0.087

Πίνακας **7.10** Αποτελέσματα της μεθόδου Sweep για το set P των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	Sw Vehicles	Sw Value	Error	Time
P-n16-k8	15	8	450	10	595	32%	0.004
P-n19-k2	18	2	212	2	236	11%	0.022
P-n20-k2	19	2	216	2	238	10%	0.028
P-n21-k2	20	2	211	2	238	13%	0.038
P-n22-k2	21	2	216	2	237	10%	0.046
P-n22-k8	21	8	603	10	690	14%	0.006
P-n23-k8	22	8	529	11	687	30%	0.005
P-n40-k5	39	5	458	5	512	12%	0.045
P-n45-k5	44	5	510	5	533	5 %	0.108
P-n50-k7	49	7	554	7	598	8 %	0.091
P-n50-k8	49	8	631	9	693	10%	0.042
P-n50-k10	49	10	696	11	798	15%	0.027
P-n51-k10	50	10	741	11	808	9 %	0.040
P-n55-k7	54	7	568	7	644	13%	0.120
P-n55-k10	54	10	694	10	763	10%	0.062
P-n55-k15	54	15	989	18	1138	15%	0.018
P-n60-k10	59	10	744	11	823	11%	0.105
P-n60-k15	59	15	968	16	1091	13%	0.027
P-n65-k10	64	10	792	11	864	9 %	0.099
P-n70-k10	69	10	827	11	903	9 %	0.209
P-n76-k4	75	4	593	4	631	6 %	2.383
P-n76-k5	75	5	627	5	672	7 %	1.158
P-n101-k4	100	4	681	4	714	5 %	6.944

7.3 Υπολογιστικά αποτελέσματα για την μέθοδο κατασκευής των Clarke και Wright.

Πίνακας **7.11** Στατιστικά στοιχεία του αλγοριθμου εξοικονόμησης για τα set A,B,P,E

Set	Mean	Standard Deviation	Minimum	Maximum
E	18.72%	4.61%	9%	23%
A	14.59%	4.03%	5%	21%
B	13.95%	4.86%	3%	22%
P	13.08%	6.55%	3%	30%

Πίνακας **7.12**: Αποτελέσματα της μεθόδου των Clarke και Wright για το set των Christofides & Eilon

Πρόβλημα	Customers	Vehicles	Optimal	CW Vehicles	CW Value	Error	Time
E-n22-k4	21	4	375	4	448	19%	0.012
E-n23-k3	22	3	569	3	696	22%	0.007
E-n30-k3	29	3	534	3	608	14%	0.034
E-n33-k4	32	4	835	4	912	9%	0.034
E-n51-k5	50	5	521	5	623	20%	0.327
E-n76-k7	75	7	682	7	826	21%	1.997
E-n76-k8	75	8	735	8	905	23%	1.872
E-n76-k10	75	10	830	10	1016	22%	1.937
E-n76-k14	75	14	1021	15	1158	13%	1.903
E-n101-k8	100	8	815	8	997	22%	6.756
E-n101-k14	100	14	1067	14	1296	21%	6.343

Πίνακας **7.13**: Αποτελέσματα της μεθόδου των Clarke και Wright για το set A των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	CW Vehicles	CW Value	Error	Time
A-n32-k5	31	5	784	5	904	15%	0.036
A-n33-k5	32	5	661	5	759	15%	0.054
A-n33-k6	32	6	742	6	813	10%	0.052
A-n34-k5	33	5	778	5	884	14%	0.069
A-n36-k5	35	5	799	5	916	15%	0.065
A-n37-k5	36	5	699	5	777	16%	0.078
A-n37-k6	36	6	949	6	999	5%	0.066
A-n38-k5	37	5	730	5	832	14%	0.082
A-n39-k5	38	5	822	5	907	10%	0.079
A-n39-k6	38	6	831	6	939	13%	0.085
A-n44-k6	43	6	937	6	1101	18%	0.136
A-n45-k6	44	6	944	6	1120	19%	0.168
A-n45-k7	44	7	1146	7	1256	10%	0.160
A-n46-k7	45	7	914	7	1071	17%	0.201
A-n48-k7	47	7	1073	7	1303	21%	0.186
A-n53-k7	52	7	1010	7	1167	16%	0.314
A-n54-k7	53	7	1167	7	1298	11%	0.330
A-n55-k9	54	9	1073	9	1275	19%	0.443
A-n60-k9	59	9	1354	9	1562	15%	0.576
A-n61-k9	60	9	1034	9	1186	15%	0.668
A-n62-k8	61	8	1288	8	1415	10%	0.556
A-n63-k9	62	9	1616	9	1932	20%	0.655
A-n63-k10	62	10	1314	10	1466	12%	0.692
A-n64-k9	63	9	1401	9	1643	17%	0.641
A-n65-k9	64	9	1174	9	1441	23%	0.869
A-n69-k9	68	9	1159	9	1312	13%	1.149
A-n80-k10	79	10	1763	10	1950	11%	1.642

Πίνακας **7.14**: Αποτελέσματα της μεθόδου των Clarke και Wright για το set B των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	CW Vehicles	CW Value	Error	Time
B-n31-k5	30	5	672	5	707	5 %	0.045
B-n34-k5	33	5	788	5	881	11%	0.057
B-n35-k5	34	5	955	5	1121	17%	0.058
B-n38-k6	37	6	805	6	915	13%	0.105
B-n39-k5	38	5	549	5	672	22%	0.103
B-n41-k6	40	6	829	6	851	3 %	0.114
B-n43-k6	42	6	742	6	873	18%	0.149
B-n44-k7	43	7	909	7	1015	12%	0.143
B-n45-k5	44	5	751	5	856	14%	0.207
B-n45-k6	44	6	678	7	740	9 %	0.210
B-n50-k7	49	7	741	7	872	18%	0.357
B-n50-k8	49	8	1312	8	1453	11%	0.257
B-n51-k7	50	7	1032	7	1119	8 %	0.435
B-n52-k7	51	7	747	7	882	18%	0.378
B-n56-k7	55	7	707	7	818	16%	0.458
B-n57-k7	56	7	1153	8	1246	8 %	0.562
B-n57-k9	56	9	1598	9	1772	11%	0.406
B-n63-k10	62	10	1496	10	1661	11%	0.683
B-n64-k9	63	9	861	9	977	13%	1.154
B-n66-k9	65	9	1316	9	1533	16%	0.866
B-n67-k10	66	10	1032	10	1253	21%	1.102
B-n68-k9	67	9	1272	9	1398	10%	0.927
B-n78-k10	77	10	1221	10	1419	16%	1.720

Πίνακας **7.15**: Αποτελέσματα της μεθόδου των Clarke και Wright για το set P των Augerat, et al.

Πρόβλημα	Customers	Vehicles	Optimal	CW Vehicles	CW Value	Error	Time
P-n16-k8	15	8	450	9	482	7 %	0.006
P-n19-k2	18	2	212	2	238	12%	0.005
P-n20-k2	19	2	216	2	234	8 %	0.005
P-n21-k2	20	2	211	2	236	12%	0.006
P-n22-k2	21	2	216	2	240	11%	0.008
P-n22-k8	21	8	603	9	626	4 %	0.015
P-n23-k8	22	8	529	9	541	3 %	0.015
P-n40-k5	39	5	458	5	509	11%	0.112
P-n45-k5	44	5	510	5	622	22%	0.199
P-n50-k7	49	7	554	7	640	16%	0.336
P-n50-k8	49	8	631	9	723	15%	0.342
P-n50-k10	49	10	696	10	739	7 %	0.344
P-n51-k10	50	10	741	10	848	15%	0.356
P-n55-k7	54	7	568	7	665	17%	0.505
P-n55-k10	54	10	694	10	765	10%	0.499
P-n55-k15	54	15	989	16	1033	4 %	0.517
P-n60-k10	59	10	744	10	862	16%	0.733
P-n60-k15	59	15	968	16	1052	9 %	0.734
P-n65-k10	64	10	792	10	916	16%	1.022
P-n70-k10	69	10	827	10	953	15%	1.407
P-n76-k4	75	4	593	4	730	23%	1.827
P-n76-k5	75	5	627	5	812	30%	1.895
P-n101-k4	100	4	681	4	804	18%	6.867

7.4 Υπολογιστικά αποτελέσματα της αναζήτησης μεταβλητής γειτονιάς

Σε αυτή την υποενότητα παρουσιάζονται τα αποτελέσματα της υλοποίησης του αλγορίθμου βασισμένου στην GVNS. Τα προβλήματα χωρίστηκαν σε 6 κατηγορίες ανάλογα με την διάστασή τους δηλαδή ανάλογα με τον αριθμό των πελατών προς εξυπηρέτηση. Οι κατηγορίες αυτές είναι για προβλήματα με λιγότερους από 30 πελάτες, με 30 έως 40 πελάτες, με 40 έως 50 πελάτες, με 50 έως 60 πελάτες, με 60 έως 70 πελάτες και με περισσότερους από 70 πελάτες και τα αποτελέσματα δίνονται στους πίνακες **7.16**, **7.17**, **7.18**, **7.19**, **7.20**, **7.21** αντίστοιχα. Σε όλα τα υπολογιστικά πειράματα που ακολουθούν έχει επιλεγεί σαν αρχική λύση ο αλγόριθμος αξοικονόμησης.

Στην πρώτη στήλη δίνεται το όνομα του στιγμιοτύπου ενώ στην δεύτερη στήλη Optimal value δίνεται η βέλτιστη τιμή της αντικειμενικής συνάρτησης. Ο αλγόριθμος εκτελέστηκε από 10 φορές για κάθε στιγμιότυπο για χρόνο 30 δευτερολέπτων και από 10 φορές για χρόνο 60 δευτερολέπτων. Στις στήλες 30 sec avg και 60 sec avg δίνεται ο μέσος όρος της αντικειμενικής συνάρτησης για τις 10 εκτελέσεις των 30 δευτερολέπτων και τις 10 εκτελέσεις των 60 δευτερολέπτων αντίστοιχα ενώ στις στήλες 30 sec Error και60 sec Error καταγράφεται το αντίστοιχο σχετικό σφάλμα. Για κάθε έναν από τους παρακάτω πίνακες δίνεται το μέσο, το μέγιστο και το ελάχιστο μέσο σφάλμα για τα στιγμιότυπα του πίνακα.

Πίνακας **7.16** Αποτελέσματα της VNS για στιγμιότυπα διάστασης έως 30

Πρόβλημα	Optimal Value	30 sec avg	30 sec Error	60 sec avg	60 sec Error
P-n16-k8	450	450.0	0.0%	-	-
P-n19-k2	212	214.2	1%	213.5	0.7%
P-n20-k2	216	216.4	0.2%	-	-
P-n21-k2	211	211.0	0.0%	-	-
P-n22-k2	216	216.0	0.0%	-	-
P-n22-k8	603	606.6	0.6%	-	-
P-n23-k8	529	532.7	0.7%	-	-
E-n22-k4	375	375.7	0.2%	-	-
E-n23-k3	569	569.2	0.0%	-	-
E-n30-k3	534	539.4	1.0%	537.0	0.6%

- **Μέσος όρος (30 sec)**:0.4%, **Μέγιστο σφάλμα**:1.0%, **Ελάχιστο σφάλμα**:0.0%

Πίνακας **7.17** Αποτελέσματα της VNS για στιγμιότυπα διάστασης από 30 έως 40

Πρόβλημα	Optimal Value	30 sec avg	30 sec Error	60 sec avg	60 sec Error
A-n32-k5	784	809.5	3.2%	807.5	3.0%
A-n33-k5	661	663.4	0.3%	-	- %
A-n33-k6	742	745.3	0.4%	-	- %
A-n34-k5	778	786.3	1.1%	785.0	0.9%
A-n36-k5	799	819.4	2.5%	816.2	2.1%
A-n37-k5	669	688.1	2.9%	675.0	0.8%
A-n37-k6	949	961.4	1.3%	959.9	1.1%
A-n38-k5	730	741.2	1.5%	739.5	1.3%
A-n39-k5	822	839.8	2.1%	834.5	1.5%
A-n39-k6	831	855.9	3.0%	840.9	1.1%
B-n31-k5	672	680.2	1.2%	679.8	1.2%
B-n34-k5	788	789.5	0.2%	-	- %
B-n35-k5	955	957.3	0.2%	-	- %
B-n38-k6	805	809.8	0.6%	-	- %
B-n39-k5	549	553.7	0.8%	-	- %
P-n40-k5	458	462.5	1.0%	461.6	0.8%
E-n33-k4	835	846.3	1.4%	842.0	0.8%

- **Μέσος όρος (30 sec)**:1.4%, **Μέγιστο σφάλμα**:2.9%, **Ελάχιστο σφάλμα**:0.2%

Πίνακας **7.18** Αποτελέσματα της VNS για στιγμιότυπα διάστασης από 40 έως 50

Πρόβλημα	Optimal Value	30 sec avg	30 sec Error	60 sec avg	60 sec Error
A-n44-k6	937	971.5	3.7%	949.0	1.3%
A-n45-k6	944	1018.3	7.8%	1007.6	6.7%
A-n45-k7	1146	1192.4	4.0%	1185.3	3.4%
A-n46-k7	914	947.8	3.7%	939.6	2.8%
A-n48-k7	1073	1094.6	2.0%	1131.5	5.4%
B-n41-k6	829	837.8	1.0%	837.1	1.0%
B-n43-k6	742	760.3	2.5%	755.2	1.8%
B-n44-k7	909	949.8	4.5%	946.4	4.1%
B-n45-k5	751	766.7	2.1%	765.6	1.9%
B-n45-k6	678	718.0	5.9%	712.6	5.1%
B-n50-k7	741	751.1	1.4%	749.1	1.1%
B-n50-k8	1312	1337.4	1.9%	1327.8	1.2%
P-n45-k5	510	515.7	1.1%	515.4	1.0%
P-n50-k7	554	564.9	2.0%	563.4	1.7%
P-n50-k8	631	661.3	4.8%	655.6	3.9%
P-n50-k10	696	720.4	3.5%	714.1	2.6%

- **Μέσος όρος (30 sec)**:3.2%, **Μέγιστο σφάλμα**:7.8%, **Ελάχιστο σφάλμα**:1.0%
- **Μέσος όρος (60 sec)**:2.8%, **Μέγιστο σφάλμα**:6.7%, **Ελάχιστο σφάλμα**:1.0%

Πίνακας **7.19** Αποτελέσματα της VNS για στιγμιότυπα διάστασης από 50 έως 60

Πρόβλημα	Optimal Value	30 sec avg	30 sec Error	60 sec avg	60 sec Error
A-n53-k7	1010	1061.5	5.0%	1052.0	4.1%
A-n54-k7	1167	1217.3	4.3%	1197.2	2.6%
A-n55-k9	1073	1103.7	2.8%	1093.0	1.9%
A-n60-k9	1354	1428.3	5.4%	1407.6	3.9%
B-n51-k7	1032	1054.8	2.2%	1050.6	1.8%
B-n52-k7	747	765.3	2.4%	760.8	1.8%
B-n56-k7	707	730.3	3.3%	726.2	2.7%
B-n57-k7	1153	1254.3	5.4%	1200.3	4.1%
B-n57-k9	1598	1628.4	1.9%	1624.0	1.6%
P-n51-k10	741	790.2	6.6%	781.2	5.4%
P-n55-k7	568	591.9	4.2%	583.2	2.7%
P-n55-k10	694	716.1	3.2%	709.8	2.3%
P-n55-k15	989	1021.6	3.3%	1016.7	2.8%
P-n60-k10	744	781.3	5.0%	771.0	3.6%
P-n60-k15	968	1023.2	5.7%	1015.4	4.9%
E-n51-k5	521	581.8	11.6%	577.4	10.8%

- **Μέσος όρος (30 sec)**:4.5%, **Μέγιστο σφάλμα**:11.6%, **Ελάχιστο σφάλμα**:1.9%
- **Μέσος όρος (60 sec)**:3.6%, **Μέγιστο σφάλμα**:10.8%, **Ελάχιστο σφάλμα**:1.6%

Πίνακας **7.20** Αποτελέσματα της VNS για στιγμιότυπα διάστασης από 60 έως 70

Πρόβλημα	Optimal Value	30 sec avg	30 sec Error	60 sec avg	60 sec Error
A-n61-k9	1034	1092.8	5.7%	1085.8	5.0%
A-n62-k8	1288	1372.9	6.6%	1358.2	5.4%
A-n63-k9	1616	1704.9	5.5%	1696.0	4.9%
A-n63-k10	1314	1371.1	4.3%	1350.6	2.8%
A-n64-k9	1401	1502.2	7.2%	1488.5	6.2%
A-n65-k9	1174	1234.7	5.1%	1224.4	4.3%
A-n69-k9	1159	1210.0	4.4%	1192.4	2.8%
B-n63-k10	1496	1596.6	6.7%	1578.0	5.4%
B-n64-k9	861	887.9	3.1%	882.7	2.5%
B-n66-k9	1316	1383.1	5.1%	1370.2	4.1%
B-n67-k10	1032	1103.6	6.9%	1093.3	5.9%
B-n68-k9	1272	1317.7	3.6%	1310.4	3.0%
P-n65-k10	792	832.9	5.2%	831.8	5.0%
P-n70-k10	827	896.6	8.6%	892.6	7.9%

- **Μέσος όρος (30 sec)**:5.6%, **Μέγιστο σφάλμα**:8.6%, **Ελάχιστο σφάλμα**:3.1%
- **Μέσος όρος (60 sec)**:4.7%, **Μέγιστο σφάλμα**:7.9%, **Ελάχιστο σφάλμα**:2.5%

Πίνακας **7.21** Αποτελέσματα της VNS για στιγμιότυπα διάστασης μεγαλύτερης από 70

Πρόβλημα	Optimal Value	30 sec avg	30 sec Error	60 sec avg	60 sec Error
A-n80-k10	1763	1881.1	6.7%	1870.6	6.1%
B-n78-k10	1221	1299.2	6.4%	1288.2	5.5%
P-n76-k4	593	623.0	5.0%	621.5	4.8%
P-n76-k5	627	652.5	4.0%	651.4	3.9%
P-n101-k4	681	712.4	4.6%	710.2	4.3%
E-n76-k7	682	739.4	8.4%	731.1	7.2%
E-n76-k8	735	786.2	7.0%	778.4	5.9%
E-n76-k10	830	896.6	8.0%	891.4	7.4%
E-n76-k14	1021	1104.5	8.1%	1098.6	7.6%
E-n101-k8	815	852.3	4.5%	845.1	3.7%
E-n101-k14	1067	1150.1	7.8%	1133.4	6.2%

- **Μέσος όρος (30 sec)**:6.4%, **Μέγιστο σφάλμα**:8.4%, **Ελάχιστο σφάλμα**:4.0%
- **Μέσος όρος (60 sec)**:5.7%, **Μέγιστο σφάλμα**:7.2%, **Ελάχιστο σφάλμα**:3.9%

ΚΕΦΑΛΑΙΟ 8

Συμπεράσματα και Μελλοντική Έρευνα

8.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία είχε ως στόχο την αποδοτική υλοποίηση αλγορίθμων κατασκευής αρχικής λύσης και βελτιστοποίησης για το πρόβλημα δρομολόγησης οχημάτων με περιορισμό χωρητικότητας καθώς επίσης και ενός αλγορίθμου βασισμένου στην γενική αναζήτηση μεταβαλλόμενης γειτνίασης.

Στο κεφάλαιο 7 παρουσιάστηκαν τα αποτελέσματα των μεθόδων αρχικοποίησης του εγγύτερου γείτονα, της σάρωσης και της εξοικονόμησης. Είναι εμφανές ότι η μέθοδος του εγγύτερου γείτονα αν και είναι η γρηγορότερη με μεγάλη διαφορά, αδυνατεί να μας δώσει ποιοτικά αποτελέσματα. Οι άλλες δυο μέθοδοι είναι πολύ κοντά όσον αφορά την ποιότητα των αποτελεσμάτων. Από άποψη ταχύτυτας υπερτερεί η μέθοδος σάρωσης, όμως επιλέχθηκε η μέθοδος εξοικονόμησης ως βάση για την μετέπειτα βελτιστοποίηση διότι αποδείχθηκε κατα μικρό ποσοστό αποτελεσματικότερη τόσο στην τιμή της αντικειμενικής συνάρτησης όσο και στην πρόβλεψη για τον απαιτούμενο αριθμό οχημάτων.

Όσον αφορά την υλοποίηση του αλγορίθμου GVNS καταφέρνει όπως φαίνεται από τους πίνακες **7.16** και **7.17** να λύσει τα στιγμιότυπα αυτής της κατηγορίας μεγέθους αφού σε μόλις 30 δευτερόλεπτα εκτέλεσης μας δίνει μέσο σφάλμα 0.4% και 1.4% αντίστοιχα. Είναι λογικό και αναμενόμενο πως όσο μεγαλώνει η διασταση των προβλημάτων τόσο περισσότερος χρόνος χρειάζεται για την επίλυση τους. Στις κατηγορίες μεγέθους 40-50, 50-60 και 60-70 ο αλγόριθμος δίνει μέσο σφάλμα 3.2%, 4.5% και 5.6% αντίστοιχα για εκτέλεση διάρκειας 30 δευτερολέπτων, ενώ οι αντίστοιχες τιμές των μέσων σφαλμάτων μειώνονται σε 2.8%, 3.6% και 4.7% αντίστοιχα για εκτέλεση διάρκειας 60 δευτερολέπτων (πίνακες **7.18**, **7.19** και **7.20**). Στα μεγαλύτερα από τα στιγμιότυπα που δοκιμάστηκαν με διάσταση μεγαλύτερη από 70 ο αλγόριθμος κατάφερε να δώσει μέσο σφάλμα της τάξης του 6.4% όταν εκτελείται για 30 δευτερόλεπτα, το οποίο μειώνεται σε 5.7% για εκτέλεση διάρκειας 60 δευτερολέπτων.

Παρόλο που είναι δύσκολο να συγκρίνουμε διαφορετικές υλοποιήσεις, με διαφορετικές παραμέ-

τρους και φυσικά και με διαφορετική διαθέσιμη υπολογιστική ισχύ, μπορούμε να δούμε ότι παίρνουμε ικανοποιητικά αποτελέσματα για τους χρόνους εκτέλεσης που δοκιμάστηκαν. Στη διεθνή βιβλιογραφία κυριαρχούν συνδυαστικές υλοποιήσεις με την GVNS και οι παραλλαγές της παίζουν πρωταγωνιστικό ρόλο στη σύγχρονη έρευνα. Πολλές από αυτές τις υλοποιήσεις που προτείνονται από τους ερευνητές του πεδίου συνδυάζουν διάφορες τεχνικές από διαφορετικές μεθευρετικές μεθόδους (Escobar, Linfati, Baldoquin, and Toth 2014, Faiz, Arief, et al. 2018) για να επιτύχουν ακόμα πιο γρήγορους χρόνους εκτέλεσης ή παρουσιάζουν τεχνικές που έχουν σκοπό την αντιμετώπιση στιγμιοτύπων πολύ μεγάλου μεγέθους (Kytöjoki, Nuortio, Bräysy, and Gendreau 2007).

8.2 Μελλοντική Έρευνα

Ένα από τα σημαντικότερα κομμάτια της παρούσας εργασίας ήταν η υλοποίηση των αλγορίθμων στη γλώσσα Python. Η αποτελεσματική υλοποίηση αποτελεί πρόκληση για κάθε ερευνητή και παίζει σημαντικό ρόλο στην παραγωγή καλών λύσεων. Ένα ενδεχόμενο πεδίο μελλοντικής έρευνας θα ήταν βελτιστοποίηση του κώδικα και η επέκταση του έτσι ώστε με νέες γειτονιές να μπορούν να αποδώσουν ακόμα καλύτερα αποτελέσματα.

Ακόμη οι νέες γειτονιές μπορούν να βοηθήσουν ο αλγόριθμος με την κατάλληλη προσαρμογή να μπορεί να ανταπεξέλθει και σε άλλες παραλλαγές του VRP. Επίσης ένα ακόμα κομμάτι που επιδέχεται ακόμη μεγαλύτερη διερεύνηση είναι ο ρόλος που παίζουν οι μεταβλητές που καθορίζουν την λειτουργία της VNS όπως είναι ο αριθμός των γειτονιών που πρέπει να χρησιμοποιηθούν, το ποιες γειτονιές είναι οι πιο χρήσιμες αλλά ακόμη και η σειρά που τις εξετάζει η VND.

Όλα αυτά θα είναι εξαιρετικά χρήσιμα στο να κατανοήσουμε βαθύτερα τη φύση αυτών των τόσο περίπλοκων προβλημάτων.

ΑΝΑΦΟΡΕΣ

Aarts, E. and J. Korst (1988). Simulated annealing and boltzmann machines.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life reviews 2*(4), 353–373.

CHRISTOPHER, M. (1999). Logistics and supply chain management: Strategies for reducing cost and improving service (second edition). *International Journal of Logistics Research and Applications 2*(1), 103–104.

Clarke, G. and J. W. Wright (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research 12*(4), 568–581.

Cook, S. (2006). The p versus np problem. *The millennium prize problems*, 87–104.

Cordeau, J.-F., M. Gendreau, and G. Laporte (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks: An International Journal 30*(2), 105–119.

Crevier, B., J.-F. Cordeau, and G. Laporte (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research 176*(2), 756–773.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research 6*(6), 791–812.

Dantzig, G. B. and J. H. Ramser (1959). The truck dispatching problem. *Management Science 6*(1), 80–91.

de Werra, D. and A. Hertz (1989). Tabu search techniques. *Operations-Research-Spektrum 11*(3), 131–141.

Doerner, K. F., R. F. Hartl, G. Kiechle, M. Lucka, and M. Reimann (2004). Parallel ant systems for the capacitated vehicle routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pp. 72–83. Springer.

Dorigo, M. and M. Birattari (2010). *Ant colony optimization*. Springer.

Dorigo, M. and T. Stützle (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of metaheuristics*, pp. 250–285. Springer.

Eberhart, R. and J. Kennedy (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43. Ieee.

Eiben, A. E., P.-E. Raue, and Z. Ruttkay (1994). Genetic algorithms with multi-parent recombination. In *International Conference on Parallel Problem Solving from Nature*, pp. 78–87. Springer.

Englert, M., H. Röglin, and B. Vöcking (2007). Worst case and probabilistic analysis of the 2-opt algorithm for the tsp: Extended abstract. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, Philadelphia, PA, USA, pp. 1295–1304. Society for Industrial and Applied Mathematics.

Escobar, J. W., R. Linfati, M. G. Baldoquin, and P. Toth (2014). A granular variable tabu neighborhood search for the capacitated location-routing problem. *Transportation Research Part B: Methodological 67*, 344–356.

Faiz, A., U. M. Arief, et al. (2018). An efficient meta-heuristic algorithm for solving capacitated vehicle routing problem. *International Journal of Advances in Intelligent Informatics 4*(3), 212–225.

Fortnow, L. (2009). The status of the p versus np problem. *Communications of the ACM 52*(9), 78–86.

Gavish, B. and S. C. Graves (1978). The travelling salesman problem and related problems.

Gendreau, M., A. Hertz, and G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. *Management science 40*(10), 1276–1290.

Gillett, B. E. and L. R. Miller (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research 22*(2), 340–349.

Glover, F. (1989). Tabu search - part i. *ORSA Journal on computing 1*(3), 190–206.

Glover, F. W. and G. A. Kochenberger (2006). *Handbook of metaheuristics*, Volume 57. Springer Science & Business Media.

Golden, B., S. Raghavan, and E. Wasil (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces Series. Springer US.

Golden, B. L., E. A. Wasil, J. P. Kelly, and I.-M. Chao (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In *Fleet management and logistics*, pp. 33–56. Springer.

Hansen, P. and N. Mladenović (1999). An introduction to variable neighborhood search. In *Meta-heuristics*, pp. 433–458. Springer.

Hansen, P. and N. Mladenović (2001). Variable neighborhood search: Principles and applications. *European journal of operational research 130*(3), 449–467.

Hansen, P. and N. Mladenović (2002). Developments of variable neighborhood search. In *Essays and surveys in metaheuristics*, pp. 415–439. Springer.

Hansen, P. and N. Mladenović (2003). Variable neighborhood search. In *Handbook of metaheuristics*, pp. 145–184. Springer.

Hansen, P. and N. Mladenović (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics 154*(5), 802–817.

Hansen, P., N. Mladenović, J. Brimberg, and J. A. M. Pérez (2019). Variable neighborhood search. In *Handbook of metaheuristics*, pp. 57–97. Springer.

Hertz, A. and D. de Werra (1990). The tabu search metaheuristic: how we used it. *Annals of Mathematics and Artificial Intelligence 1*(1), 111–121.

Joshi, S. and S. Kaur (2015). Nearest neighbor insertion algorithm for solving capacitated vehicle routing problem. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 86–88. IEEE.

Kennedy, J. (2006). Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pp. 187–219. Springer.

Kennedy, J. (2010). Particle swarm optimization. *Encyclopedia of machine learning*, 760–766.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *science 220*(4598), 671–680.

Knuth, D. E. (1974, April). Postscript about np-hard problems. *SIGACT News 6*(2), 15–16.

Kytöjoki, J., T. Nuortio, O. Bräysy, and M. Gendreau (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research 34*(9), 2743–2757.

Lai, D. S., O. C. Demirag, and J. M. Leung (2016). A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transportation Research Part E: Logistics and Transportation Review 86*, 32–52.

Laporte, G. and F. Semet (2002). Classical heuristics for the capacitated vrp. In *The vehicle routing problem*, pp. 109–128. SIAM.

Lourenço, H. R., O. C. Martin, and T. Stützle (2003). Iterated local search. In *Handbook of metaheuristics*, pp. 320–353. Springer.

Lysgaard, J. (1997). Clarke & wright savings algorithm. *Department of Management Science and Logistics, The Aarhus School of Business 44*.

Mladenović, N. and P. Hansen (1997). Variable neighborhood search. *Computers & operations research 24*(11), 1097–1100.

Ochoa, G., S. Verel, and M. Tomassini (2010). First-improvement vs. best-improvement local optima networks of nk landscapes. In *International Conference on Parallel Problem Solving from Nature*, pp. 104–113. Springer.

Osman, I. H. and G. Laporte (1996, Oct). Metaheuristics: A bibliography. *Annals of Operations Research 63*(5), 511–623.

Papadimitriou, C. H. (1977). The euclidean travelling salesman problem is np-complete. *Theoretical computer science 4*(3), 237–244.

Pearl, J. (1984). Heuristics: intelligent search strategies for computer problem solving.

Psaraftis, H. N. (1983). k-interchange procedures for local search in a precedence-constrained routing problem. *European Journal of Operational Research 13*(4), 391–402.

Ralphs, T. K., L. Kopman, W. R. Pulleyblank, and L. E. Trotter (2003). On the capacitated vehicle routing problem. *Mathematical programming 94*(2-3), 343–359.

Shi, Y. et al. (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, Volume 1, pp. 81–86. IEEE.

Sicilia, J. A., C. Quemada, B. Royo, and D. Escuín (2016). An optimization algorithm for solving the rich vehicle routing problem based on variable neighborhood search and tabu search metaheuristics. *Journal of Computational and Applied Mathematics 291*, 468–477.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research 35*(2), 254–265.

Ting, C.-J. and C.-H. Chen (2013). A multiple ant colony optimization algorithm for the capacitated location routing problem. *International Journal of Production Economics 141*(1), 34–44.

Toth, P. and D. Vigo (2002). *The vehicle routing problem*. SIAM.

Van Laarhoven, P. J. and E. H. Aarts (1987). Simulated annealing. In *Simulated annealing: Theory and applications*, pp. 7–15. Springer.

Vincent, F. Y., A. P. Redi, Y. A. Hidayat, and O. J. Wibowo (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing 53*, 119–132.

Wang, C., D. Mu, F. Zhao, and J. W. Sutherland (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering 83*, 111–122.

Yu, B., Z.-Z. Yang, and B. Yao (2009). An improved ant colony optimization for vehicle routing problem. *European journal of operational research 196*(1), 171–176.