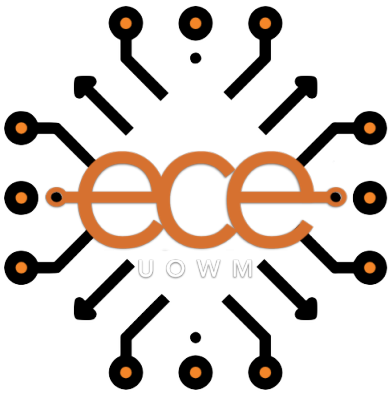


Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αλγόριθμοι συνδυαστικής βελτιστοποίησης για το πρόβλημα του πλανόδιου πωλητή

Βασίλειος Μαρκόπουλος (ΑΜ: 1506)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

29 Φεβρουαρίου 2024

Περίληψη

Το πρόβλημα του πλανόδιου πωλητή αναφέρεται σε ένα απο τα πιο γνωστά προβλήματα στον χώρο των υπολογιστών και της βελτιστοποίησης. Πιο συγκεκριμένα μελετάει έναν πλανόδιο πωλητή ο οποίος πρέπει να κάνει τη συντομότερη διαδρομή σε ένα δίκτυο από πόλεις χωρίς να επισκεφτεί κάποια από τις πόλεις πάνω από μία φορά και η διαδρομή πρέπει να τελειώσει στην πόλη που ξεκίνησε. Στη διπλωματική θα εξεταστούν διάφοροι τύποι αλγορίθμων από διάφορες κατηγορίες όπως προσεγγιστικοί, μεθευρετικοί, επαναληπτικής βελτίωσης αλλά και λύσεις με προγραμματισμό ακεραίων με τη βοήθεια του λογισμικού Gurobi. Επιπλέον, η παρούσα διπλωματική εξετάζει πιθανές βελτιστοποιήσεις και προσαρμογές των αλγορίθμων αυτών, καθώς και την εφαρμογή τους σε πραγματικά προβλήματα του πλανόδιου πωλητή. Τέλος, πραγματοποιείται εξήγηση και ανάλυση των αποτελεσμάτων της έρευνας σε σχέση με τις λύσεις που είναι αποδεδειγμένες.

Λέξεις κλειδιά: Πρόβλημα Πλανόδιου Πωλητή, Συνδυαστική Βελτιστοποίηση, Θεωρία Γράφων, Αλγόριθμοι Γράφων, Gurobi

Abstract

The traveling salesman problem refers to one of the most well-known problems in computing and optimization. More specifically, it studies a traveling salesman who has to find the shortest route through a network of cities without visiting any of the cities more than once, and the route has to end in the city he started from. In the thesis, various types of algorithms from various categories will be examined such as approximate, metahuristic, iterative improvement as well as programming solutions with the help of the Gurobi software. In addition, this thesis examines possible optimizations and adaptations of these algorithms, as well as their application to real traveling salesman problems. Finally, the research results are explained and analyzed in comparison to the proven solutions.

Keywords: Traveling Salesman Problem, Combinatorial Optimization, Graph Theory, Graph Algorithms, Gurobi

Δήλωση Πνευματικών Δικαιωμάτων

Δήλωση Πνευματικών Δικαιωμάτων Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Αλγόριθμοι συνδυαστικής βελτιστοποίησης για το πρόβλημα του πλανόδιου πωλητή" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Νικόλαο Πλόσκα αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Υπογραφή Φοιτητή

Περιεχόμενα

1 Εισαγωγή **10**

1.1 Το Πρόβλημα του Πλανόδιου Πωλητή 10

1.2 Κίνητρα και Στόχοι Υλοποίησης της Εργασίας 10

1.3 Δομή Εργασίας 11

2 Βιβλιογραφική Ανασκόπηση **13**

2.1 Ιστορική Αναδρομή 13

2.2 Μελέτη ως Πρόβλημα Γράφων 15

2.3 Μοντελοποίηση Miller-Tucker-Zemlin (MTZ) 16

2.4 Κατηγορίες Αλγορίθμων για την Επίλυσή του 18

2.5 Παραλλαγές και Κατηγορίες του Προβλήματος 20

2.5.1 Κλασικό TSP: Χαρακτηριστικά και Προκλήσεις 20

2.5.2 Συμμετρικό TSP: Βελτιστοποίηση Συμμετρικών Διαδρομών . . 21

2.5.3 Ασύμμετρο TSP: Διαχείριση Μεταβλητού Κόστους Ταξιδιού . . 21

2.5.4 Χρονοεξαρτώμενο TSP: Μοντελοποίηση Δυναμικών Χρόνων Ταξιδιού 21

2.5.5 Πολλαπλό TSP: Συντονισμός Πολλαπλών Πωλητών 22

3 Αλγόριθμοι Επίλυσης για το Πρόβλημα **23**

3.1 Brute Force 24

3.2 Nearest Neighbour 25

3.3 Christofides 28

3.4 Hill climbing 30

3.5 2-opt 33

3.6 Ant Colony Optimization 35

3.7 Gurobi 38

4 Υπολογιστικά Αποτελέσματα **40**

4.1 Προβλήματα που Μελετήθηκαν 40

4.2 Αποτελέσματα για τα Προβλήματα της Γεννήτριας Τυχαίων Αριθμών . 42

4.3 Αποτελέσματα για τα Προβλήματα της Βιβλιοθήκης TSPLIB 45

4.4 Γραφικές Παραστάσεις Αποτελεσμάτων Χρόνου Εκτέλεσης και Ποιότητας Λύσης 51

5 Συμπεράσματα **54**

5.1 Γενικά Συμπεράσματα 54

5.2 Μελλοντικές Εξελίξεις 55

Α΄ Αποτελέσματα Βιβλιοθήκης TSPLIB **59**

Κατάλογος σχημάτων

3.1 Παραδείγμα Brute Force 25
3.2 Παραδείγμα Nearest Neighbour 27
3.3 Παράδειγμα Christofides 30
3.4 Παραδείγμα Hill Climbing 33
3.5 Παραδείγμα 2-opt 35
3.6 Παραδείγμα Ant Colony Optimization(ACO) 38

4.1 Χρόνος Εκτέλεσης Αλγορίθμων 52
4.2 Απόκληση Αλγορίθμων από τη Βέλτιστη Λύση 53
4.3 Μέση Απόκληση Αλγορίθμων από τη Βέλτιστη Λύση 53

Κατάλογος αλγορίθμων

1 Brute Force 25
2 Nearest Neighbor 26
3 Christofides 29
4 Hill Climbing 32
5 2-opt 34
6 Ant Colony Optimization 37

Κατάλογος πινάκων

2.1 Ιστορικά Ορόσημα με Βάση το Μέγεθος 15

4.1 Προβλήματα Γεννήτριας Τυχαίων Αριθμών 40
4.2 Προβλήματα Βιβλιοθήκης TSPLIB 41
4.3 Χαρακτηριστικά Υπολογιστή για τις Προσομοιώσεις 41
4.4 Πρόβλημα rng5 42
4.5 Πρόβλημα rng8 43
4.6 Πρόβλημα rng10 43
4.7 Πρόβλημα rng12 44
4.8 Πρόβλημα rng15 44
4.9 Αλγόριθμος Nearest Neighbor 46
4.10 Αλγόριθμος Hill Climbing 47
4.11 Αλγόριθμος 2-opt 48
4.12 Αλγόριθμος Christofides 49
4.13 Αλγόριθμος ACO 50
4.14 Αλγόριθμος ACO 51

Α΄.1 Πρόβλημα eil51 59
Α΄.2 Πρόβλημα berlin52 60
Α΄.3 Πρόβλημα pr76 61
Α΄.4 Πρόβλημα rat99 63
Α΄.5 Πρόβλημα bier127 65
Α΄.6 Πρόβλημα ch130 65
Α΄.7 Πρόβλημα tsp225 66
Α΄.8 Πρόβλημα a280 66
Α΄.9 Πρόβλημα pr439 67
Α΄.10Πρόβλημα rat575 67

Α΄.11 Πρόβλημα p654 68
Α΄.12 Πρόβλημα rat783 68
Α΄.13 Πρόβλημα pr1002 68
Α΄.14 Πρόβλημα pcb1173 69
Α΄.15 Πρόβλημα u1432 69
Α΄.16 Πρόβλημα rl1889 69
Α΄.17 Πρόβλημα pr2392 70
Α΄.18 Πρόβλημα pcb3038 70
Α΄.19 Πρόβλημα fnl4461 71
Α΄.20 Πρόβλημα rl5934 71

Κεφάλαιο 1

Εισαγωγή

1.1 Το Πρόβλημα του Πλανόδιου Πωλητή

Το πρόβλημα του πλανόδιου πωλητή είναι ένα πρόβλημα βελτιστοποίησης που σκοπός του είναι η εύρεση της συντομότερης διαδρομής σε ένα πλήθος από πόλεις που ενώνονται μεταξύ τους με δρόμους. Ένας πωλητής ξεκινάει από μία πόλη και προσπαθεί να βρει τη διαδρομή η οποία θα τον οδηγήσει από όλες τις πόλεις που πρέπει να περάσει, χωρίς να επισκεφθεί κάποια από αυτές δεύτερη φορά και η τελική του πόλη να είναι ίδια με την αρχική.

Η αξία του προβλήματος αυτού γίνεται φανερή μόλις γίνει κατανοητό πόσο δύσκολο είναι να βρεθεί η διαδρομή ελαχίστου μήκους. Αυτό συμβαίνει γιατί για ανήκει σε μία κατηγορία προβλημάτων που ονομάζονται NP-Hard. Για τα παραπάνω προβλήματα υπάρχουν μέθοδοι επαλήθευσης τους σε πολυωνυμικό χρόνο, αλλά δεν υπάρχουν τρόποι επίλυσης τους σε πολυωνυμικό χρόνο. Συνεπώς, αφού δεν υπάρχουν αλγόριθμοι πολυωνυμικού χρόνου τα προβλήματα μεγάλου μεγέθους (από εκατοντάδες και μετά) γίνεται πολύ δύσκολο και χρονοβόρο το να λυθούν. Η δυσκολία επίλυσης του προβλήματος του πλανόδιου πωλητή αντικατοπτρίζει την περίπλοκη φύση του, ενισχύοντας τη σημασία της ανάπτυξης αποτελεσματικών αλγορίθμων βελτιστοποίησης για πραγματικές εφαρμογές, κυρίως σε προβλήματα μεγάλης κλίμακας.

1.2 Κίνητρα και Στόχοι Υλοποίησης της Εργασίας

Η επιλογή του προβλήματος αυτού για τη συγγραφή της εργασίας είναι η σημαντικότητα του στη σε πολλούς τομείς της επιστήμης, κυρίως των υπολογιστών, αλλά

όχι μόνο. Ανήκει στη κατηγορία NP-Complete, το οποίο σημαίνει ότι αν βρεθεί ένας αλγόριθμος για την επίλυση του μπορεί να χρησιμοποιηθεί για την επίλυση όλων των προβλημάτων της κατηγορίας NP. Είναι ένα πρόβλημα το οποίο συναντάται παντού καθημερινά και ακόμα δεν υπάρχει καλός αλγόριθμος για την επίλυσή του [1].

Στόχος της εργασίας είναι η υλοποίηση αποδοτικών αλγορίθμων και η υπολογιστική μελέτη στα αποτελέσματα των αλγορίθμων τόσο στην ποιότητα της λύσης όσο και στον χρόνο εκτέλεσης τους. Με την ολοκλήρωση αυτής της εργασίας θα υπάρχει μια πιο βαθιά κατανόηση στο πρόβλημα του πλανόδιου πωλητή που θα βοηθήσει στην πιο αποτελεσματική μελέτη του.

1.3 Δομή Εργασίας

Στο 2ο κεφάλαιο θα υπάρξει θεωρητική μελέτη του προβλήματος. Αρχικά θα ακολουθήσουν ιστορικά γεγονότα σχετικά με το πρόβλημα του πλανόδιου πωλητή. Στη συνέχεια θα υπάρξει επεξήγηση του προβλήματος ως πρόβλημα γράφου και μαθηματική περιγραφή του. Επιπλέον, θα υπάρξει μελέτη του προβλήματος ως πρόβλημα ακέραιου γραμμικού προγραμματισμού όπως το μοντελοποίησαν οι Miller – Trucker – Zemlin. Επίσης, θα αναφερθούν οι διάφορες κατηγορίες των αλγορίθμων που χρησιμοποιούνται για την επίλυση του προβλήματος και οι διαφορές μεταξύ τους. Τέλος, θα αναφερθούν κάποιες κατηγορίες και παραλλαγές του προβλήματος του πλανόδιου πωλητή όπως το s-TSP, a-TSP και m-TSP.

Στο 3ο κεφάλαιο θα υπάρξει ανάλυση και επεξήγηση των αλγορίθμων που υλοποιήθηκαν. Αναλυτικά παραδείγματα για κάθε αλγόριθμο και ανάλυση του ψευδοκώδικα του. Επίσης, θα υπάρχουν απλά, εικονικά παραδείγματα πάνω σε γράφο μικρού μεγέθους για την καλύτερη κατανόηση των αλγορίθμων.

Στο 4ο κεφάλαιο θα αναλυθούν αποτελέσματα που προέκυψαν των αλγορίθμων που υλοποιήθηκαν. Τα αποτελέσματα χωρίζονται σε δύο διαφορετικές κατηγορίες. Αυτά που είναι παράχθηκαν τυχαία με τη βοήθεια του υπολογιστή και αυτά που υπάρχουν στην πλατφόρμα TSPLIB. Θα γίνει μελέτη και επεξήγηση των αποτελεσμάτων τόσο ως προς την ταχύτητα τους και ως προς την ποιότητα της λύσης τους. Τελικά θα παρουσιαστούν γραφικές παραστάσεις, με τους χρόνους εκτέλεσης και τις ποιότητες των λύσεων, που δείχνουν τη διαφορά ανάμεσα στους αλγορίθμους

πιο ξεκάθαρα.

Στο 5ο και τελευταίο κεφάλαιο θα υπάρχουν τα συμπεράσματα που προέκυψαν από την έρευνα που έγινε και γενικά αποτελέσματα για τους αλγορίθμους που υλοποιήθηκαν. Τέλος, θα εξεταστούν πιθανές βελτιώσεις αλλά και προτάσεις για μελλοντικές εργασίες που αποτελούν επέκταση αυτής.

Κεφάλαιο 2

Βιβλιογραφική Ανασκόπηση

2.1 Ιστορική Αναδρομή

Δεν υπάρχει σαφής προέλευση για το πρόβλημα του πλανόδιου πωλητή. Τα πρώτα ίχνη εντοπίζονται στις αρχές του 19ου αιώνα. Το πρόβλημα απέκτησε περισσότερη έμφαση στις δεκαετίες του 1930 και του 1940 όταν μελετήθηκε από μαθηματικούς και ερευνητές επιχειρήσεων.

Πρόβλημα Χαμιλτόνιου Κυκλώματος (19ος αιώνας): Η πρώτη σχετική έρευνα με το πρόβλημα εμφανίζεται το πρόβλημα του πλανόδιου πωλητή σχετίζεται στενά με το Πρόβλημα Χαμιλτόνιου Κυκλώματος το οποίο περιλαμβάνει την εύρεση ενός κλειστού βρόχου που επισκέπτεται κάθε κορυφή ενός γραφήματος ακριβώς μία φορά. Ο William Rowan Hamilton, ένας Ιρλανδός μαθηματικός, εισήγαγε αυτό το πρόβλημα στα μέσα του 19ου αιώνα.

Πρώτη Μαθηματική Διατύπωση (1930): Το TSP, ως συγκεκριμένο πρόβλημα, άρχισε να κερδίζει την προσοχή στη δεκαετία του 1930. Στα μέσα του 1930 ο Καρλ Μένγκερ, ένας Αυστριακός μαθηματικός, εισήγαγε τον όρο Χαμιλτόνιο κύκλωμα, απόδειξε τη σχέση του με το πρόβλημα του πλανόδιου πωλητή, διατύπωσε το πρόβλημα μαθηματικά και απέδειξε κάποια πρώιμα αποτελέσματα.

Dantzig, Fulkerson και Johnson (1954)[2]: Το TSP κέρδισε μεγαλύτερη σημασία τη δεκαετία του 1950 όταν οι George Dantzig, Ray Fulkerson και Selmer Johnson ανέπτυξαν τη μέθοδο του επιπέδου κοπής (cutting planes) για την επίλυση ακέραιων γραμμικών προγραμμάτων. Εφάρμοσαν αυτή τη μέθοδο για να λύσουν περιπτώσεις του TSP, σημειώνοντας ένα σημαντικό βήμα στις αλγοριθμικές προσεγγίσεις του προβλήματος.

Ανάπτυξη Τεχνικών Βελτιστοποίησης (δεκαετίες 1950-1970): Κατά τη διάρκεια αυτής της περιόδου, διάφορες τεχνικές βελτιστοποίησης, όπως διακλάδωση και περιορισμός (Branch and Bound), δυναμικός προγραμματισμός (Dynamic programming) και γραμμικός προγραμματισμός (linear programming), εφαρμόστηκαν για την επίλυση του TSP. Η ανάπτυξη των υπολογιστών στις δεκαετίες του 1950 και του 1960 επέτρεψε στους ερευνητές να εξερευνήσουν πιο αποτελεσματικούς αλγόριθμους.

Χρήση Ευρετικών Μεθόδων (Heuristics) (1950-1960): Λόγω της NP-Hard φύσης του TSP, οι ακριβείς αλγόριθμοι ήταν περιορισμένοι στην ικανότητά τους να επιλύουν μεγάλες περιπτώσεις. Στις δεκαετίες του 1950 και του 1960, εισήχθησαν ευρετικοί αλγόριθμοι και προσεγγίσεις για την εύρεση σχεδόν βέλτιστων λύσεων πιο αποτελεσματικά. Οι πιο αξιοσημείωτοι ευρετικοί αλγόριθμοι περιλαμβάνουν τον αλγόριθμο του πλησιέστερου γείτονα (Nearest Neighbour) και τον αλγόριθμο ελάχιστης έκτασης δέντρων (Minimum Spanning Tree).

Αποτελέσματα πολυπλοκότητας (δεκαετίες 1970-1980): Στις δεκαετίες του 1970 και του 1980, οι ερευνητές απέδειξαν αποτελέσματα πολυπλοκότητας, δείχνοντας ότι το TSP είναι NP-Hard πρόβλημα. Αυτό συνέβαλε στην κατανόηση της εγγενούς δυσκολίας εύρεσης βέλτιστων λύσεων για μεγάλες περιπτώσεις.

Προόδους στους Αλγόριθμους (1990-Σήμερα): Σε αυτήν την περίοδο οι εξελίξεις στις αλγοριθμικές τεχνικές, όπως οι γενετικοί αλγόριθμοι (Genetic algorithms), η προσομοίωση της ανόπτησης (simulated annealing) και η βελτιστοποίηση αποικίας μυρμηγκιών (Ant Colony Optimization), έχουν εφαρμοστεί στο TSP. Αυτές οι μέθοδοι, μαζί με βελτιώσεις στην υπολογιστική ισχύ, επέτρεψαν στους ερευνητές να αντιμετωπίσουν μεγαλύτερες περιπτώσεις του προβλήματος [3].

Παρακάτω ακολουθεί ένας πίνακας με τα πιο σημαντικά ορόσημα στην ιστορία του προβλήματος σε μέγεθος κόμβων.

Πίνακας 2.1: Ιστορικά Ορόσημα με Βάση το Μέγεθος

Έτος	**Όνόματα Στη δημοσίευση**	**Μέγεθος**	**Αρχείο**
1954	G. Dantzig et al. [2]	49	dantzig42
1971	Held & Karp [4]	64	64 Τυχαίοι
1975	Camerini et al. [5]	67	67 Τυχαίοι
1977	Grötschel [6]	120	gr120
1980	Crowder & Padberg [7]	318	lin318
1987	Padberg & Rinaldi [8]	532	att532
1987	Grötschel & Holland [9]	666	gr666
1987	Padberg & Rinaldi [10]	2,392	pr2392
1994	Applegate et al. [11]	7,397	pla7397
1998	Applegate et al. [12]	13,509	usa13509
2001	Applegate et al. [13]	15,112	d15112
2004	Applegate et al. [11]	24,978	sw24798
2009	Applegate et al. [14]	85,900	pla85900

Μέχρι και τη σημερινή ημερομηνία το μεγαλύτερο πρόβλημα που έχει λυθεί σε μέγεθος, με τη βέλτιστη λύση, είναι το ίδιο από το 2009 και είναι το pla85900. Έχουν λυθεί και πολύ μεγαλύτερα εκατομμυρίων πόλεων αλλά η καλύτερη λύση που έχει βρεθεί είναι στο 1-2% από τη βέλτιστη.

2.2 Μελέτη ως Πρόβλημα Γράφων

Το πρόβλημα του πλανόδιου πωλητή (TSP) είναι ένα κλασικό πρόβλημα βελτιστοποίησης στον τομέα της θεωρίας γραφημάτων και της συνδυαστικής βελτιστοποίησης. Όπως έχει προαναφερθεί ο στόχος είναι να βρεθεί η συντομότερη δυνατή διαδρομή που επισκέπτεται ένα δεδομένο σύνολο πόλεων ξεκινώντας και τελειώνοντας στην ίδια πόλη, με την προϋπόθεση ότι κάθε πόλη πρέπει να επισκεφθεί ακριβώς μία φορά.

Το παραπάνω πρόβλημα μοντελοποιείται σαν πρόβλημα γράφου ως εξής. Οι πόλεις αναπαρίστανται ως κορυφές (κόμβοι) σε έναν πλήρη γράφο συνήθως. Αυτό σημαίνει ότι κάθε ζεύγος διαφορετικών πόλεων συνδέεται με μια ακμή. Στις ακμές αποδίδονται βάρη που αντιπροσωπεύουν τις αποστάσεις ή το κόστος της διαδρομής μεταξύ των πόλεων. Αφού ορίσουμε το πρόβλημα ως έναν γράφο τότε ο στόχος είναι να βρεθεί ένας κύκλος Χάμιλτον (ένας κύκλος που επισκέπτεται κάθε κορυφή ακριβώς μία φορά) με το ελάχιστο δυνατό συνολικό βάρος ακμών. Με άλλα λόγια δεδομένου ενός γραφήματος και μιας αρχικής κορυφής, ποιός είναι ο κύκλος

Χάμιλτον με το μικρότερο συνολικό βάρος;

Η μαθηματική διατύπωση του προβλήματος είναι η εξής. Έστω $G = (V, E)$ να είναι το πλήρες γράφημα μη κατευθυνόμενο, όπου V είναι το σύνολο των κορυφών (πόλεις) και E είναι το σύνολο των ακμών με τα σχετικά βάρη. Σε περίπτωση που το γράφημα δεν είναι πλήρες αναπαρίσταται σαν ένα κατευθυνόμενο γράφημα $G = (V, A)$. Το V αναπαρίσταται ως ένα σύνολο της μορφής $V = 1, 2, 3, ..., n$ με μέγεθος ίσο με τον αριθμό των πόλεων. Το E είναι το σύνολο των ακμών και έχει τη μορφή $E = \{(i, j) : i, j \in V, i < j\}$. Σε περίπτωση τώρα που το γράφημα δεν είναι πλήρες και χρησιμοποιούμε τον δεύτερο τύπο γραφήματος τότε το A αναπαρίσταται ως $A = \{(i, j) : i, j \in V, i \neq j\}$.

Η πλειοψηφία των προβλημάτων που μελετιούνται είναι πάνω σε ένα επίπεδο. Αυτό σημαίνει ότι θα πρέπει να ισχύει η τριγωνική ανισότητα. Έτσι λοιπόν άμα ορίσουμε c_{ji} ως το κόστος από την πόλη i στην πόλη j, τότε για κάθε i,j,k θα πρέπει να ισχύει $c_{\text{ji}} < c_{\text{ik}} + c_{\text{kj}}$. Επίσης, συνήθως τα περισσότερα προβλήματα είναι ευκλείδεια, δηλαδή το κόστος ανάμεσα σε 2 πόλεις υπολογίζεται με την ευκλείδεια απόσταση με τον τύπο $d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$. Αν βρίσκεται σε αυτήν την κατηγορία τότε ο γράφος γίνεται κατευθείαν συμμετρικός. Δηλαδή για κάθε 2 πόλεις i,j που η απόσταση από την πόλη i στην πόλη j είναι c_{ij} και η απόσταση από την j στην i είναι c_{ji} ισχύει $c_{\text{ij}} = c_{\text{ji}}$.

2.3 Μοντελοποίηση Miller-Tucker-Zemlin (MTZ)

Είναι σημαντικό το να καταφέρουμε να φτιάξουμε ένα μοντέλο για να μπορούμε να χρησιμοποιήσουμε προγράμματα που λύνουν προβλήματα ακέραιου γραμμικού προγραμματισμού όπως το CPLEX και το Gurobi για να βρούμε τη λύση. Αυτό ακριβώς κάνει η διατύπωση MTZ (Miller-Tucker-Zemlin). Είναι ένας τρόπος για να μοντελοποιήσουμε το πρόβλημα του Πλανόδιου Πωλητή ως πρόβλημα Γραμμικού Ακέραιου Προγραμματισμού. Αυτό το καταφέρνουμε φτιάχνοντας ένα μαθηματικό μοντέλο με μεταβλητές, περιορισμούς για τις μεταβλητές και μια αντικειμενική συνάρτηση. Η βασική διαφορά ανάμεσα στις μεθόδους που υπάρχουν για τη διατύπωση του προβλήματος είναι ο τρόπος που χειρίζονται τις μεταβλητές και τους περιορισμούς που είναι για το άμα ή λύση είναι ένα κύκλωμα και όχι πολλά μικρότερα κυκλώματα. [15]

Καταρχάς έχουμε 2 σετ τα οποία είναι τα V και E τα οποία είναι αντίστοιχα τα σετ που περιέχουν τις κορυφές (πόλεις) και τις ακμές (δρόμους) ανάμεσα στις πόλεις. Επίσης, έχουμε το την παράμετρο c_{ij} η οποία είναι το κόστος (απόσταση) από την πόλη i στην πόλη j. Τέλος, έχουμε 2 μεταβλητές απόφασης οι οποίες οι τιμές που θα πάρουν είναι η λύση. Η πρώτη είναι η x_{ij} η οποία έχει δυαδική τιμή και αν είναι 1 τότε το η ακμή από την πόλη i στην πόλη j είναι μέσα στη λύση, αλλιώς αν είναι 0 δεν είναι μέσα στη λύση. Η δεύτερη είναι η u_{i} η οποία είναι μια συνεχής ακέραια μεταβλητή η οποία μας δείχνει τη θέση της πόλης i στην τελική διαδρομή που είναι και η λύση. Ο σκοπός του προβλήματος όπως έχει προαναφερθεί είναι να βρούμε την ελάχιστη διαδρομή η οποία περνάει από όλους τους κόμβους, αυτό το καταφέρνουμε με την αντικειμενική συνάρτηση $min \sum_{i \in V} \sum_{j \in V j \neq i} c_{\mathrm{ij}} x_{\mathrm{ij}}$. Πέρα από όλα αυτά υπάρχουν και περιορισμοί οι οποίοι είναι ότι για κάθε πόλη πρέπει να υπάρχει ακριβώς μία ακμή που φεύγει από αυτή και ακριβώς μία ακμή να φεύγει από αυτήν. Επιπλέον, υπάρχουν και οι περιορισμοί που εγγυούνται ότι η διαδρομή θα είναι ένας μόνο συνεχόμενος κύκλος. Αυτοί είναι οι περιορισμοί MTZ οι βεβαιώνουν ότι αμα μια πόλη x_{ij} υπάρχει στην τελική λύση τότε η μεταβλητή u_{i} πρέπει να επισκεφθεί πριν από την u_{j}. Μαθηματικά λοιπόν έχουμε:

Παράμετροι:

$\mathbf{c}_{\mathrm{ij}}$ είναι το κόστος απο την πόλη i στην πόλη j.

Μεταβλητές:

$\mathbf{x}_{\mathrm{ij}}$ Δυαδική μεταβλητή που δείχνει αν η ακμή (i,j) περιλαμβάνεται στη λύση (1 εάν περιλαμβάνεται, 0 διαφορετικά).

$\mathbf{u}_{\mathrm{i}}$ Είναι συνεχής μεταβλητή που δίχνει τη θεση της πόλης i στην τελική διαδρομή.

Αντικειμενική Συνάρτηση:

$$min \sum_{i \in V} \sum_{j \in V j \neq i} c_{\mathrm{ij}} x_{\mathrm{ij}} \tag{2.1}$$

Αυτή είναι η συνάρτηση της οποίας θέλουμε να βρούμε το ελάχιστο και αυτό είναι που θα μας δώσει και το αποτέλεσμα.

Περιορισμοί:

$$\sum_{j \in V, j \neq i} x_{\mathrm{ij}} = 1, \forall i \in V \tag{2.2}$$

$$\sum_{i \in V, i \neq j} x_{\mathrm{ij}} = 1, \forall j \in V \tag{2.3}$$

Οι 2 παραπάνω περιορισμοί χρησιμοποιούνται γιατί κάθε πόλη πρέπει να έχει μία μόνο ακμή που να βγαίνει και μία μόνο ακμή που να μπαίνει για να δημιουργηθεί σωστά η διαδρομή.

$$u_{\mathrm{i}} - u_{\mathrm{j}} + n x_{\mathrm{ij}} \leq n - 1 \qquad \forall i, j \in V, i \neq j, i, j \neq 1 \tag{2.4}$$

Αυτός είναι ο περιορισμός MTZ και ο ρόλος τους είναι η λύση να είναι μια συνεχόμενη διαδρομή και όχι πολλοί μικρότεροι κύκλοι.

$$x_{\mathrm{ij}} \in \{0, 1\}, \qquad \forall i, j \in V \tag{2.5}$$

$$u_1 = 1, 2 \leq u_{\mathrm{i}} \leq n, \qquad \forall i \in V, i \neq 1 \tag{2.6}$$

Επιπλέον, έχουμε τους δευτερεύοντες περιορισμούς για να δείξουμε ότι το $\mathrm{x_{ij}}$ είναι δυαδική μεταβλητή και παίρνει μόνος τις τιμές 0 και 1 και για να ορίσουμε το $\mathrm{u_i}$ [15].

Αυτή η διατύπωση διασφαλίζει ότι η λύση αντιπροσωπεύει μια έγκυρη διαδρομή που επισκέπτεται κάθε πόλη ακριβώς μία φορά και ελαχιστοποιεί το συνολικό κόστος ή την απόσταση που διανύθηκε. Οι περιορισμοί MTZ είναι ζωτικής σημασίας για την εξάλειψη των δευτερευουσών διαδρομών και για τη διασφάλιση ότι η λύση σχηματίζει έναν κλειστό βρόχο.

2.4 Κατηγορίες Αλγορίθμων για την Επίλυσή του

Οι τρόποι με τους οποίους μπορούμε να προσεγγίσουμε το πρόβλημα του πλανόδιου πωλητή είναι πολλοί. Οι βασικότερες κατηγορίες αλγορίθμων που χρησιμοποιούνται για την επίλυση του είναι τέσσερις και διαφέρουν ανάλογα με το πόσος χρόνος υπάρχει διαθέσιμος για να βρεθεί λύση και πόσο κοντά πρέπει να είναι η λύση που θα βρεθεί στη βέλτιστη. Οι κατηγορίες αυτές είναι ακριβείς, προσεγγιστικοί, μεθευρετικοί και τοπικής αναζήτησης. Δεν είναι απόλυτα ξεκάθαρες οι κατηγορίες αλγορίθμων δηλαδή μπορεί ένας αλγόριθμος να ανήκει σε περισσότερες από μια κατηγορίες.

Ξεκινώντας με την πρώτη κατηγορία τους, είναι οι ακριβείς. Αυτοί εγγυούνται ότι θα βρούνε λύση και μάλιστα η λύση που θα βρούνε θα είναι η βέλτιστη, δηλαδή η καλύτερη πιθανή. Το βασικό τους μειονέκτημα είναι ότι πρέπει να ψάξουν όλον τον χώρο αναζήτησης για να βρούνε την καλύτερη λύση. Αυτό τους καθιστά ως τους πιο αργούς σε ταχύτητα αλγορίθμους, με συνέπεια σε προβλήματα μεγάλου μεγέθους να μην μπορούνε και χρησιμοποιηθούνε. Σε αυτήν την κατηγορία ανήκουν αλγόριθμοι όπως ο ωμής βίας (brute force) και κλάδεμα και Περιορισμός (Branch and Bound).

Η δεύτερη κατηγορία είναι οι προσεγγιστικοί. Εγγυούνται και αυτοί ότι θα βρούνε λύση αλλά έχουνε ένα ποσοστό, στο οποίο θα επέχει στη χειρότερη πιθανή περίπτωση από τη βέλτιστη. Αυτοί είναι συνήθως πολύ γρήγοροι σε ταχύτητα οπότε μπορούνε να χρησιμοποιηθούνε ακόμα και σε πολύ μεγάλα προβλήματα. Είναι πολύ χρήσιμοι όταν δεν μας ενδιαφέρει πολύ η ποιότητα της λύσης και θέλουμε τη λύση γρήγορα. Για παράδειγμα σε μεταφορική εταιρία που κάθε μέρα έχουν πολλά και διαφορετικά δρομολόγια αυτή είναι η καλύτερη επιλογή. Το μεγαλύτερο μειονέκτημα είναι ότι δεν είναι σίγουρο πόσο κοντά θα είναι η λύση στη βέλτιστη οπότε αν χρειάζεται μεγαλύτερη ακρίβεια στη λύση δεν αρκούν.

Η τρίτη κατηγορία είναι οι μεθευρετικοί. Αυτοί έχουνε μία στρατηγική που προσπαθεί να οδηγήσει τον αλγόριθμο στην προς τη βέλτιστη λύση. Σκοπός τους είναι να εξερευνηθεί ο χώρος πιο αποτελεσματικά και να βρεθεί γρήγορα λύση που να είναι όσο πιο κοντά γίνεται στη βέλτιστη. Κύριο τους πλεονέκτημα είναι ότι έχουνε συνήθως πολλές παραμέτρους οι οποίες μπορούν να ρυθμιστούν ανάλογα με το επιθυμητό αποτέλεσμα. Οπότε υπάρχει η δυνατότητα να ρυθμιστεί η ποιότητα της λύσης αλλά όσο περισσότερο τόσο πιο πολύ ώρα θα πάρει ο αλγόριθμος να τρέξει. Μπορεί όμως εύκολα αυτό να γίνει μειονέκτημα αν οι παράμετροι δεν ρυθμιστούν σωστά. Δεν έχουνε κάποια εγγύηση με τη λύση, δηλαδή υπάρχουν αλγόριθμοι που δεν είναι σίγουρο ότι θα βρούνε λύση και αν βρούνε μπορεί να απέχει αρκετά από τη βέλτιστη. Σε αυτή την κατηγορία ανήκουν συνήθως αλγόριθμοι που προσπαθούν να αντιγράψουν ένα φαινόμενο της φύσης. Μερικά παραδείγματα είναι γενετικοί αλγόριθμοι (genetic algorithms) που αντιγράφουν τη φυσική επιλογή δηλαδή τη διαδικασία που οδηγεί στη βιολογική εξέλιξη, προσομοίωση ανόπτησης (simulated annealing) που αντιγράφει την ανόπτηση ενός μετάλλου στην τέχνη της μεταλλουρ-

γίας και η βελτιστοποίηση αποικίας μυρμηγκιών (ant colony optimization) που προσομοιώνει πως μια αποικία μυρμηγκιών βρίσκει τη συντομότερη διαδρομή για το φαΐ.

Η τέταρτη και τελευταία από τις βασικές κατηγορίες είναι οι τοπικής αναζήτησης. Αυτοί παίρνουν μία αρχική λύση η οποία κάποιες φορές είναι τυχαία αλλά συνήθως ξεκινάνε από μία λύση καλύτερη η οποία βρίσκεται από έναν γρήγορο αλγόριθμο, παράδειγμα κοντινότερου γείτονα, και στη συνέχεια προσπαθούν επαναληπτικά να τη βελτιώσουν. Αυτοί μπορούν να χρησιμοποιηθούν μετά από κάθε αλγόριθμο που βρήκε λύση για να βελτιώσουν την ποιότητα της. Το κύριο τους μειονέκτημα είναι ότι μπορεί να βρεθούν σε μία τοπικά βέλτιστη λύση και δεν μπορούν να ξεφύγουν από αυτήν. Οπότε δεν μπορούν να τη βελτιώσουν περεταίρω. Εδώ ανήκουν αλγόριθμοι όπως ο 2opt, 3opt και ο αλγόριθμος των Lin-Kernighan.

Συνοψίζοντας λοιπόν οι κατηγορίες των αλγορίθμων για να λύσουμε το πρόβλημα του πλανόδιου πωλητή είναι πολλές. Πολλοί από τους αλγόριθμους δεν ανήκουν καν σε μία κατηγορία αλλά σε περισσότερες από αυτές. Η επιλογή της κατάλληλης κατηγορίας γίνεται με βάση την ποιότητα της λύσης που χρειάζεται, τον διαθέσιμο χρόνο που υπάρχει αλλά και το μέγεθος του προβλήματος.

2.5 Παραλλαγές και Κατηγορίες του Προβλήματος

Σε αυτό το κεφάλαιο μελετιούνται διάφορες κατηγορίες και παραλλαγές του TSP. Δύο βασικές αποτελούν το Συμμετρικό και Ασύμμετρο TSP και στη συνέχεια ακολουθούν δυο ειδικές κατηγορίες το προβλήματος το χρονοεξαρτώμενο TSP και η παραλλαγή του TSP όπου έχουμε πολλούς πωλητές.

2.5.1 Κλασικό TSP: Χαρακτηριστικά και Προκλήσεις

Μέχρι τώρα έχει αναφερθεί μόνο μία κατηγορία του προβλήματος, το Κλασσικό Πρόβλημα του πλανόδιου πωλητή (CTSP), το οποίο είναι η θεμελιώδης κατηγορία των προβλημάτων. Στο CTSP, όπως έχει προαναφερθεί, στόχος είναι να προσδιοριστεί η συντομότερη διαδρομή που επισκέπτεται κάθε πόλη ακριβώς μία φορά και επιστρέφει στο σημείο εκκίνησης. Αυτή η κατηγορία δεν έχει κάποια ιδιαίτερη παραλλαγή και είναι η πιο γενική κατηγορία του προβλήματος με όλες τις απλοποιήσεις. Πέρα από αυτήν όμως έχουμε διάφορες κατηγορίες που θα μελετήσουμε

στη συνέχεια.

2.5.2 Συμμετρικό TSP: Βελτιστοποίηση Συμμετρικών Διαδρομών

Πρώτη βασική κατηγορία είναι το Συμμετρικό TSP (STSP). Αυτό αποτελεί μια συγκεκριμένη παραλλαγή του προβλήματος όπου η απόσταση μεταξύ δύο πόλεων είναι η ίδια ανεξάρτητα από τη σειρά με την οποία επισκέπτονται. Αυτό μαθηματικά σημαίνει ότι αν είναι c_{ij} η απόσταση από την πόλη i στην πόλη j τότε $c_{ij} = c_{ji}$. Συνήθως σε αυτές τις περιπτώσεις οι πόλεις μας δίνονται με συντεταγμένες x,y πάνω σε ένα επίπεδο. Αν συμβαίνει αυτό τότε το πρόβλημα ονομάζεται ευκλείδειο (Eucidean TSP) και οι αποστάσεις από την πόλη A στην πόλη B είναι$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$. Η συμμετρικότητα του προβλήματος αυτού μας βοηθάει στο να αναπτύξουμε αλγορίθμους που μας είναι πιο αποτελεσματικοί από τους γενικούς αλγορίθμους που χρησιμοποιούνται στις υπόλοιπες κατηγορίες του TSP.

2.5.3 Ασύμμετρο TSP: Διαχείριση Μεταβλητού Κόστους Ταξιδιού

Στο Ασύμμετρο TSP (ATSP), οι αποστάσεις μεταξύ των πόλεων δεν είναι απαραίτητα συμμετρικές, πράγμα που σημαίνει ότι το κόστος ταξιδιού από την πόλη A στην πόλη B μπορεί να διαφέρει από το κόστος ταξιδιού από το B στο A. Μαθηματικά λοιπόν αυτό σημαίνει ότι cij ≠ cji. Αυτό εισάγει πολυπλοκότητες από τις ασύμμετρες αποστάσεις, αλλά ταυτόχρονα φέρνει το πρόβλημα σε μία μορφή που αντικατοπτρίζει περισσότερο σενάρια πραγματικού κόσμου. Αυτό στην πραγματικότητα μπορεί να συμβεί για παράδειγμα σε μία διαδρομή που ο δρόμος είναι μονός και δεν υπάρχει η αντίθετη λωρίδα. Εκεί η απόσταση για τη μετάβαση από την πόλη A στην πόλη B θα είναι διαφορετική από την απόσταση της επιστροφής.

2.5.4 Χρονοεξαρτώμενο TSP: Μοντελοποίηση Δυναμικών Χρόνων Ταξιδιού

Μία κατηγορία που φέρνει το πρόβλημα ακόμα πιο κοντά στην πραγματικότητα είναι το Χρονοεξαρτώμενο TSP (TD-TSP). Το παραπάνω εισάγει χρονικές εκτιμήσεις ενσωματώνοντας μεταβλητούς ταξιδιού μεταξύ των πόλεων, αντανακλώντας τη δυναμική του πραγματικού κόσμου, όπως οι διακυμάνσεις της κυκλοφορίας. Σε αυτή την περίπτωση είναι ιδιαίτερα δύσκολη το να βρεθεί λύση και ο κύριος λόγος

είναι ότι αυξάνεται η πολυπλοκότητα υπερβολικά. Ενσωματώνονται πολλοί περιορισμοί που αφορούν τον χρόνο για την επίσκεψη σε κάθε πόλη οπότε αυξάνεται η δυσκολία υπολογισμού της λύσης. Αυτό αλλάζει τον τρόπο προσέγγισης το προβλήματος, από τη στιγμή που οι απλοί αλγόριθμοι δεν ισχύουν πλέον, και εισάγει καινούριους αλγορίθμους διαφορετικής λογικής για την επίλυσή του.

2.5.5 Πολλαπλό TSP: Συντονισμός Πολλαπλών Πωλητών

Το Πρόβλημα Πολλαπλών Ταξιδιωτών Πωλητών (mTSP) επεκτείνει την κλασική σύνθεση εισάγοντας πολλούς πωλητές, καθένας από τους οποίους έχει καθήκον να βρει τη βέλτιστη διαδρομή. Στο mTSP υπάρχει ένα σύνολο πόλεων και ο στόχος είναι να προσδιοριστεί ο βέλτιστος τρόπος αντιστοίχισης κάθε πόλης σε έναν από πολλούς πωλητές, έτσι ώστε κάθε πωλητής να έχει μια κλειστή περιήγηση (επισκέπτεται ένα σύνολο πόλεων και επιστρέφει στην πόλη έναρξης) και η συνολική απόσταση που διανύθηκε να ελαχιστοποιείται. Αυτή η μοντελοποίηση συναντάται στην καθημερινότητα σε εταιρίες που κάνουν μεταφορές και πλησιάζει πολύ στο πρόβλημα δρομολόγησης οχημάτων. Πρόκειται για μία απλοποίηση του, γιατί δεν λαμβάνεται υπόψιν το πόσο φορτίο μπορεί να κουβαλήσει το κάθε φορτηγό και το ποιες είναι οι απαιτήσεις των πελατών. Είναι αρκετά δύσκολο στην επίλυσή του και δεν είναι από τα λιγότερο μελετημένα κομμάτια του TSP.

Κεφάλαιο 3

Αλγόριθμοι Επίλυσης για το Πρόβλημα

Σε αυτό το κεφάλαιο θα εξεταστούν οι αλγόριθμοι που υλοποιήθηκαν. Μία βασική παροχή που γίνεται στους αλγόριθμους αυτούς είναι ότι οι γράφοι πάνω στους οποίους εκτελούνται είναι πλήρης. Οι αλγόριθμοι ανήκουν σε διάφορες κατηγορίες όπως ακριβείς, προσεγγιστικοί και μεθευρετικοί. Πιο συγκεκριμένα μελετήθηκαν οι παρακάτω εφτά αλγόριθμοι.

Ο πρώτος είναι ο Brute Force. Αυτός ανήκει στην κατηγορία των εξαντλητικών αλγορίθμων οπότε είναι ακριβής αλγόριθμος. Συνεπώς εξετάζονται όλοι οι πιθανοί συνδυασμοί των λύσεων και σαν τελική λύση κρατάμε αυτήν με τη μικρότερη τιμή.

Ο δεύτερος αλγόριθμος είναι ο Neighrest Neighbour. Αυτός είναι προσεγγιστικός αλγόριθμος οπότε δεν είναι σίγουρο οτι θα βρει τη βέλτιστη λύση αλλά είναι πολύ γρήγορος στον χρόνο εκτέλεσης. Γενικά μπορεί να μην βρει λύση αλλά με την υπόθεση που κάναμε ότι οι γράφοι είναι πλήρεις πάντα θα υπάρχει λύση.

Ο τρίτος είναι ο Christofides. Ο αλγόριθμος του Christofides [16] ανήκει και αυτός είναι στην κατηγορία των προσεγγιστικών αλγορίθμων και δίνει την εγγύηση ότι η βέλτιστη λύση που θα βρει είναι μιάμιση φορά χειρότερη από τη βέλτιστη λύση. Επίσης, λειτουργεί μόνο σε προβλήματα που είναι συμμετρικά.

Οι επόμενοι δύο είναι ο hill climbing και ο 2-opt. Αυτοί οι 2 ανήκουν στους αλγορίθμους τοπικής αναζήτησης, δηλαδή παίρνουν μία έτοιμη λύση και προσπαθούν να τη βελτιώσουν επαναληπτικά.

Ο έκτος είναι ο ant colony optimization που είναι ο μόνος που υλοποιήθηκε από την κατηγορία μεθευρετικών. Σκοπός του είναι να προσομοιώσει το πως βρίσκουν φαΐ τα μυρμήγκια σε μία αποικία.

Ο έβδομος και τελευταίος είναι η λύση του μοντέλο γραμμικού προγραμματισμού

με τον Gurobi. Αυτό είναι ένα πρόγραμμα το οποίο δέχεται ένα μαθηματικό μοντέλο γραμμικού προγραμματισμού και βγάζει αυτόματα το αποτέλεσμα. Κατατάσσεται και αυτός στην κατηγορία με τους ακριβείς αλγορίθμους αν τρέξει μέχρι τέλους.

3.1 Brute Force

Όπως αναφέρθηκε πιο πριν ο πρώτος αλγόριθμος που μελετήθηκε είναι ο αλγόριθμος Brute Force. Είναι ιδιαίτερα εύκολος στην υλοποίησή του. Ανήκει στους αλγόριθμους εξαντλητικής αναζήτησης, το οποίο σημαίνει ότι πρέπει να εξερευνήσει όλον τον χώρο για να βρει τη λύση. Βέβαια εγγυάται σαν αλγόριθμος ότι θα βρει τη βέλτιστη λύση στο πρόβλημα. Δεν χρησιμοποιείται γενικά σε προβλήματα μεγάλου μεγέθους γιατί πολύ απλά δεν μπορεί να τρέξει λόγο τις πολυπλοκότητας του.

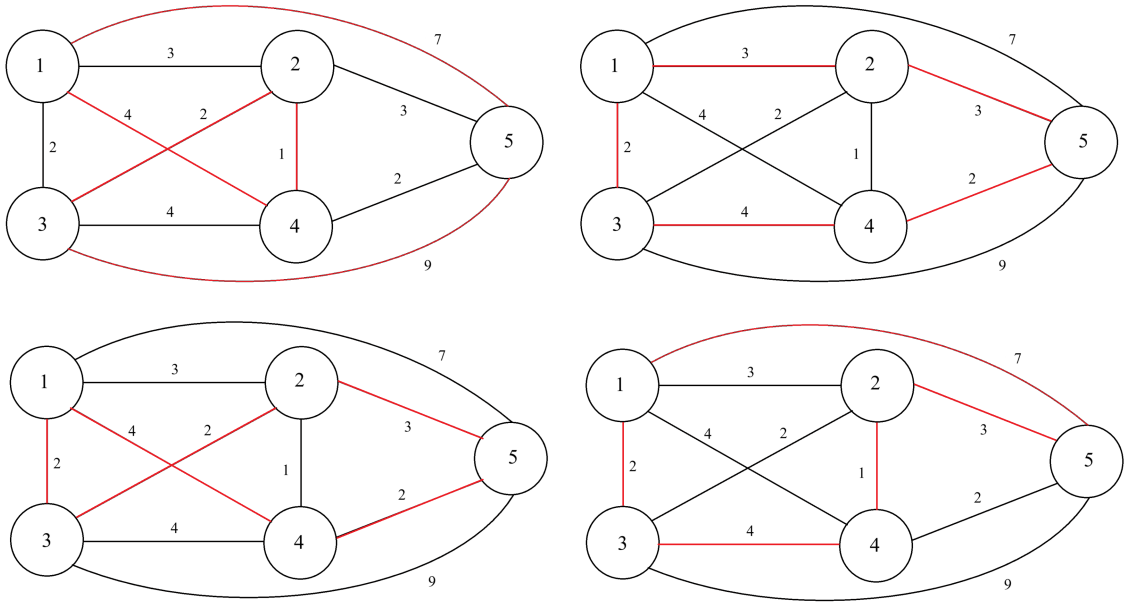
Για παράδειγμα αν έχουμε ένα πρόβλημα με δέκα πόλεις που είναι γενικά μικρού μεγέθους ο αλγόριθμος αυτός θα εκτελέσει 9! = 362,880. Αυτό συμβαίνει γιατί για παράδειγμα αν έχουμε δέκα πόλεις στην πρώτη επιλογή έχει να διαλέξει ανάμεσα από εννιά διαφορετικές πόλεις για να πάει στη συνέχεια. Στο δεύτερο βήμα θα έχει να διαλέξει από 8 διαφορετικές αφού έχουμε επισκεφθεί ήδη την αρχική και άλλη μία στο πρώτο βήμα. Συνεπώς λοιπών το αποτέλεσμα είναι 9*8*...*2=9!, δηλαδή αν υπάρχουν n πόλεις τότε έχουμε n-1! συνδυασμούς. Αυτό κατατάσσει τον αλγόριθμο να είναι στην κατηγορία των O(n!). Είναι η χειρότερη πιθανή κατηγορία σε αλγορίθμους σε πολυπλοκότητα χρόνου αλλά σε αντίθεση χρειάζεται μόνο O(1) χώρο για αποθήκευση της καλύτερης λύσης που έχει βρει μέχρι στιγμής.

Συνεπώς λοιπόν ο αλγόριθμος αυτός τρέχει μόνο σε προβλήματα μικρού μεγέθους μέχρι δεκαοκτώ περίπου πόλεις, το οποίο αλλάζει βέβαια ανάλογα με την υπολογιστική δύναμη που έχει το κάθε σύστημα. Σε μεγαλύτερα προβλήματα χρειάζεται χρόνια για να ολοκληρωθεί. Παρόλα αυτά είναι ιδιαίτερα χρήσιμος για να κρίνουμε πόσο καλή απόδοση έχουν οι αλγόριθμοι οι οποίοι είναι προσεγγιστικοί και μεθευρετικοί.

Παρακάτω ακολουθεί ο Αλγόριθμος 1 που χρησιμοποιήθηκε για τον Brute Force και κάποια στιγμιότυπα από την εκτέλεση για ένα παράδειγμα με 5 κόμβους στο σχήμα 3.1 γιατί είναι αδύνατο να απεικονιστούν όλες οι πιθανές διαφορετικές καταστάσεις που εξετάζει ο αλγόριθμος.

Input: Graph G
Result: Exact Solution
$min\_distance = +\infty$;
$tour = \{1, 2, 3...n\}$;
for *all possible permutations of tour* **do**
    $tour\_distance$ = distance of tour in the current permutation;
    if *tour_distance*< *min_distance* **then**
        $min\_distance = tour\_distance$;
    end
end

Αλγόριθμος 1: Brute Force



Σχήμα 3.1: Παραδείγμα Brute Force

3.2 Nearest Neighbour

Ο δεύτερος αλγόριθμος στη σειρά είναι ο Nearest Neighbour. Ανήκει στην κατηγορία των άπληστων αλγορίθμων, λόγο της λογικής που ακολουθεί, η οποία είναι η επίσκεψη του πλησιέστερου γείτονα. Αυτό σημαίνει ότι παίρνει την καλύτερη απόφαση εκείνη τη στιγμή χωρίς να κρίνει τι επίδραση μπορεί να έχει αυτή στο μέλλον.

Γενικά η παρόλο της ευκολίας του στην υλοποίηση και στον χρόνο που θέλει για να τρέξει επιφέρει πολύ καλά αποτελέσματα [17].

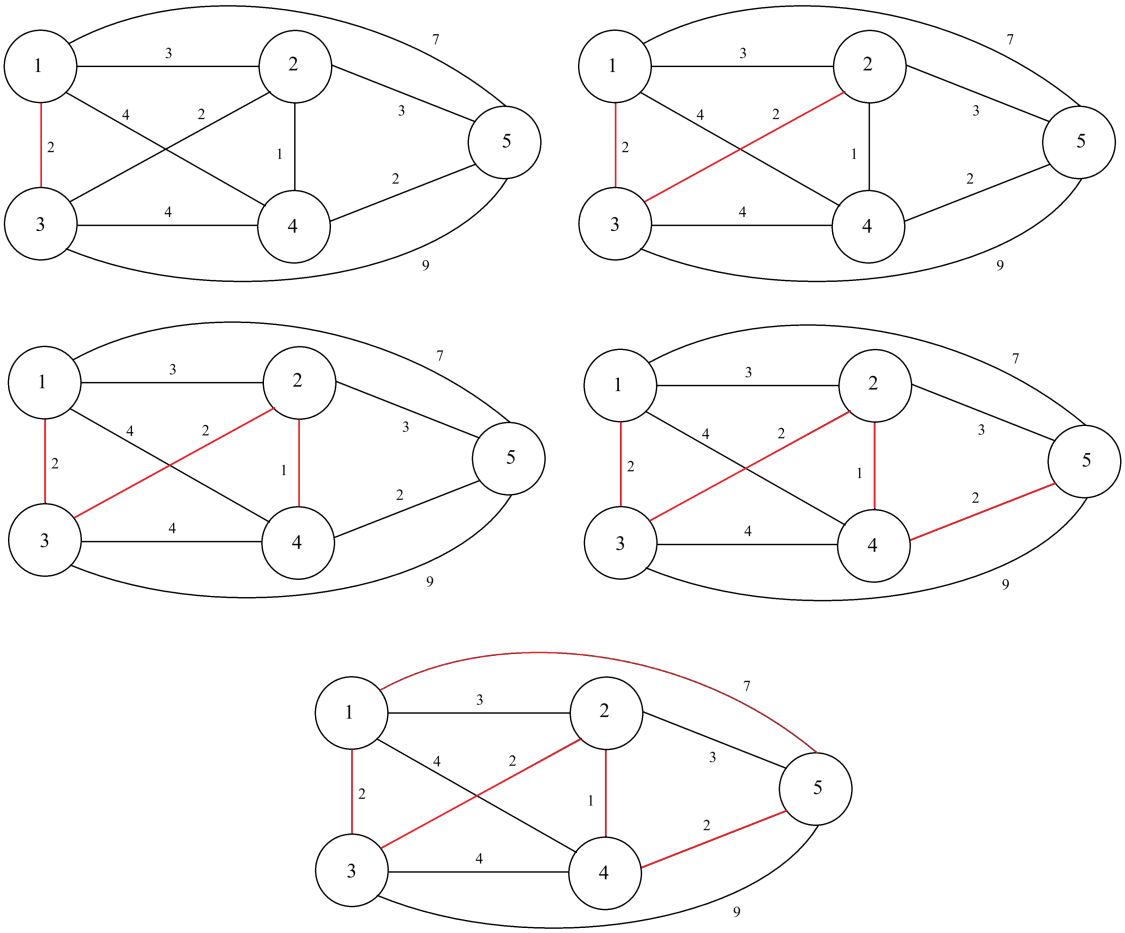
Ο αλγόριθμος που χρησιμοποιήθηκε για το πρόβλημα ακολουθεί 5 απλά βήματα: 1) Αρχικοποίηση όλων των κορυφών ως κορυφές που δεν έχουν επισκεφτεί, 2) Επιλογή της αρχικής κορυφής ως τρέχουσα και ορισμός της ως κουφή που έχει επισκεφτεί, 3) Εύρεση της συντομότερης ακμής που συνδέει την τρέχουσα κορυφή με μια κορυφή που δεν έχει επιστεφθεί, 4) Ορισμός αυτής της κορυφής ως τρέχουσα και επισήμανση αυτής ως κουφή που έχει επισκεφτεί, και 5) Εάν έχουν επισκεφθεί όλες οι κορυφές τότε τερματισμός. Διαφορετικά, μετάβαση στο βήμα 3. Παρακάτω ακολουθεί ο Αλγόριθμος 2 αναλυτικά.

Input: Graph G
Result: Exact Solution
$visited[n] = \{false\}$;
$current = 0$;
$solution$.add($current$);
$visited[current]$ = **true**;
while $solution.size < n$ **do**
    $next$ = minimum distance from current to a not visited city;
    $visited[next]$ = true;
    $solution$.add($next$);
    $current = next$;
end

Αλγόριθμος 2: Nearest Neighbor

Αναλυτικότερα θα ακολουθήσει ένα παράδειγμα με 5 πόλεις που απεικονίζεται στο Σχήμα 3.2. Η αρχική μας πόλη είναι η πόλη 1 από την οποία υπάρχουν 4 επιλογές για να μετακινηθούμε σε άλλες πόλεις. Αυτές είναι οι πόλεις 2, 3, 4, 5 με αντίστοιχα βάρη στις ακμές 3, 2, 4, 7. Οπότε σύμφωνα με τη λογική του αλγορίθμου θα επιλεχθεί η πόλη που συνδέεται με την πιο μικρή ακμή. Εδώ είναι η πόλη 3 που έχει βάρος 2. Στο δεύτερο βήμα υπάρχουν οι επιλογές 2, 4, 5, αφού έχει επισκεφθεί την 1, με αντίστοιχα βάρη 2, 4, 9 οπότε θα μετακινηθεί στην πόλη 2 αφού έχει τη μικρότερη ακμή. Στο τρίτο βήμα έχουν μείνει μόνο οι επιλογές 4 και 5 με 1 και 3. Εδώ θα επιλεγεί η πόλη 4 που έχει το βάρος 1. Στη συνέχεια υπάρχει μόνο μία επιλογή στο τελευταίο βήμα οπότε θα πάει στην πόλη 5 με απόσταση 2 και τέλος

θα επιστρέψει στην αρχική πόλη με απόσταση 7. Οπότε το τελικό αποτέλεσμα της διαδρομής είναι $2 + 2 + 1 + 2 + 7 = 14$.



Σχήμα 3.2: Παραδείγμα Nearest Neighbour

Συμπερασματικά ο αλγόριθμος αυτός μπορεί να τρέξει ακόμα και για πολύ μεγάλα μεγέθη αφού η πολυπλοκότητα του είναι μόνο $O(n^2)$. Γιατί για κάθε βήμα πρέπει να ελέγξει όλες τις πιθανές πόλεις οι οποίες είναι n και αυτό γίνεται για n επαναλήψεις μέχρι να σχηματιστεί το ταξίδι. Ωστόσο, δεν εγγυάται μια βέλτιστη λύση και η απόδοσή του μπορεί να ποικίλλει ανάλογα με τη συγκεκριμένη περίπτωση του TSP. Σε ορισμένες περιπτώσεις, η λύση που παράγεται από τον αλγόριθμο του πλησιέστερου γείτονα μπορεί να είναι σημαντικά μεγαλύτερη από τη βέλτιστη λύση. Παρόλα αυτά, μπορεί να χρησιμεύσει ως καλό σημείο εκκίνησης για πιο εξελιγμένους αλγόριθμους βελτιστοποίησης ή ως γρήγορη λύση για πρακτικές

εφαρμογές όπου δεν απαιτείται ακριβής βέλτιστη.

3.3 Christofides

Ένας ακόμα προσεγγιστικός αλγόριθμος που μελετήθηκε σε αυτήν την εργασία είναι ο αλγόριθμος του Christofides. Όπως και οι περισσότεροι αλγόριθμοι λειτουργεί μόνο σε περιπτώσεις που βρίσκονται σε ευκλείδειο χώρο και το πρόβλημα είναι συμμετρικό. Ο αλγόριθμος πήρε το όνομα του από τον Nikos Christofides που το δημοσίευσε το 1976 και μέχρι και σήμερα αποτελεί έναν από τους πιο αποτελεσματικούς αλγόριθμους για το πρόβλημα του πλανόδιου πωλήτη [16].

Η εγγύηση του σαν προσεγγιστικός είναι ότι η λύση που θα βρει είναι μιάμιση φορά χειρότερη από τη βέλτιστη λύση. Για να ξεκινήσουμε την ανάλυσή μας για τον αλγόριθμο Χριστοφίδη, θα ορίσουμε έστω S τη βέλτιστη λύση στην περίπτωση του Metric-TSP και έστω T η διαδρομή που παράγεται από τον αλγόριθμο προσέγγισης Χριστοφίδη. Επειδή το S περιλαμβάνει ένα Minimum Spanning Tree και το M είναι ένα ελάχιστο εκτεινόμενο δέντρο στο G, $c(M) \leq c(S)$. Επιπλέον, έστω R είναι μια λύση στο πρόβλημα του πλανόδιου πωλητή στο H. Δεδομένου ότι οι ακμές στο G (και, επομένως, στο H) ικανοποιούν την ανισότητα του τριγώνου, και όλες οι ακμές του H είναι επίσης στο G, $c(R) \leq c(S)$. Δηλαδή, η επίσκεψη περισσότερων κορυφών από ό,τι στη διαδρομή R δεν μπορεί να μειώσει το συνολικό κόστος της. Εξετάζουμε τώρα το κόστος μιας τέλειας αντιστοίχισης (Perfect Matching), P, του H, και πώς σχετίζεται με το R, μία βέλτιστη διαδρομή στο H. Απαριθμούμε τις ακμές του R και αγνοούμε το τελευταίο άκρο (που επιστρέφει στην κορυφή της αρχής). Το κόστος του συνόλου των περιττών ακμών και του συνόλου των άρτιων ακμών στο R αθροίζονται σε c(R). Ως εκ τούτου, ένα από αυτά τα δύο σύνολα έχει συνολικό κόστος το πολύ το μισό από αυτό του R, δηλαδή κόστος το πολύ $c(R)/2$. Επιπλέον, το σύνολο των περιττών ακμών και το σύνολο των ζυγών ακμών στο R είναι και τα δύο τέλεια αντιστοίχιση. Ως εκ τούτου, το κόστος του P, ένα τέλειο ταίριασμα ελάχιστου βάρους στις άκρες του H, θα είναι το πολύ μικρότερο από αυτά τα δύο. Δηλαδή, $c(P) \leq c(R)/2$. Επομένως, $c(M)+c(P) \leq c(S)+c(R)/2 \leq 3c(S)/2$. Εφόσον οι άκρες στο G ικανοποιούν την ανισότητα του τριγώνου, μπορούμε να βελτιώσουμε το κόστος μιας περιήγησης μόνο κάνοντας συντομεύσεις που αποφεύγουν τις κορυφές που έχουμε επισκεφτεί στο παρελθόν. Έτσι, $c(T) \leq c(M) + c(P)$, που σημαίνει ότι $c(T) \leq 3c(S)/2$ [18].

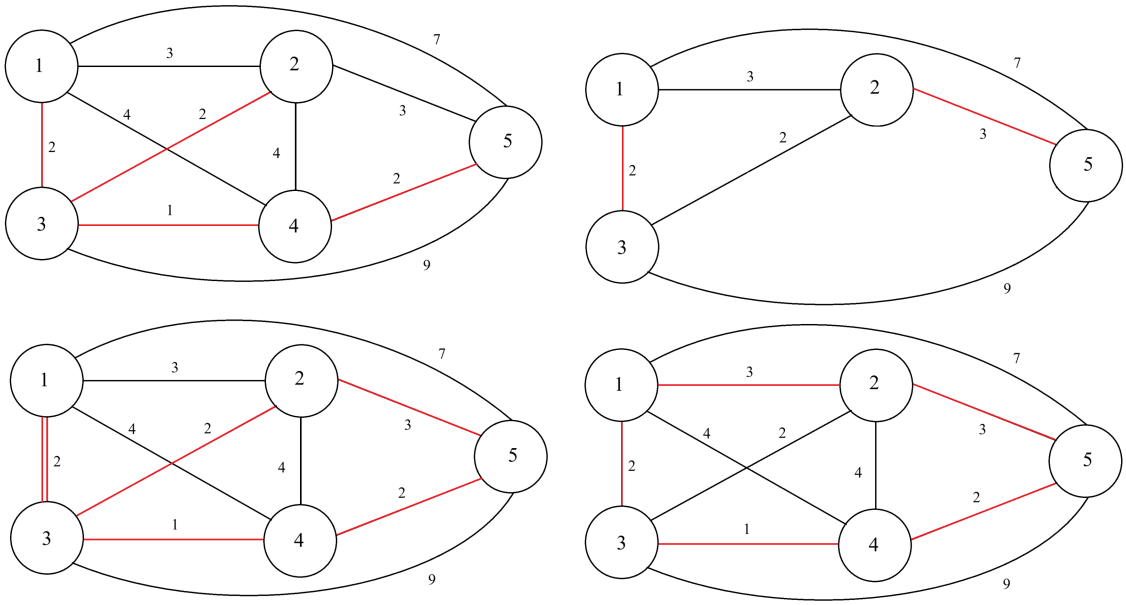
Η λογική του αλγορίθμου είναι αρκετά απλή. Απαρτίζεται από επτά απλά βήματα. Το πρώτο από αυτά είναι ο σχηματισμός του ελαχίστου έκτασης δέντρο (Minimum Spanning Tree) στον γράφο. Στο δεύτερο βήμα δημιουργούμε ένα σετ Ο από κόμβους που έχουν περιττό βαθμό στο παραπάνω δέντρο. Στη συνέχεια δημιουργούμε ένα υπογράφημα του γραφήματος που έχει μόνο τις ακμές του σετ Ο. Στο επόμενο βήμα ενώνουμε βρίσκουμε την τέλεια αντιστοίχιση στο υπογράφημα. Στο πέμπτο βήμα εννοούμε την τέλεια αντιστοίχιση με το MST και δημιουργούμε ένα πολύγραφο όπου κάθε κόμβος έχει άρτιο βαθμό. Στο έκτο βήμα βρίσκουμε το κύκλωμα Euler στον πολύγραφο. Τέλος, μετατρέπουμε το κύκλωμα που βρέθηκε στο προηγούμενο βήμα σε ένα κύκλωμα Hamilton παρακάμπτοντας επαναλαμβανόμενες κορυφές. Πιο αναλυτικά ακολουθεί ένας αλγόριθμος για το πως υλοποιήθηκε.

Input: Graph G
Result: Approximate solution
Construct a minimum spanning tree T of G;
Let O be the set of vertices in T with odd degree;
Find a minimum-weight perfect matching M of the vertices in O;
Merge T and M to form a multigraph H;
Find an Eulerian circuit E in H;
Construct a Hamiltonian circuit H from E by shortcutting;

Αλγόριθμος 3: Christofides

Παρακάτω ακολουθεί ένα παράδειγμα με τον αλγόριθμο το γνώριμο πλέον γράφημα με τους πέντε κόμβους. Αρχικά χρησιμοποιούμε τον αλγόριθμο του Kruskal για να βρούμε το MST στο γράφημα. Ο αλγόριθμος ταξινομεί όλες τις ακμές και τις προσθέτει μία μία στο MST όσο δεν σχηματίζουν κύκλο. Βάζουμε λοιπόν τις ακμές 3-4 που έχει τιμή 1, την 1-3, 2-3 και 4-5 που έχουν όλες την τιμή 2. Έτσι λοιπόν μας μένουν οι κόμβοι 1-2-3-5 με περιττό βαθμό. Δηλαδή έχουν περιττό αριθμό ακμών που ακουμπάνε σε αυτές. Βρίσκουμε την τέλεια αντιστοίχιση αυτών των κόμβων η οποία είναι οι ακμές 1-2 και 2-5 με βάρη 2 και 3 αντίστοιχα. Στη συνέχεια ενώνουμε τα δύο αυτά και δημιουργούμε ένα πολύγραφο όπως φαίνεται στη συνέχεια στο σχήμα που ακολουθεί. Υπολογίζουμε σε αυτόν τον γράφο τον κύκλο Euler ο οποίος σε αυτό το παράδειγμα είναι 1-3-2-5-4-3-1. Τέλος, αφαιρούμε τις επαναλαμβανόμενες ακμές στον κύκλο (shortcutting). Από τη στιγμή που κάνουμε

παράκαμψη δεν γίνεται να αυξηθεί το κόστος του, αφού ισχύει η τριγωνική ανοσιότητα και βγαίνει το αποτέλεσμα του αλγορίθμου που είναι 1-2-5-4-3-1 το οποίο σε αυτό το παράδειγμα είναι και η βέλτιστη λύση. Οπτικά υπάρχουν στο σχήμα 3.3 ο σχηματισμός του MST το Perfect Matching Ο πολύγραφος και το τελικό αποτέλεσμα.



Σχήμα 3.3: Παράδειγμα Christofides

3.4 Hill climbing

Ο επόμενος αλγόριθμος είναι ένας που ανήκει στην κατηγορία των επαναληπτικής βελτίωσης. Στόχος αυτών τον αλγορίθμων είναι να πάρουν μία ήδη έτοιμη λύση και να τη βελτιώσουν περαιτέρω. Ο συγκεκριμένος αλγόριθμος αφού δεχθεί ως είσοδο τη λύση βλέπει αν μπορεί να αλλάξει τη σειρά από δύο πόλεις για να φτάσει σε ένα αποτέλεσμα που είναι πολύ πιο κοντά στη λύση. Γενικά σε αλγορίθμους τέτοιου τύπου είναι συχνό στο πρόβλημα του πλανόδιου πωλητή να παίρνουμε ως αρχική λύση όχι μια τυχαία, αλλά τη λύση που παράγει ο αλγόριθμος πλησιέστερου γείτονα. Αυτό γίνεται γιατί είναι πολύ γρήγορος στην εκτέλεση και σχεδόν πάντα θα αποφέρει μία καλύτερη λύση από αυτήν που θα παραγόταν αν βάζαμε τους κόμβους σε μία τυχαία σειρά. Έτσι λοιπόν έγινε και σε αυτήν την εργασία.

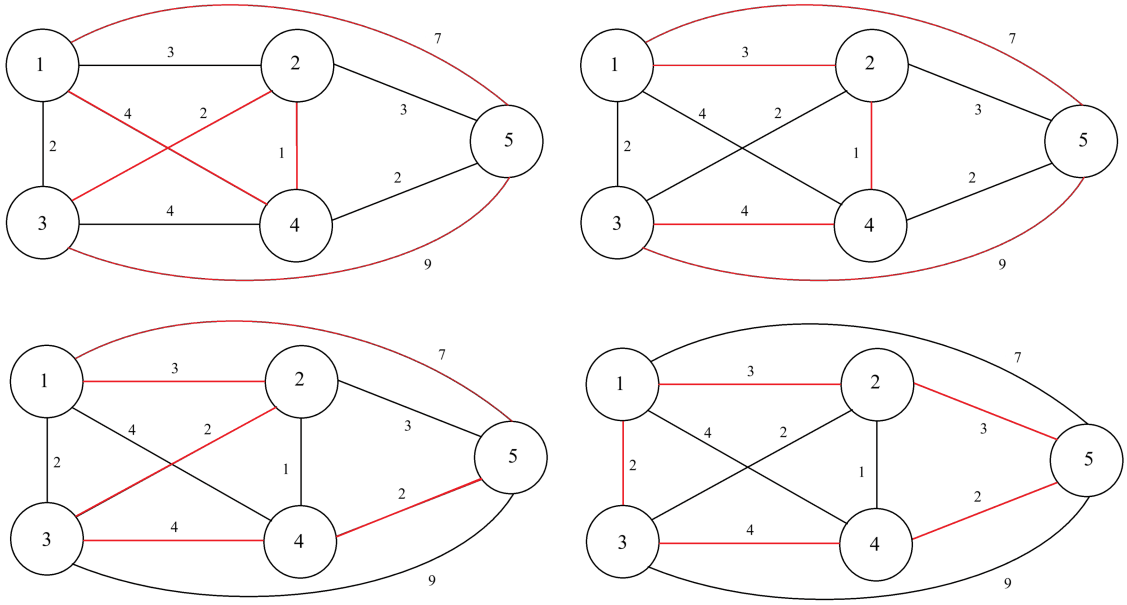
Τυπικά ο πολυπλοκότητα του είναι O(n!) γιατί μπορεί να χρειαστεί να ελέγξει όλα τα πιθανά ταξίδια αν ξεκινήσουμε από τη χειρότερη λύση και φτάσουμε μέχρι τη βέλτιστη. Στην πραγματικότητα όμως η στην πλειοψηφία των προβλημάτων η πολυπλοκότητα του αλγορίθμου αυτού είναι πιο κοντά στο n_2. Αυτό γιατί όπως φαίνεται δεν κάνει πολλές επαναλήψεις μέχρι να φτάσει στην τοπικά βέλτιστη κατάσταση. Έτσι λοιπόν είναι πολύ γρήγορος σε ταχύτητα αλλά είναι πολύ έχει το βασικό πρόβλημα όλων των αλγορίθμων αυτής της κατηγορίας ότι μπορεί να πέσει σε μία κατάσταση που την ονομάζουμε Local Optima από την οποία δεν μπορεί να βγει γιατί δεν υπάρχει περεταίρω βελτίωση αλλά ταυτόχρονα δεν είναι και η βέλτιστη λύση.

Ο ψευδοκώδικας του αλγορίθμου που υλοποιήθηκε εδώ είναι ο εξής. Αρχικά βρίσκουμε μία έτοιμη λύση η οποία εδώ είναι αυτή του πλησιέστερου γείτονα. Στη συνέχεια κάνουμε μία διπλή επανάληψη πάνω από όλα τα ζεύγη των πόλεων και εναλλάσσουμε τις θέσεις των δύο επιλεγμένων πόλεων. Αν το αποτέλεσμα είναι καλύτερο μετά από την αλλαγή, κρατάμε αυτήν την αλλαγή και ανανεώνουμε το ελάχιστο κόστος αλλιώς ξανακάνουμε την αλλαγή για να επιστρέψουμε στην προηγούμενη κατάσταση. Όλο αυτό γίνεται μέχρι να φτάσουμε σε σημείο που δεν υπάρχει βελτίωση. Πιο αναλυτικά στον Αλγόριθμο 4 που ακολουθεί.

```
Input: Graph G
Initialize: current_solution = Nearest_Neighbour(G);
repeat
    foreach city i of current_solution do
        foreach city j of current_solution do
            swap(i,j);
            if new_cost < old_cost) then
                best_cost = current_cost;
            end
            else
                swap(i,j);
            end
        end
    end
until No improvement is made;
```

Αλγόριθμος 4: Hill Climbing

Ακολουθεί ένα παράδειγμα στο Σχήμα 3.4. Πάνω αριστερά φαίνεται η αρχική κατάσταση από την οποία ξεκινήσαμε η οποία ακολουθεί τη διαδρομή 1-5-3-2-4-1. Στο επόμενο γράφημα φαίνεται μια προσπάθεια να αλλάξουν οι πόλεις 2 με 4 αλλά όπως φαίνεται οδηγεί σε μία διαδρομή με μεγαλύτερο κόστος οπότε δεν γίνεται δεκτή η αλλαγή αυτή. Στη συνέχεια έχουμε την αλλαγή 2 με 5 η οποία καταλήγει σε καλύτερη διαδρομή με κόστος 20 από κόστος 24 οπότε αυτή γίνεται δεκτή. Τέλος, έχουμε ακόμα μία αλλαγή που είναι η πόλη 3 με την 5 που και αυτό οδηγεί σε βελτίωση από 20 που είχαμε πριν σε 16 η οποία συμβαίνει εδώ να είναι και η βέλτιστη πράγμα το οποίο δεν θα συμβαίνει πάντα.



Σχήμα 3.4: Παραδείγμα Hill Climbing

3.5 2-opt

Ο επόμενος αλγόριθμος ανήκει και αυτός στην κατηγορία των αλγορίθμων επαναληπτικής βελτιστοποίησης όπως και ο προηγούμενος. Παρουσιάστηκε από τον Cores το 1958 αλλά είχε προταθεί πιο παλιά το 1956 από τον Flood. Αυτός είναι ο 2-opt, ένας από τους πιο γνωστούς αλγόριθμους για την επίλυση του προβλήματος τους πλανόδιου πωλητή. Η λογική του βασίζεται στο να ξεμπλέξει «κόμπους» που έχουν δημιουργηθεί από τον αλγόριθμο που βρήκε τη λύση. Αυτό το κάνει παίρνοντας μια διαδρομή που διασχίζει τον εαυτό της και την αλλάζει έτσι ώστε να μην συμβαίνει αυτό. Επίσης, εδώ χρησιμοποιείτε συνήθως ο αλγόριθμος πλησιέστερου γείτονα για την αρχική λύση όπως και έγινε και σε αυτήν την εργασία.

Η πολυπλοκότητα του είναι όπως και στην προηγούμενη περίπτωση O(n!) που στην πρώτη όψη είναι κακό αλλά πάλι είναι πιο κοντά στο n_2. Αυτό συμβαίνει γιατί οι περισσότερες πιθανές αλλαγές θα γίνουν στην πρώτη επανάληψη και είναι πολύ σπάνιο να προκύψουν καινούριες αλλαγές αλλά όχι απίθανο. Επιπλέον, εξακολουθεί να έχει το πρόβλημα του ότι μπορεί να βρεθεί σε ένα Local Optima όπως αναφέρθηκε πιο πάνω.

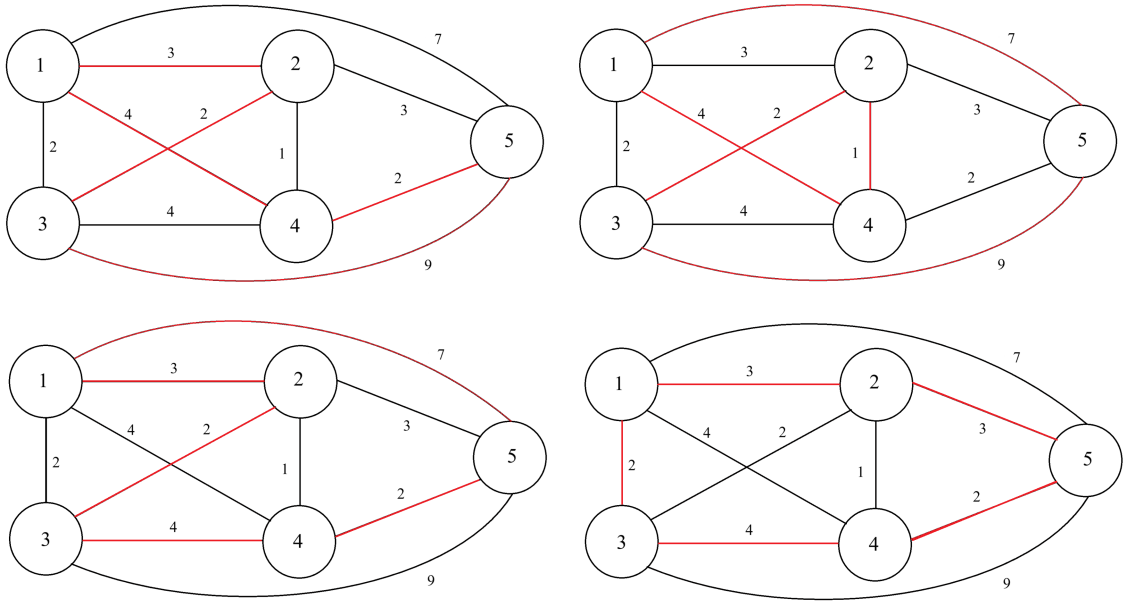
Ο Αλγόριθμος που ακολουθήθηκε στην εργασία είναι αρκετά απλός και είναι ο εξής. Αρχικά βρίσκουμε την αρχική λύση από τον αλγόριθμο του πλησιέστερου γείτονα. Στη συνέχεια για κάθε ζευγάρι από πόλεις i και j στη διαδρομή που έχουμε, δημιουργούμε μία νέα διαδρομή που είναι από την αρχή μέχρι την πόλη i ίδια, από την πόλη i μέχρι την πόλη j αντεστραμμένη και το υπόλοιπο από j μέχρι τέλος ίδια. Αν αυτή έχει μικρότερο κόστος από την προηγούμενη την κρατάμε σαν την καλύτερη λύση που είναι. Αυτό επαναλαμβάνεται μέχρι να μην γίνεται κάποια περεταίρω βελτίωση. Πιο παραστατικά φαίνεται στην παρακάτω ο Αλγόριθμος 5 που χρησιμοποιήθηκε [17].

Input: Graph G
Initialize: $current\_solution$ = Nearest_Neighbour(G);
repeat
  $improvement =$ false;
  for *each pair of edges* (i, j) *and* (k, l) *where* i, j, k, l *are distinct cities* **do**
    if $d(i, j) + d(k, l) > d(i, k) + d(j, l)$ **then**
      Reverse the segment between j and k in the tour;
      $improvement = true$;
    end
  end
until *No improvement is made*;
return *Optimized tour*;

Αλγόριθμος 5: 2-opt

Στη συνέχεια θα ακολουθήσει όπως συνηθίζεται ένα παράδειγμα στον κλασσικό γράφο που γίνεται μέχρι τώρα. Αρχικά έστω ότι η αρχική λύση που πήραμε είναι η 1-2-3-5-4-1 όπως φαίνεται και στο πάνω αριστερά μέρος του Σχήματος 3.5. Μία από τις πιθανές δοκιμές που θα κάνει που δεν θα φέρει καλύτερο αποτέλεσμα είναι να αλλάξει την πόλη 2 με την πόλη 5. Όταν γίνει ο έλεγχος αυτός θα φανεί ότι η διαδρομή γίνεται από 20 23 οπότε δεν θα γίνει δεκτή η αλλαγή. Εδώ αξίζει να σημειωθεί πως όταν οι δυο πόλεις που επιλέγονται είναι η μία δίπλα στην άλλη στη διαδρομή τότε αυτή αποτελεί μια κίνηση που θα έκανε ο προηγούμενος αλγόριθμος

που επεξηγήθηκε, ο hill climbing. Μία άλλη δοκιμή όπως φαίνεται στο κάτω αριστερά μέρος την απεικόνισης είναι να επιλεγούν οι πόλεις 3 και 5. Όταν γίνει αυτή η επιλογή η διαδρομή θα αλλάξει σε 1-3-4-5-2-1 και έτσι το κόστος της θα μειωθεί σε 18. Τέλος, μια ακόμα θετική αλλαγή που μπορεί να κάνει είναι να επιλέξει τις πόλεις 3 και 5 οπότε να αλλάξουν τη διαδρομή σε 1-2-5-4-3-1 που έχει βάρος 14 και τυχαίνει πάλι να είναι η βέλτιστη διαδρομή γιατί το παράδειγμα είναι μικρό και δεν έχει πολλές πιθανές λύσεις.



Σχήμα 3.5: Παραδείγμα 2-opt

3.6 Ant Colony Optimization

Ο επόμενος αλγόριθμος αλλάζει εντελώς τη λογική των αλγορίθμων που είχαμε μέχρι τώρα και ανήκει στην κατηγορία των μεθευρετικών. Ονομάζεται βελτιστοποίηση αποικίας μυρμηγκιών (Ant Colony Optimization). Είναι σίγουρο σαν αλγόριθμος ότι θα βρει λύση άλλα δεν υπάρχει κάποια εγγύηση για την ποιότητα της οπότε μπορεί να απέχει πολύ από τη βέλτιστη. Η βασική λογική του αλγορίθμου προέρχεται από τον τρόπο που τα μυρμήγκια μίας φωλιάς προσπαθούν να βρουν τροφή. Μιμείται τη λογική τους για να βρει τη βέλτιστη διαδρομή για να πάνε στο φαΐ. Αυτό το

καταφέρνουν τα μυρμήγκια με το να εκκρίνουν ορμόνες στις διαδρομές που παίρνουν για να πάνε στην τροφή. Όσο περισσότερες ορμόνες έχει μία διαδρομή τόσο πιο πιθανό είναι να επιλεχθεί από το επόμενο μυρμήγκι σαν επιλογή όταν ξεκινήσει από τη φωλιά. Επιπλέον, όσο πιο κοντά είναι μία διαδρομή τόσο περισσότερες φερομόνες θα εκκρίνουν τα μυρμήγκια οπότε να είναι πιο πιθανό να επιλεγεί από τα επόμενα [19].

Η πολυπλοκότητα του αλγορίθμου είναι $O(arn^2)$ όπου a ο αριθμός των μυρμηγκιών στην προσομοίωση που θα γίνει, r ο αριθμός των επαναλήψεων που θα γίνει και n ο αριθμός των πόλεων. Βασικό πλεονέκτημα αυτού αλλά και γενικά των αλγορίθμων που ανήκουν σε αυτήν την κατηγορία είναι ότι μπορούμε να ρυθμίσουμε με τις παραμέτρους (παράδειγμα τις επαναλήψεις, τον αριθμό των μυρμηγκιών, το πόσο φερομόνη θα εκκρίνουν και λοιπά) την αναλογία ανάμεσα στην ποιότητα των λύσεων και στον χρόνο εκτέλεσης. Οπότε ανάλογα με το πρόβλημα και τη χρήση για την οποία χρειάζεται θα υπάρχει η δυνατότητα αυτή.

Ο ψευδοκώδικας του Αλγορίθμου 6 που ακολουθεί έχει την εξής λογική. Αρχικά υπάρχει μία επανάληψη που θα τρέξει για όσες φορές χρειάζεται. Αυτή είναι παράμετρος που ρυθμίζεται ανάλογα με το κάθε πρόβλημα. Μέσα σε αυτήν υπάρχει μία δεύτερη επανάληψη που τρέχει για αριθμό όσο με τα μυρμήγκια που υπάρχουν. Η μεταβλητή του αριθμού των μυρμηγκιών είναι και αυτή παράμετρος που ρυθμίζεται πριν την έναρξη του. Στη συνέχεια γίνεται δημιουργία των διαδρομών που θα κάνει κάθε μυρμήγκι και έκκριση των φερομόνων ανάλογα με το πόσο καλή είναι η διαδρομή που έχει κάνει. Τέλος, έξω από την επανάληψη των μυρμηγκιών αλλά μέσα στην επανάληψη την εξωτερική γίνεται εξάτμιση των φερομόνων.

Input: Graph G

for *number of iterations* **do**

  for *number of ants* **do**

    Generate Ant Path based on pheromones;

    Update pheromone trails based on the tour length;

  end

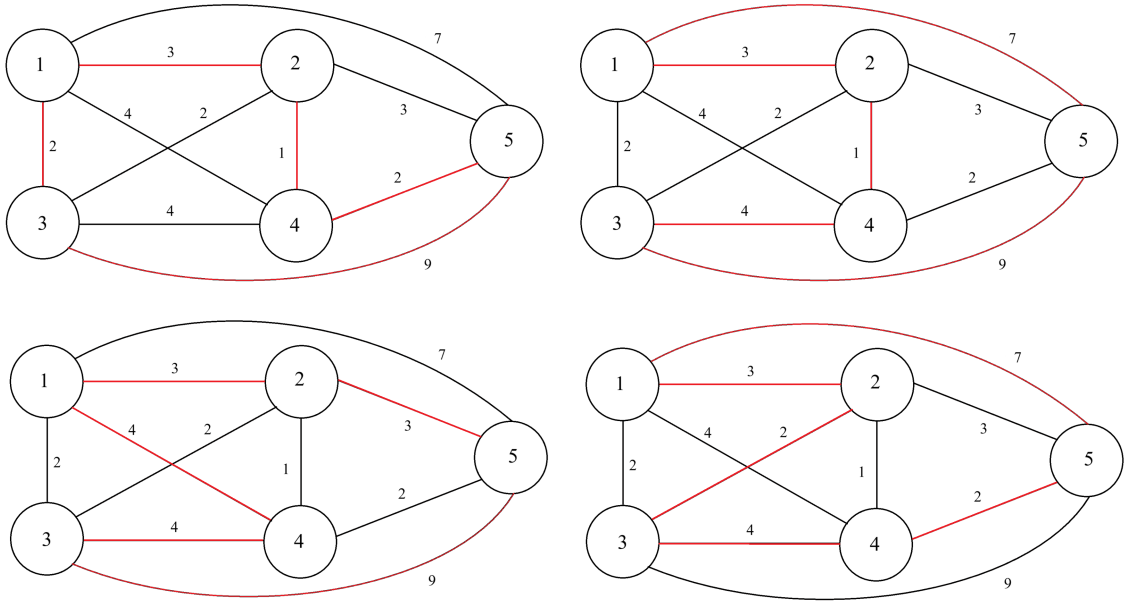
  Evaporate pheromone trails;

end

Select the best tour found;

Αλγόριθμος 6: Ant Colony Optimization

Ας δούμε τώρα λοιπόν ένα παράδειγμα με δύο μυρμήγκια από ένα τμήμα εκτέλεσης του αλγορίθμου. Έστω λοιπόν ότι οι διαδρομές που έκαναν τα δύο μυρμήγκια είναι οι 1-2-4-5-3-1 και 1-2-4-3-5-1 αντίστοιχα όπως φαίνεται και στο πάνω μέρος του σχήματος 3.6. Τώρα αυτές οι διαδρομές έχουν βάρος 15 και 24 αντίστοιχα. Άρα αφού το πρώτο μυρμήγκι έχει βρει καλύτερη και συντομότερη διαδρομή θα εκκρίνει περισσότερες φερομόνες στη διαδρομή οπότε στην επόμενη επανάληψη να υπάρχει μεγαλύτερη πιθανότητα να χρησιμοποιηθούν οι ακμές αυτές. Έτσι και γίνεται λοιπόν και οι ακμές που είναι στη διαδρομή του πρώτου μυρμηγκιού έχουν μεγαλύτερες τιμές με φερομόνες από αυτές που έχουν οι ακμές του δεύτερου αλλά οι κοινές έχουν ακόμα παραπάνω. Εδώ είναι το σημείο που εξατμίζονται οι ορμόνες που έχουν παραχθεί μέχρι τώρα κατά ένα ποσοστό. Στην επόμενη επανάληψη θα δημιουργηθούν παράδειγμα 2 καινούριες διαφορετικές, άλλα όχι απαραίτητα, διαδρομές. Αυτές θα είναι παράδειγμα οι 1-2-5-3-4-1 με βάρος 23 και οι 1-2-3-4-5-1 με βάρος 16 όπως φαίνεται στο κάτω μέρος του σχήματος. Ξανά λοιπόν Η διαδικασία θα συνεχίσει μέχρι τις επαναλήψεις που έχουν οριστεί. Τέλος, θα επιστραφεί η καλύτερη διαδρομή από όλες που έχουν παραχθεί.



Σχήμα 3.6: Παραδείγμα Ant Colony Optimization(ACO)

3.7 Gurobi

Ο τελευταίος τρόπος επίλυσης που μελετήθηκε δεν είναι ακριβώς αλγόριθμος αλλά είναι το λογισμικό Gurobi. Αυτό είναι ένα λογισμικό που δημιουργήθηκε από τους Dr. Zonghao Gu, Dr. Edward Rothberg, και Dr. Robert Bixby το 2008. Σκοπός αυτού του λογισμικού είναι να αποδίδει λύσεις σε προβλήματα γραμμικού προγραμματισμού. Η μοντελοποίηση που ακολουθήθηκε είναι η παρακάτω. Αρχικά δημιουργήθηκαν οι μεταβλητές του προβλήματος οι οποίες αν το πρόβλημα έχει n πόλεις θα είναι ένας πίνακας n x n όπου. Στον πίνακα θα έχει την τιμή 1 η θέση i,j αν και μόνο αν η διαδρομή περιέχει την ακμή που οδηγεί από το i στο j. Επιπλέον, έχουμε τις μεταβλητές αφαίρεσης των υποκυκλωμάτων MTZ. Όπως αναφέρθηκε και στο κομμάτι της βιβλιογραφικής ανασκόπησης είναι n τιμές και σκοπός τους είναι να κρατάνε τη θέση της κάθε πόλης στο ταξίδι ώστε να υπάρχει η βεβαιότητα ότι η λύση είναι μια συνεχόμενη διαδρομή και δεν είναι πολλές μικρότερες.

Στη συνέχεια προσθέτουμε περιορισμούς για το πρόβλημα. Ο περιορισμοί είναι τρεις βασικοί για το πρόβλημα το πλανόδιου πωλητή με τη μοντελοποίηση που έγινε. Οι πρώτοι δύο αφορούν το ότι κάθε πόλη πρέπει να επισκεφθεί μία φορά.

Με άλλα λόγια να έχει μόνο μια ακμή που εισέρχεται σε αυτήν και μία μόνο που εξέρχεται στην τελική λύση. Τέλος, έχουμε και τον περιορισμό MTZ που δίνει τις τιμές στις n μεταβλητές που δημιουργήθηκαν.

Πέρα από όλα αυτά υπάρχει και η αντικειμενική συνάρτηση που σε αυτό το πρόβλημα είναι η συνάρτηση που δίνει την τελική τιμή και πρέπει να έχει την ελάχιστή πιθανή τιμή. Αυτή είναι λοιπόν το άθροισμα όλων των γινόμενων κάθε μεταβλητής από τις n x n με το αντίστοιχο κόστος της ακμής εκείνης. Δηλαδή πιο απλά άμα η ακμή περιέχετε στο ταξίδι δηλαδή η τιμή της είναι 1 τότε προστίθεται το αντίστοιχο κόστος αυτής της ακμής. Αλλιώς, αν είναι 0, δεν προστίθεται τίποτα [20].

Το μαθηματικό μοντέλο μπορεί λοιπόν να μοντελοποιηθεί ως εξής [21]:

$$min \sum_{i \in V} \sum_{j \in V j \neq i} c_{\mathrm{ij}} x_{\mathrm{ij}} \tag{3.1}$$

$$\sum_{j \in V, j \neq i} x_{\mathrm{ij}} = 1, \forall i \in V \tag{3.2}$$

$$\sum_{i \in V, i \neq j} x_{\mathrm{ij}} = 1, \forall j \in V \tag{3.3}$$

$$u_{\mathrm{i}} - u_{\mathrm{j}} + n x_{\mathrm{ij}} \leq n - 1 \qquad \forall i, j \in V, i \neq j, i, j \neq 1 \tag{3.4}$$

$$x_{\mathrm{ij}} \in \{0, 1\}, \qquad \forall i, j \in V \tag{3.5}$$

$$u_1 = 1, 2 \leq u_{\mathrm{i}} \leq n, \qquad \forall i \in V, i \neq 1 \tag{3.6}$$

Σημαντικό είναι επίσης ότι το λογισμικό θα ελέγξει όλον τον χώρο αναζήτησης. Οπότε άμα του δοθεί επαρκής χρόνος θα μπορέσει πάντα να καταλήξει στη βέλτιστη λύση. Αυτό όμως τον κάνει πολύ αργό σε προβλήματα μεγάλου μεγέθους πόλεων.

Κεφάλαιο 4

Υπολογιστικά Αποτελέσματα

4.1 Προβλήματα που Μελετήθηκαν

Τα προβλήματα που μελετήθηκαν ανήκουν σε δύο ξεχωριστές κατηγορίες. Η πρώτη είναι 5 προβλήματα που δημιουργήθηκαν με γεννήτρια τυχαίων αριθμών που έχουν συντεταγμένες τυχαία από 0 μέχρι 100. Αυτά έχουν πολύ μικρό μέγεθος και ο λόγος που δημιουργήθηκαν είναι κυρίως για τον αλγόριθμο Brute Force. Αυτά τα προβλήματα έχουν μέγεθος 5, 8, 10, 12 και 15 πόλεων. Ακόμα και για αυτά τα προβλήματα ο αλγόριθμος Brute Force είναι αρκετά χρονοβόρος. Στο μεγαλύτερο από αυτά είναι με 15 πόλεις που σημαίνει ότι θα πρέπει να ελεγχθούν 14! = 87,178,291,200 πιθανοί συνδυασμοί που στη συνέχεια θα φανεί ότι δεν μπορούν να ελεγχθούν σε λογικό χρονικό διάστημα. Τα ονόματα από τα προβλήματα αυτά φαίνονται στον Πίνακα 4.1.

Πίνακας 4.1: Προβλήματα Γεννήτριας Τυχαίων Αριθμών

rng5	rng8	rng10	rng12	rng15

Πέρα από αυτά τα προβλήματα με τη γεννήτρια τυχαίων αριθμών, μελετήθηκαν και προβλήματα από τη βιβλιοθήκη TSPLIB [22]. Αυτή δημιουργήθηκε το 1991 από τον Gerhard Reinelt. Είναι μια βιβλιοθήκη από προβλήματα κυρίως του πλανόδιου πωλητή αλλά και άλλων παρόμοιων προβλημάτων όπως το πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem) ή η εύρεση ενός Χαμιλτόνιου κύκλου σε έναν γράφο. Συγκεκριμένα για το πρόβλημα του πλανόδιου πωλητή έχει προβλήματα από μέγεθος 51 κόμβων το eil51 μέχρι 85900 πόλεις το pla85900 που είναι και το μεγαλύτερο πρόβλημα που έχει λυθεί βέλτιστα ακόμα και σήμερα. Το μικρότερο

πρόβλημα λοιπόν έχει 50! = 3.04140932e64 πιθανούς συνδυασμούς οπότε ο αλγόριθμος Brute Force θα πάρει κάτι αιώνες για να τελειώσει μέχρι να τους ελέγξει όλους. Τα προβλήματα στα οποία πάνω έτρεξαν οι αλγόριθμοι έχουν μέγεθος από 51 που είναι το eil51 μέχρι και το rl5934 που έχει μέγεθος 5934 όπως και φαίνεται στο όνομα του. Η πλήρης λίστα είναι στον Πίνακα 4.2 [22].

Πίνακας 4.2: Προβλήματα Βιβλιοθήκης TSPLIB

eil51	berlin52	pr76	rat99	bier127
ch130	tsp225	a280	pr439	rat575
p654	rat783	pr1002	pcb1173	u1432
rl1889	pr2392	pcb3038	fnl4461	rl5934

Σε κάθε ένα από αυτά τα προβλήματα υπολογίστηκε η τιμή του κόστους που προκύπτει για κάθε μονοπάτι, δηλαδή το άθροισμα των μεταβάσεων ανάμεσα σε όλους τους κόμβους κάθε μονοματιού, η διαδρομή, δηλαδή η σειρά με την οποία επισκέφτηκε κάθε κόμβο η κάθε διαδρομή και τέλος ο χρόνος εκτέλεσης του κάθε αλγορίθμου. Οπού το πρόβλημα ήταν πολύ μεγάλο για τις ικανότητες του κάθε αλγορίθμου δόθηκε ένα άνω περιθώριο 5 λεπτών (300 δευτερολέπτων) σε κάθε αλγόριθμο. Όπου ξεπερνούσε αυτόν τον χρόνο τερματιζόταν ο αλγόριθμος και σαν τελική τιμή πάρθηκε η καλύτερη που είχε παραχθεί μέχρι εκείνη στη στιγμή. Αν δεν είχε τίποτα τότε απλά θεωρήθηκε ότι δεν βρήκε λύση μέσα στο χρονικό περιθώριο που το δινόταν. Σε όλα αυτά τα προβλήματα μελετήθηκαν οι αλγόριθμοι υλοποιήθηκαν στην αντικειμενοστραφή γλώσσα C++ στο περιβάλλον της Microsoft Visual Studio Code(VSCode) και έτρεξαν όλα σε Single Thread, δηλαδή κανένα από αυτά δεν ήταν παράλληλο, σε υπολογιστή με χαρακτηριστικά που φαίνονται στον Πίνακα 4.3.

Πίνακας 4.3: Χαρακτηριστικά Υπολογιστή για τις Προσομοιώσεις

Επεξεργαστής	Ryzen 5 5600 4.4 GHz
RAM	32GB 3000 MHz DDR4
SSD	Samsung 970 M.2 256GB
Καρτα Γραφικών	RTX 3060 Ti

4.2 Αποτελέσματα για τα Προβλήματα της Γεννήτριας Τυχαίων Αριθμών

rng5

Αυτό είναι το πρώτο πρόβλημα που μελετήθηκε στην εργασία αλλά παράλληλα είναι και το μικρότερο όλων. Έχει μέγεθος μόνο 5 κόμβους οπότε ακόμα και ο αλγόριθμος Brute Force, που είναι παραγοντικής πολυπλοκότητας, έχει να υπολογίσει μόνο (5-1)! δηλαδή 24 μονοπάτια για να βρει τη βέλτιστη λύση. Τέλος, λόγω του μικρού του μεγέθους παρατηρείτε ότι όλοι οι αλγόριθμοι βρήκαν τη βέλτιστη λύση η οποία έχει απόσταση ίση με 134.17.

Πίνακας 4.4: Πρόβλημα rng5

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
Brute Force	134.17	1 2 5 4 3 1	0 msec
NN	134.17	1 2 5 4 3 1	0 msec
Hill climbing	134.17	1 2 5 4 3 1	0 msec
2-opt	134.17	1 2 5 4 3 1	0 msec
Christofides	134.17	1 2 5 4 3 1	0 msec
ACO	134.17	1 2 5 4 3 1	2 msec
Gurobi	134.17	1 2 5 4 3 1	9 msec

rng8

Το επόμενο πρόβλημα που μελετήθηκε είναι και αυτό από τη γεννήτρια τυχαίων αριθμών με τη μόνη διαφορά από το προηγούμενο ότι έχουν προστεθεί ακόμα 3 πόλεις ώστε σαν σύνολο να γίνουν 8. Συνεπώς όλοι οι πιθανοί συνδυασμοί είναι 7! που είναι ίσο με 5040. Ήδη είναι φανερό λοιπόν ακόμα και από τόσο μικρό μέγεθος πόσο αυξήθηκε η κλίμακα και η δυσκολία του προβλήματος και κανένας αλγόριθμος δεν βρήκε τη βέλτιστη λύση. Σε αυτό το πρόβλημα λοιπόν η βέλτιστη λύση είναι ίση με 141.365.

Πίνακας 4.5: Πρόβλημα rng8

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος
Brute Force	141.365	1 2 6 8 5 4 3 7 1	0 msec
NN	146.088	1 7 8 6 2 5 4 3 1	0 msec
Hill climbing	143.333	1 7 2 6 8 5 4 3 1	0 msec
2-opt	143.333	1 7 2 6 8 5 4 3 1	0 msec
Christofides	146.088	1 7 8 6 2 5 4 3 1	0 msec
ACO	146.088	1 7 8 6 2 5 4 3 1	8 msec
Gurobi	141.365	1 7 3 4 5 8 6 2 1	11 msec

rng10

Το επόμενο πρόβλημα που μελετήθηκε είναι και αυτό από τη γεννήτρια τυχαίων αριθμών με τη μόνη διαφορά από το προηγούμενο ότι έχουν προστεθεί ακόμα 3 πόλεις ώστε σαν σύνολο να γίνουν 8. Συνεπώς όλοι οι πιθανοί συνδυασμοί είναι 7! που είναι ίσο με 5,040. Ήδη είναι φανερό λοιπόν ακόμα και από τόσο μικρό μέγεθος πόσο αυξήθηκε η κλίμακα και η δυσκολία του προβλήματος και κανένας αλγόριθμος δεν βρήκε τη βέλτιστη λύση. Σε αυτό το πρόβλημα λοιπόν η βέλτιστη λύση είναι ίση με 141.365.

Πίνακας 4.6: Πρόβλημα rng10

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος
Brute Force	143.142	1 2 6 8 5 4 3 7 10 9 1	66 msec
NN	147.866	1 9 10 7 8 6 2 5 4 3 1	0 msec
Hill climbing	145.11	1 9 10 7 2 6 8 5 4 3 1	0 msec
2-opt	145.11	1 9 10 7 2 6 8 5 4 3 1	0 msec
Christofides	144.442	1 9 3 4 5 8 6 2 7 10 1	0 msec
ACO	147.866	1 9 10 7 8 6 2 5 4 3 1	12 msec
Gurobi	143.142	1 2 6 8 5 4 3 7 10 9 1	131 msec

rng12

Το επόμενο πρόβλημα έχει μέγεθος 12 κόμβων. Ήδη από αυτό ακόμα το μέγεθος βλέπουμε ότι οι αλγόριθμοι που δεν είναι ακριβείς δεν μπορούν να βρουν τη βέλτιστη λύση. Επιπλέον, ο Brute Force έχει αρχίσει ήδη να παίρνει έναν αρκετά σημαντικό χρόνο για να εκτελεστεί, 8 δευτερολέπτων. Τέλος, η βέλτιστη λύση του έχει κόστος 156.148.

Πίνακας 4.7: Πρόβλημα rng12

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
Brute Force	156.148	1 9 12 11 3 4 5 8 6 2 7 10 1	8251 msec
NN	169.291	1 9 10 12 11 7 8 6 2 5 4 3 1	0 msec
Hill climbing	162.23	1 9 12 10 11 7 2 6 8 5 4 3 1	0 msec
2-opt	162.23	1 9 12 10 11 7 2 6 8 5 4 3 1	0 msec
Christofides	157.01	1 9 12 3 4 5 8 6 2 7 11 10 1	0 msec
ACO	159.222	1 9 10 7 11 3 4 5 8 6 2 12 1	19 msec
Gurobi	156.148	1 9 12 11 3 4 5 8 6 2 7 10 1	132 msec

rng15

Το επόμενο και τελευταίο πρόβλημα από τη γεννήτρια τυχαίων αριθμών έχει μέγεθος ίσο με 15 και η βέλτιστη λύση σε αυτό είναι 201.355. Εδώ λοιπόν αρχίζει ο αλγόριθμος Brute Force να μην τρέχει. Έχει να εξερευνήσει 14!=87,178,291,200 διαφορετικά μονοπάτια για να βρει λύση, το οποίο τον καθιστά ιδιαίτερα χρονοβόρο και δεν μπορεί να τερματίσει στο χρονικό περιθώριο που του δόθηκε.

Πίνακας 4.8: Πρόβλημα rng15

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
Brute Force	-	-	Time-Out
NN	221.712	1 9 10 13 12 11 7 8 6 2 5 4 3 14 15 1	0 msec
Hill climbing	210.645	1 10 9 12 13 11 7 2 6 8 5 4 3 14 15 1	0 msec
2-opt	210.645	1 10 9 12 13 11 7 2 6 8 5 4 3 14 15 1	0 msec
Christofides	203.43	1 9 12 15 14 3 4 5 8 6 2 7 11 13 10 1	0 msec
ACO	220.688	1 9 10 13 12 14 15 3 4 5 2 6 8 7 11 1	26 msec
Gurobi	201.355	1 10 11 7 2 6 8 5 4 3 14 15 13 12 9 1	169 msec

Συμπερασματικά λοιπόν βλέπουμε ότι οι περισσότεροι αλγόριθμοι δεν έχουν πρόβλημα με το να βρούνε μία πολύ καλή λύση κοντά στη βέλτιστη και οι χρόνοι είναι ακόμα πολύ χαμηλοί για όλους, εκτός από τον Brute Force που δεν κατάφερε να τερματίσει στο τελευταίο πρόβλημα με τους 15 κόμβους μέσα στο χρονικό περιθώριο των 5 λεπτών που το δόθηκε.

4.3 Αποτελέσματα για τα Προβλήματα της Βιβλιοθήκης TSPLIB

Από τη βιβλιοθήκη TSPLIB πάρθηκαν 20 προβλήματα με διαφορετικά μεγέθη τα οποία ξεκινάνε από 51 κόμβους με το eli51 μέχρι 5,934 με το rl5934. Όλα τα προβλήματα δίνονται με την μορφή συντεταγμένων x,y οπότε όλα ανήκουν στην κατηγορία των ευκλείδειων. Επιπλέον δεν έτρεξε ο αλγόριθμος Brute Force καθώς δεν κατάφερε να τερματίσει ούτε στο πρόβλημα rng15 της προηγούμενης κατηγορίας. Τα πλήρη αποτελέσματα φαίνονται στο παράρτημα στο τέλος του κειμένου Α΄.

Ο αλγόριθμος Nearest Neighbor κατάφερε να τερματίσει σε όλα τα προβλήματα και με γρήγορες ταχύτητες, μέχρι 898 χιλιοστά του δευτερολέπτου, ακόμα και στο μεγαλύτερο πρόβλημα που του δόθηκε. Αυτό ωστόσο τον έρχεται σε αντίθεση με τα αποτελέσματα που έδωσε για την ποιότητα της λύσης. Ήταν από 14.76% μέχρι 41.88% μακριά από την βέλτιστη λύση με μέσο όρο 20.52% ο οποίος τον καθιστά τον χειρότερο αλγόριθμο σε ποιότητα λύσης.

Πίνακας 4.9: Αλγόριθμος Nearest Neighbor

Πρόβλημα	**Λύση**	**Χρόνος**
eil51	513.610	0 sec
berlin52	8,980.92	0 sec
pr76	153,462	0 sec
rat99	1,564.72	0 sec
bier127	135,752	0 sec
ch130	7,575.01	0 sec
tsp225	4,826.35	1 msec
a280	3,148.11	1 msec
pr439	131,282	5 msec
rat575	8,449.32	7 msec
p654	43,410.5	9 msec
rat783	11,255.1	16 msec
pr1002	315,597	26 msec
pcb1173	70,277.9	32 msec
u1432	188,815	46 msec
rl1889	400,685	85 msec
pr2392	461,207	137 msec
pcb3038	175,573	223 msec
fnl4461	227,158	544 msec
rl5934	683,806	898 msec

Στη συνέχεια έχουμε τον αλγόριθμο Hill Climbing ο οποίος θυσιάζει περισσότερο χρόνο από τον απλό Nearest Neighbor για να διορθώσει το βασικό του πρόβλημα που είναι η ποιότητα της λύσης. Κατάφερε να μειώσει τη μέση τιμή της ποιότητας της λύσης από 20.52% σε 16.99%. Για να γίνει όμως αυτό το αποτέλεσμα ο χρόνος εκτέλεσης αυξήθηκε μέχρι και 47 δευτερόλεπτα στο μεγαλύτερο πρόβλημα αύξηση επί 50 από τον απλό Nearest Neighbor.

Πίνακας 4.10: Αλγόριθμος Hill Climbing

Πρόβλημα	**Λύση**	**Χρόνος**
eil51	507,818	0 sec
berlin52	8,797.92	0 sec
pr76	150,417	0 sec
rat99	1,476.20	1 msec
bier127	129,644	4 msec
ch130	7,404.87	3 msec
tsp225	4,589.03	7 msec
a280	3,100.49	7 msec
pr439	127,920	26 msec
rat575	8,211.86	54 msec
p654	41,900.5	69 msec
rat783	10,825.2	205 msec
pr1002	305,436	235 msec
pcb1173	67,889.3	363 msec
u1432	180,956	627 msec
rl1889	393,536	1.9 sec
pr2392	448,229	1.4 sec
pcb3038	170,948	3.0 sec
fnl4461	220,001	9.3 sec
rl5934	669,774	47.5 sec

Παρόμοια με του Hill Climbing είναι και η λογική του 2-opt αλλά κατάφερε να αποφέρει πολύ καλύτερα αποτελέσματα. Μείωσε ακόμα περισσότερο τη λύση και πλησίασε σε ποσοστό 10.52% από τη βέλτιστη στα προβλήματα που μπόρεσε να τερματίσει και 16.48% αν προσθέσουμε και τα δύο τελευταία που δεν τερμάτισε. Έτσι βέβαια αυξήθηκε ακόμα παραπάνω ο χρόνος εκτέλεσης με αποτέλεσμα να μην μπορεί να τρέξει στα δύο τελευταία προβλήματα και το τελευταίο πρόβλημα που τερμάτισε ήταν το pcb3038 με χρόνο 299 δευτερόλεπτα.

Πίνακας 4.11: Αλγόριθμος 2-opt

Πρόβλημα	**Λύση**	**Χρόνος**
eil51	491,086	1 msec
berlin52	8,384.19	1 msec
pr76	140,038	4 msec
rat99	1,485.78	10 msec
bier127	132,02	16 msec
ch130	7,022.66	32 msec
tsp225	4,207.08	104 msec
a280	2,797.03	253 msec
pr439	117,611	971 msec
rat575	7,445.53	2.5 sec
p654	38,578.9	3.6 sec
rat783	9,919.04	7.200 sec
pr1002	278,846	13.1 sec
pcb1173	63,174.5	20.1 sec
u1432	169,486	37.4 sec
rl1889	362,013	71.7 sec
pr2392	412,534	178 sec
pcb3038	154,846	299 sec
fnl4461	-	300 sec
rl5934	-	300 sec

Ένας ακόμα αλγόριθμος που έδωσε πολύ καλά αποτελέσματα ήταν ο αλγόριθμος Christofides που κατάφερε και αυτός να τερματίσει σε όλα τα προβλήματα. Η μέση απόκλιση από τη βέλτιστη λύση του ήταν 13.49% και οι τιμές σε όλα τα προβλήματα κυμαίνονται από 10.94% μέχρι 20% που τον καθιστά τον δεύτερο καλύτερο αλγόριθμο σε ποιότητα λύσης μετά τον 2-opt αλλά κατάφερε να τερματίσει σε όλα τα προβλήματα με μόνο 78 δευτερόλεπτα στο μεγαλύτερο πρόβλημα που του δόθηκε.

Πίνακας 4.12: Αλγόριθμος Christofides

Πρόβλημα	**Λύση**	**Χρόνος**
eil51	538,417	0 sec
berlin52	8,481.08	0 sec
pr76	127,335	1 msec
rat99	1,330.15	3 msec
bier127	132,825	5 msec
ch130	7,333.77	5 msec
tsp225	4,698.38	19 msec
a280	3,054.42	33 msec
pr439	122,011	103 msec
rat575	7,922.89	249 msec
p654	41,213.5	361 msec
rat783	10,378.8	631 msec
pr1002	300,603	1.2 sec
pcb1173	67,139	1.7 sec
u1432	180,906	3.3 sec
rl1889	356,487	3.6 sec
pr2392	434,253	12.0 sec
pcb3038	160,127	23.8 sec
fnl4461	217,705	88.5 sec
rl5934	616,856	78.8 sec

Ο αλγόριθμος ACO είχε πολύ καλά αποτελέσματα στα αρχικά προβλήματα αλλά καθώς ανέβαινε το μέγεθος του προβλήματος άρχισαν να πέφτουν οι αποδόσεις του. Αυτό οφείλεται κυρίως όμως στις παραμέτρους που δόθηκαν καθώς άμα επαναληφθεί με διαφορετικές παραμέτρους ενδεχομένως να δώσει καλύτερα αποτελέσματα αλλά και με αυξημένο χρόνο εκτέλεσης. Η μέση λύση του απείχε 18.44% από τη βέλτιστη στα προβλήματα που τερμάτισε και 52.74% αν προσθέσουμε και αυτά που δεν τερμάτισε και το μεγαλύτερο πρόβλημα που κατάφερε να λύσει είναι το pr439 με χρόνο 147 δευτερόλεπτα.

Πίνακας 4.13: Αλγόριθμος ACO

Πρόβλημα	**Λύση**	**Χρόνος**
eil51	502,550	432 msec
berlin52	8,918.718	460 msec
pr76	131,95	1.2 sec
rat99	1,510.37	2.5 sec
bier127	141,483	5 sec
ch130	8,172.38	5.1 sec
tsp225	4,799.37	24.4 sec
a280	3,701.17	45.3 sec
pr439	146.947	171 sec
rat575	-	300 sec
p654	-	300 sec
rat783	-	300 sec
pr1002	-	300 sec
pcb1173	-	300 sec
u1432	-	300 sec
rl1889	-	300 sec
pr2392	-	300 sec
pcb3038	-	300 sec
fnl4461	-	300 sec
rl5934	-	300 sec

Τέλος, μελετήθηκε και η λύση με το λογισμικό Gurobi. Εδώ για τα πρώτα τέσσερα προβλήματα η λύση που βρήκε ήταν η βέλτιστη. Στη συνέχεια βρήκε λύση για τα τρία επόμενα μόνο προβλήματα με μέγεθος μέχρι 225 κόμβους. Αυτό τον καθιστά την καλύτερη επιλογή για μέγεθος μέχρι 200 περίπου κόμβους καθώς πάντα έβρισκε τη βέλτιστη λύση ή μια πολύ κοντά σε αυτή αλλά σε οτιδήποτε μεγαλύτερο δεν κατάφερε να τερματίσει.

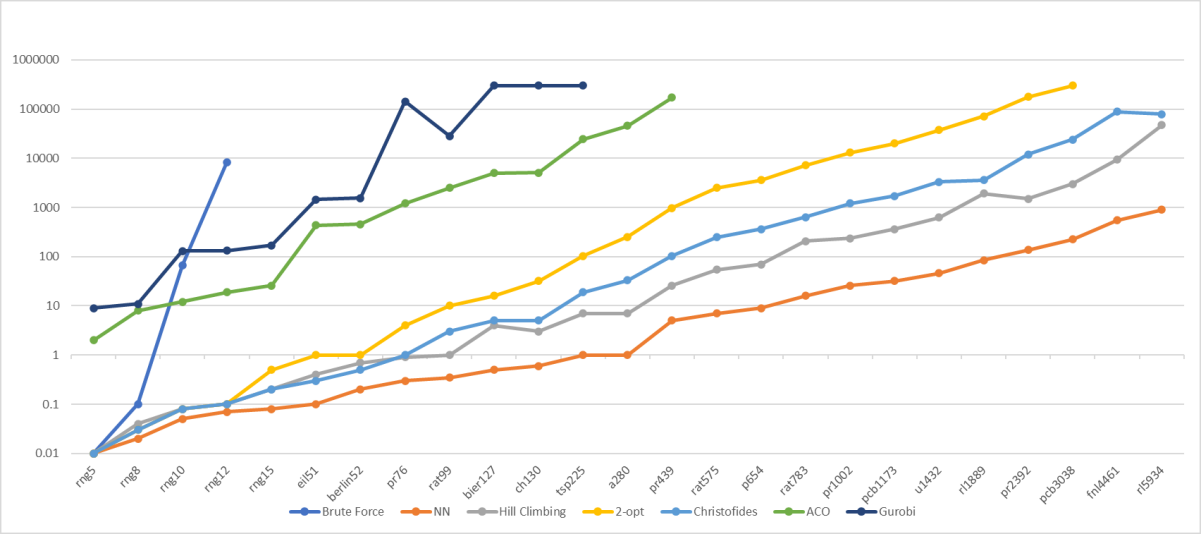
Πίνακας 4.14: Αλγόριθμος ACO

Πρόβλημα	**Λύση**	**Χρόνος**
eil51	428,872	1.4 sec
berlin52	7,544.37	1.5 sec
pr76	108,159	142 sec
rat99	1,219.24	28,1 sec
bier127	121,472	300 sec
ch130	6,310.81	300 sec
tsp225	4,409.42	300 sec
a280	-	300 sec
pr439	-	300 sec
rat575	-	300 sec
p654	-	300 sec
rat783	-	300 sec
pr1002	-	300 sec
pcb1173	-	300 sec
u1432	-	300 sec
rl1889	-	300 sec
pr2392	-	300 sec
pcb3038	-	300 sec
fnl4461	-	300 sec
rl5934	-	300 sec

4.4 Γραφικές Παραστάσεις Αποτελεσμάτων Χρόνου Εκτέλεσης και Ποιότητας Λύσης

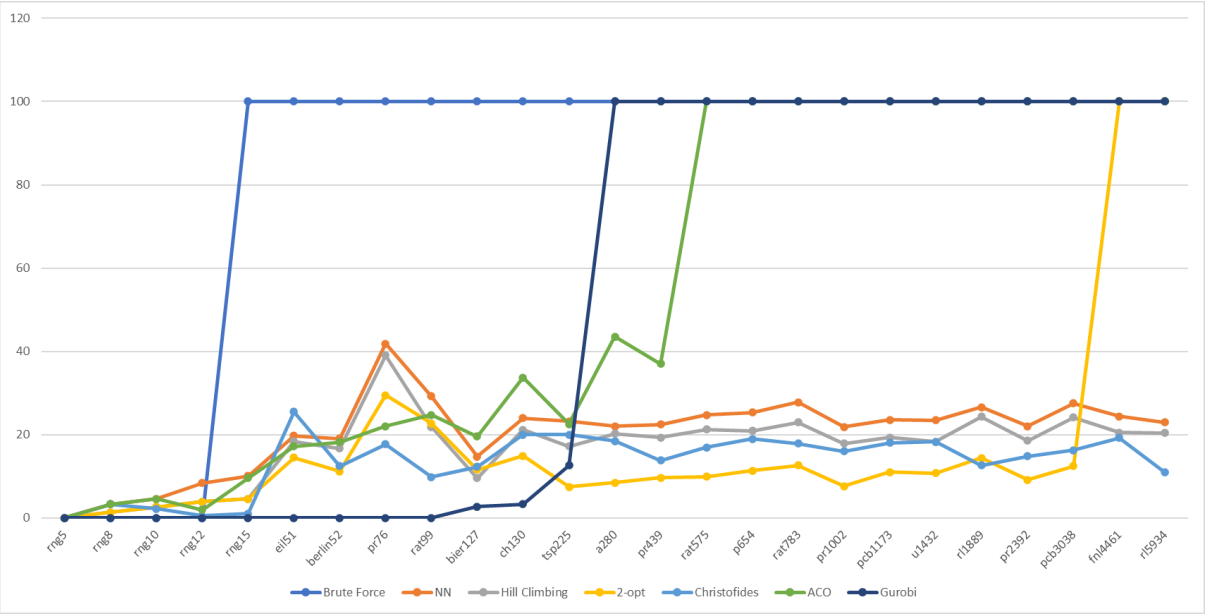
Μετά την ολοκλήρωση όλων των υπολογιστικών διαδικασιών και την παράθεση των αποτελεσμάτων των αλγορίθμων, ακολουθεί η παρουσίαση των τελικών γραφημάτων που αναφέρονται στους μέσους χρόνους εκτέλεσης για κάθε πρόβλημα που μελετήθηκε. Επίσης, παρουσιάζονται οι αποκλίσεις που προέκυψαν στις τιμές των λύσεων σε σύγκριση με τις τιμές των βέλτιστων λύσεων.

Το πρώτο γράφημα που ακολουθεί είναι στο Σχήμα 4.1 και είναι ο χρόνος που έκανε κάθε αλγόριθμος να τρέξει για κάθε πρόβλημα ξεχωριστά, σε λογαριθμική κλίμακα. Είναι φανερό ότι ο αλγόριθμος Brute Force ανέβηκε πολύ απότομα στον χρόνο εκτέλεσης λόγο της παραγοντικής πολυπλοκότητας του. Επίσης, φαίνεται ότι σαν γενικός κανόνας ισχύει ότι η σειρά στον χρόνο εκτέλεσης στους αλγορίθμους είναι η εξής: Brute Force > Gurobi > Ant Colony Optimization > 2-opt > Christofides > Hill Climbing > Nearest Neighbour. Αυτό ήταν όμως αναμενόμενο σε γενικές γραμμές από την πολυπλοκότητα του κάθε αλγορίθμου, αλλά δεν σημαίνει σε καμία περίπτωση πως η καλύτερη επιλογή είναι οι πιο γρήγοροι αλγόριθμοι όπως θα φανεί από τα επόμενα γραφήματα.



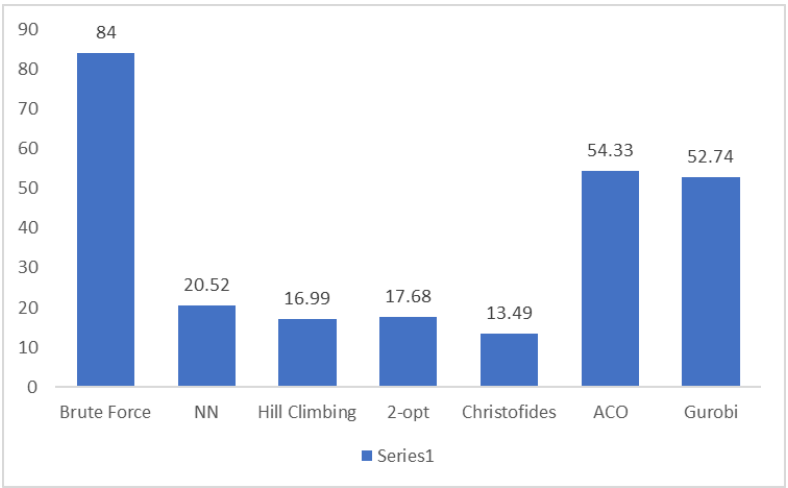
Σχήμα 4.1: Χρόνος Εκτέλεσης Αλγορίθμων

Στο επόμενο γράφημα λοιπόν στο Σχήμα 4.2 φαίνεται σε ποσοστιαία κλίμακα για κάθε αλγόριθμο το αποτέλεσμα που κατάφερε να πετύχει. Εδώ ο φαίνεται ότι το λογισμικό Gurobi μέχρι και μέγεθος περίπου 100 πόλεων καταφέρνει να βρει τη λύση στο 100% των γραφημάτων που τους δόθηκαν, το οποίο τον κάνει ιδανικό για μικρά προβλήματα. Σημαντικό είναι επίσης πως ο ACO ενώ έχει πολύ καλά αποτελέσματα αρχικά στη συνέχεια έπεσαν οι αποδόσεις του και αυτό συνέβη γιατί είναι πολύ δύσκολο να υπολογιστούν σωστά οι παράμετροί του για να αποδώσει καλύτερα. Παρόλα αυτά υπάρχει ακόμα περιθώριο στη βελτίωση του σε σχέση με άλλους αλγορίθμους. Τέλος, φαίνεται ότι ο αλγόριθμος Christofides που έχει την εγγύηση ότι η χειρότερη λύση του δεν είναι μεγαλύτερη από 3/2 της βέλτιστης δεν έχει ανέβει ποτέ πάνω από το 50%.



Σχήμα 4.2: Απόκληση Αλγορίθμων από τη Βέλτιστη Λύση

Το τελευταίο γράφημα είναι στο Σχήμα 4.3 και φανερώνει τη μέση ποσοστιαία απόσταση από τη βέλτιστη λύση. Αρχικά παρατηρείται ότι ο αλγόριθμος Brute Force έχει 0% απόσταση γιατί βρίσκει πάντα τη βέλτιστη λύση και ο Gurobi είναι πολύ κοντά γιατί όταν έφτασε το χρονικό περιθώριο που του δόθηκε σε κάποια προβλήματα κόπηκε η εκτέλεση και επέστρεψε την καλύτερη μέχρι στιγμής λύση, η οποία ήταν πολύ κοντά στη βέλτιστη άλλα όχι ίδια. Στη συνέχεια την καλύτερη απόδοση είχε ο 2-opt με 10.52% μόνο χειρότερη από τη βέλτιστη με τον αλγόριθμος του Christofides να ακολουθεί με 13.49%. Ο Hill climbing και ο 2-opt είναι σε κάθε πρόβλημα πιο αποδοτικοί από τον Nearest Neighbour γιατί χρησιμοποιούνε τη δική του λύση σαν αρχική και τη βελτιώνουν με κόστος χρόνο.



Σχήμα 4.3: Μέση Απόκληση Αλγορίθμων από τη Βέλτιστη Λύση

Κεφάλαιο 5

Συμπεράσματα

5.1 Γενικά Συμπεράσματα

Ολοκληρώνοντας λοιπόν σε αυτήν την εργασία υλοποιήθηκαν οι αλγόριθμοι Brute Force, Nearest Neighbour, Hill Climbing, 2-opt, Christofides, Ant Colony Optimization και το λογισμικό Gurobi. Οι αλγόριθμοι αυτοί έτρεξαν πάνω σε 20 προβλήματα από τη βιβλιοθήκη TSPLIB σε μεγέθη από 51 προβλήματα μέχρι 5,934 και σε 5 προβλήματα που δημιουργήθηκαν από γεννήτρια τυχαίων αριθμών με μεγέθη 5, 8, 10, 12 και 15 κόμβων. Πάνω σε αυτά τα προβλήματα υπολογίστηκε η καλύτερη λύση που βρήκε ο κάθε αλγόριθμος, η διαδρομή που έκανε για να φτάσει σε αυτήν τη λύση καθώς και ο χρόνος εκτέλεσης του κάθε προβλήματος για να φτάσει σε αυτήν τη λύση.

Συμπερασματικά, τα καλύτερα αποτελέσματα τα έδωσε ο αλγόριθμος 2-opt, με αρχική λύση αυτήν του Nearest Neighbour, με πολύ μικρούς χρόνους εκτέλεσης για τη βελτίωση που έκανε στη λύση. Κατάφερε να τρέξει σχεδόν σε όλα τα προβλήματα εκτός από τα δύο τελευταία. Η μέση λύση ήτανε κατά 10.52% χειρότερη από τη βέλτιστη με μόνο που στις περισσότερες καθημερινές περιπτώσεις είναι πολύ καλή λύση για το αποτέλεσμα.

Ο αλγόριθμος που ακολουθεί στην επίδοση είναι ο προσεγγιστικός του Christofides. Αυτός έχει απόδοση ίση με 13.49% μακριά από τη βέλτιστη και είναι και αυτός μέσα σε χρονικά περιθώρια που είναι αποδεκτά. Μόνο σε ελάχιστα προβλήματα βρήκε λύση καλύτερη από τον 2-opt αλλά ήταν πολύ πιο γρήγορος σε χρόνους εκτέλεσης γενικά.

Ένας ακόμα αλγόριθμος που μελετήθηκε είναι ο Hill Climbing, είχε παρόμοια

αποτελέσματα με τον 2-opt και αυτός χρησημοποίησε σαν αρχική λύση του Nearest Neighbour. Η κύρια διαφορά τους είναι ότι βρήκε χειρότερες λύσεις από τον 2-opt αλλά είχε πιο γρήγορους χρόνους εκτέλεσης. Η μέση λύση είναι 16.99% μακριά από τη βέλτιστη. Η επιλογή αυτού του αλγορίθμου είναι μια καλή και γρήγορη λύση για τη βελτίωση μίας υπάρχουσας λύσης.

Ο αλγόριθμος με τα χειρότερα αποτελέσματα σε λύση είναι ο Nearest Neighbour, αλλά ταυτόχρονα είχε με διαφορά και τις γρηγορότερες λύσεις λόγο της μικρής πολυπλοκότητας του. Αυτός ο αλγόριθμος είναι η καλύτερη επιλογή για μία γρήγορη λύση αλλά χωρίς να είναι πολύ καλά τα αποτελέσματα του, πλησιάζει το 20.52% μακριά από τη βέλτιστη λύση. Ο γρήγορος χρόνος εκτέλεσης του τον κάνει την καλύτερη επιλογή για μία γρήγορη λύση που μπορεί να βελτιωθεί από αλγορίθμους τοπικής αναζήτησης όπως και έγινε από αυτήν την εργασία μέσω των αλγορίθμων Hill Climbing και 2-opt.

Ο χειρότερος αλγόριθμος σε αποτελέσματα είναι ο Brute Force και κατάφερε να τρέξει μόνο σε προβλήματα μέχρι μέγεθος 12 και πάλι με πολύ μεγάλο χρόνο. Το βασικό του πλεονέκτημα είναι ότι χρησιμοποιείται για να βρει τη βέλτιστή λύση για να κρίνουμε πόσο καλά αποτελέσματα δίνουν οι άλλοι αλγόριθμοι που δεν είναι ακριβείς.

Τέλος, μελετήθηκε και η λύση με το πρόγραμμα Gurobi. Αυτή αποδείχθηκε ότι έδωσε πολύ καλά αποτελέσματα μέχρι μέγεθος 100 περίπου κόμβων και σχεδόν σε όλα τα προβλήματα που βρήκε λύση είναι η καλύτερη πιθανή εκτός από τα τελευταία προβλήματα που βρήκε λύση αλλά πολύ κοντά στη βέλτιστη λόγω του χρονικού περιθωρίου που το δόθηκε.

5.2 Μελλοντικές Εξελίξεις

Στην παρούσα διπλωματική μελετήθηκε ένα μικρό κομμάτι από τους αλγορίθμους που υπάρχουν και σε ένα μικρό κομμάτι από τα προβλήματα που υπάρχουν. Σαν εξέλιξη μπορούν να υλοποιηθούν επεκτάσεις των αλγορίθμων αυτών, όπως για παράδειγμα ο 3-opt που αποτελεί βελτίωση του 2-opt.

Επίσης, μπορούνε να μελετηθούν μεγαλύτερα προβλήματα που υπάρχουν στη βιβλιοθήκη TSPLIB που φτάνουν σε μέγεθος μέχρι και τους 59,000 κόμβους. Εδώ θα χρειαστούν πολλές βελτιώσεις στους αλγορίθμους και στις δομές δεδομένων

που χρησιμοποιήθηκαν, αλλά και στο υπολογιστικό σύστημα για να μπορέσουν να αποθηκευτούν όλες οι απαραίτητες μεταβλητές.

Τέλος, για τον αλγόριθμο Ant Colony Optimization μπορεί να γίνουν περεταίρω βελτιώσεις και να γίνουν καλύτερα τα αποτελέσματα με το να μελετηθούν περισσότερες τιμές για τις παραμέτρους του. Κάποιες από αυτές είναι ο αριθμός επαναλήψεων, ο αριθμός των μυρμηγκιών και ο βαθμός εξάτμισης της φερομόνης.

Βιβλιογραφία

[1] Papadimitriou Christos H, "The euclidean traveling salesman problem is np-complete," 1977.

[2] R. F. G. Dantzig and S. Johnson, "Solution of a large-scale traveling-salesman problem," 1954.

[3] G. Laporte, "A short history of the traveling salesman problem," 2005.

[4] M. Held and Richard M. Karp, "The traveling-salesman problem and minimum spanning trees: Part ii," 1971.

[5] L. P. M. Camerini and F. Maffioli, "On improving relaxation methods by modified gradient techniques," 1975.

[6] M. Grötschel, *Polyedrische Charakterisierungen kombinatorischer Optimierungsprobleme*. 1977.

[7] H. Crowder and M. W. Padberg, "Solving large-scale symmetric travelling salesman problems to optimality," 1980.

[8] M. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," 1987.

[9] M. Grötschel and O. Holland, "A cutting plane algorithm for minimum perfect 2-matchings," 1987.

[10] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," 1991.

[11] V. C. D. Applegate, R. Bixby and W. Cook, *The Traveling Salesman Problem*. 1994.

[12] V. C. D. Applegate, R. Bixby and W. Cook, "On the solution of traveling salesman problems," 1998.

[13] V. C. D. Applegate, R. Bixby and W. Cook, "Tsp cuts which do not conform to the template paradigm," 2001.

[14] V. C. W. C. D. E. M. G. D. Applegate, R. Bixby and K. Helsgau, "Certification of an optimal tsp tour through 85900 cities," 2009.

[15] A. C. E. Miller and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," 1960.

[16] Christofides N., "Worst-case analysis of a new heuristic for the travelling salesman problem," 1976.

[17] N. Christian, "Heuristics for the traveling salesman problem," 2003.

[18] Michael T. Goodrich and R. Tamassia, *Algorithm Design and Applications*, ch. 17, pp. 513–514. Wiley, 2015.

[19] D. Marco and Gambardella Luca Maria, "Ant colonies for the traveling salesman problem," 1997.

[20] F. Dantzig G. B. and Johnson S. M., "On a linear-programming, combinatorial approach to the traveling-salesman problem," 1959.

[21] M. Velednitsky, "Short combinatorial proof that the dfj polytope is contained in the mtz polytope for the asymmetric traveling salesman problem," 2018.

[22] G. Reinelt, "Tsplib—a traveling salesman problem library," 1991. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/` Last accessed 26 February 2024.

[23] Tucker A. W., "On directed graphs and integer programs," 1960.

[24] Lawler E. L., *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, 1985.

[25] van Bevern René and Slugina Viktoriia A., "A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem," 2020.

[26] Flood, M. M., "The traveling-salesman problem," 1956.

[27] B. Arash and M. Mohammad, "New efficient transformation of the generalized traveling salesman problem into traveling salesman problem," 2002.

[28] G. F. Rego César, Gamboa Dorabela and O. Colin, "Traveling salesman problem heuristics: leading methods, implementations and latest advances," 2011.

Παράρτημα Α΄

Αποτελέσματα Βιβλιοθήκης TSPLIB

eil51

Όλα τα παρακάτω προβλήματα που μελετήθηκαν πάρθηκαν από τη βιβλιοθήκη TSPLIB [22] και σε αυτά δεν συμπεριλαμβάνεται ο αλγόριθμος Brute Force καθώς είναι αδύνατο να βρει λύση σε τόσο μεγάλο μέγεθος. Το πρώτο από αυτά είναι το eil51 που είναι το πρόβλημα με το μικρότερο μέγεθος σε αυτήν τη βιβλιοθήκη από ευκλείδεια προβλήματα. Η βέλτιστη λύση είναι 426 σύμφωνα με το TSPLIB άλλα με στρογγυλοποίηση προς τα κάτω στις αποστάσεις. Εδώ χωρίς στρογγυλοποίηση η βέλτιστη λύση είναι 428,872 όπως φαίνεται και από τον Gurobi.

Πίνακας Α΄.1: Πρόβλημα eil51

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
NN	513.61	1 32 11 38 5 49 9 50 16 2 29 21 34 30 10 39 33 45 15 44 37 17 4 18 47 12 46 51 27 48 6 14 25 13 41 19 42 40 24 23 7 26 8 31 28 3 20 35 36 22 43 1	0 msec
Hill climbing	507.818	1 32 11 38 5 49 9 50 16 2 29 21 34 30 10 39 33 45 15 44 37 17 4 18 47 12 46 51 27 48 6 14 25 13 41 40 19 42 24 23 7 26 8 31 28 3 36 35 20 22 43 1	0 msec
2-opt	491.086	1 32 11 38 5 49 9 50 16 2 29 21 34 30 10 39 33 45 15 44 37 17 4 42 19 40 41 13 18 47 12 46 51 27 48 6 14 25 24 23	1 msec

Συνέχεια στην επόμενη σελίδα

Πίνακας Α΄.1 – Συνέχεια από την προηγούμενη σελίδα

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος
		7 26 8 31 28 36 35 20 3 22 43 1	
Christofides	538.417	1 32 11 38 5 9 50 16 2 29 21 34 30 49 10 39 4 17 37 15 45 33 44 42 19 40 41 13 25 14 18 47 12 46 51 27 48 23 24 7 43 36 35 20 3 28 31 8 26 6 22 1	0 msec
ACO	502.55	1 32 11 38 5 49 9 50 34 30 10 39 33 45 15 44 42 19 41 13 40 4 17 37 12 47 18 14 25 24 43 7 23 48 27 6 51 46 16 2 22 28 31 8 26 3 20 35 36 29 21 1	432 msec
Gurobi	428.872	1 22 8 26 31 28 3 36 35 20 29 2 16 50 21 34 30 9 49 10 39 33 45 15 44 42 19 40 41 13 25 14 24 43 7 23 48 6 27 51 37 5 38 11 32 1 46 12 47 18 4 17	1451 msec

berlin52

Το δεύτερο πρόβλημα είναι το berlin52 με 52 πόλεις όπως φαίνεται και από το όνομα και σύμφωνα με το TSPLIB η βέλτιστη λύση είναι 7542.

Πίνακας Α΄.2: Πρόβλημα berlin52

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος
NN	8,980.92	1 22 49 32 36 35 34 39 40 38 37 48 24 5 15 6 4 25 46 44 16 50 20 23 31 18 3 19 45 41 8 10 9 43 33 51 12 28 27 26 47 13 14 52 11 29 30 21 17 42 7 2 1	0 msec
Hill climbing	8,797.92	1 22 32 49 36 35 34 39 40 37 38 48 24 5 15 6 4 25 46 44 16 50 20 23 31 18 3 45 19 41 8 9 10 43 33 51 12 28 27 26 47	0 msec

Συνέχεια στην επόμενη σελίδα

Πίνακας Α΄.2 – Συνέχεια από την προηγούμενη σελίδα

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος
		13 14 52 11 29 30 21 17 42 7 2 1	
2-opt	8,384.19	1 22 32 49 36 35 34 39 40 37 38 48 24 5 15 6 4 25 28 27 26 47 13 14 52 11 12 51 33 43 10 9 8 41 19 45 3 18 31 23 20 50 44 46 16 29 30 21 17 42 7 2 1	1 msec
Christofides	8,481.08	1 22 31 18 3 17 21 42 7 2 30 23 20 50 29 16 44 34 35 36 39 40 37 38 24 48 46 23 20 47 26 27 13 14 52 28 12 51 11 25 4 6 5 15 43 33 9 10 8 41 19 45 32 49 1	0 msec
ACO	8,918.71	1 32 11 38 5 49 9 50 34 30 10 39 33 45 15 44 42 19 41 13 40 4 17 37 12 47 18 14 25 24 43 7 23 48 27 6 51 46 16 2 22 28 31 8 26 3 20 35 36 29 21 1	460 msec
Gurobi	7,544.37	1 22 8 26 31 28 3 36 35 20 29 2 16 50 21 34 30 9 49 10 39 33 45 15 44 42 19 40 41 13 25 14 24 43 7 23 48 6 27 51 37 5 38 11 32 1 46 12 47 18 4 17	1551 msec

Συνέχεια στην επόμενη σελίδα

pr76

Το επόμενο πρόβλημα είναι το pr76 με 76 πόλεις και η βέλτιστη λύση είναι 108,159. Παρατηρείτε ότι το λογισμικό Gurobi κάνει αρκετό χρόνο για να ολοκληρώσει, 2 λεπτά περίπου παρόλα αυτά ακόμα μπορεί να βρει τη βέλτιστη λύση.

Πίνακας Α΄.3: Πρόβλημα pr76

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος
NN	153,462	1 2 23 22 21 25 24 46 45 44 48 47 69 68 67 50 49 52 53 54 42 43 28 29 30 31 19 20 5 6 7 8 9	0 msec

Συνέχεια στην επόμενη σελίδα

Πίνακας Α΄.3 – Συνέχεια από την προηγούμενη σελίδα

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
		10 11 12 13 14 15 16 17 18 37 36 35 34 40 41 60 59 58 57 63 64 62 61 55 56 51 66 65 71 72 73 39 38 32 33 27 26 4 3 75 76 74 70 1	
Hill climbing	150,417	1 2 23 22 21 25 24 46 45 44 48 47 69 68 67 50 49 52 53 54 42 43 28 29 30 31 19 20 5 6 8 74 9 10 11 12 13 14 15 16 17 18 37 36 35 34 40 41 60 59 58 57 63 64 62 61 55 56 51 66 65 71 72 73 39 38 32 33 27 26 4 3 7 75 76 70 1	0 msec
2-opt	140,038	1 2 75 76 23 22 21 25 24 46 45 44 48 47 69 68 67 50 49 52 53 54 42 43 28 33 32 34 40 41 60 59 58 55 56 51 66 65 71 72 73 64 63 57 62 61 39 38 35 36 37 18 17 16 15 11 10 20 19 31 30 29 27 26 4 3 5 6 7 8 9 12 13 14 74 70 1	4 msec
Christofides	127,335	1 23 22 21 25 24 46 45 44 48 47 69 68 67 70 49 50 52 51 66 65 55 56 53 54 42 43 28 33 32 27 26 29 30 31 19 20 5 6 7 8 4 3 10 9 12 13 14 74 16 15 11 17 18 37 36 38 39 35 34 40 41 60 59 61 62 58 57 63 64 72 73 71 76 75 2 1	1 msec
ACO	131,950	1 2 23 22 21 25 24 46 45 44 48 47 69 68 67 70 71 72 73 63 64 62 61 58 57 55 56 66 65 51 50 49 52 53 54 42 43 28 33 32 29 30 31 19 20 5 6 7 8 9 10 12 11 17 18 37 36 35 34 38 39 59 60 41 40 16 15 14 13 74 75 76 27 26 3 4 1	1,2 sec
Gurobi	108,159	1 76 75 2 3 4 5 6 7 8 9 10 11 12 13 14 74 15 16 17 18 37 36 38 39 40 34 35 33 32 29 30 31 19 20 26 27 28 43 42 54 53 52 55 56 57 58 59 60 41 61 62 63 64 73 72 71 65 66 51 49 50 67 70 68 69 47 48 44 45 46 24 25 21 22 23 1	142 sec

Συνέχεια στην επόμενη σελίδα

rat99

Το πρόβλημα που ακουλουθεί είναι το rat99 με 99 πόλεις και η βέλτιστη λύση του είναι 1,211.

Πίνακας Α΄.4: Πρόβλημα rat99

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
NN	1,564.72	1 2 3 12 11 10 20 19 28 37 29 30 31 32 41 42 4334 33 44 36 35 26 25 24 23 15 16 17 18 9 8 7 65 14 4 13 22 21 38 39 48 40 50 51 52 53 54 45 6362 61 60 49 58 59 67 68 78 77 76 85 84 83 82 9192 93 94 95 86 87 88 79 80 71 70 69 72 81 90 8999 98 97 96 75 74 66 65 56 57 47 46 55 64 73 27 1	0 msec
Hill climbing	1,476.2	1 2 3 12 11 10 20 19 28 37 29 30 31 32 41 33 42 43 34 44 36 35 26 25 24 23 15 16 17 18 9 8 7 6 5 4 14 13 22 21 38 48 39 40 50 51 52 53 45 54 63 62 61 60 49 58 59 67 68 78 77 76 85 84 83 82 91 92 93 94 95 86 87 88 79 80 69 70 71 72 81 89 90 99 98 97 96 75 73 74 66 65 64 55 46 47 56 57 27 1	1 msec
2-opt	1,485.78	1 2 3 4 14 5 6 7 8 9 18 17 16 15 23 24 25 26 35 36 44 34 43 42 33 41 32 31 21 22 13 12 11 10 20 19 28 37 29 30 38 48 39 40 50 51 52 53 45 54 63 62 61 60 49 58 59 67 68 77 78 79 80 69 70 71 72 81 89 90 99 98 97 96 88 87 86 95 94 93 92 91 82 83 84 85 76 75 74 66 65 56 57 47 46 55 64 73 27 1	10 msec
Christofides	1,330.15	1 2 3 12 11 13 22 21 20 19 28 37 29 30 31 32 33 34 43 42 41 40 39 38 48 47 46 55 51 50 49 58 59 57 56 65 64 66 67 68 78 77 76 75 74 73 91 92 93 94 83 82 84 85 86 95 96	3 msec

Συνέχεια στην επόμενη σελίδα

Πίνακας Α΄.4 – Συνέχεια από την προηγούμενη σελίδα

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος**
		97 98 99 90 89 87 88 79 69 70 71 80 81 72 63 62 61 60 52 53 54 45 44 36 35 27 26 25 24 23 15 16 17 18 9 8 7 6 5 14 4 10 1	
ACO	1,510.37	1 2 10 11 12 3 4 14 5 6 7 8 9 18 17 16 15 23 24 25 26 35 36 27 44 34 43 42 33 32 41 40 39 48 38 29 37 28 19 20 21 22 13 30 31 50 51 52 53 54 45 63 62 61 60 49 58 59 67 68 69 70 71 80 81 72 90 89 88 87 95 86 85 84 83 82 91 92 93 94 96 97 98 99 79 78 77 76 75 66 65 56 57 55 46 47 73 74 64 1	2,5 sec
Gurobi	1,219.24	1 10 11 12 13 22 21 20 19 28 37 29 30 31 32 33 34 43 42 41 40 39 48 38 47 46 55 64 65 56 57 58 59 68 67 66 75 74 73 82 83 91 92 93 94 95 96 97 98 99 90 89 79 88 87 86 85 84 76 77 78 69 70 71 80 81 72 63 62 61 60 49 50 51 52 53 54 45 44 36 35 27 26 25 24 23 15 16 17 18 9 8 7 6 5 14 4 3 2 1	28,1 sec

bier127

Με το πρόβλημα bier127 έχει φτάσει το μέγεθος ήδη στους 127 κόμβους και από εδώ και μετά είναι δύσκολο να αναπαριστάτε η λύση οπότε θα φαίνονται μόνο τα αποτελέσματα της τελικής απόστασης στη διαδρομή στους πίνακες. Αυτό το πρόβλημα έχει βέλτιστη τιμή 118,282 και από ότι βλέπουμε το λογισμικό Gurobi έχει αρχίσει να φτάνει στα όρια του και δεν κατάφερε να βρει τη βέλτιστη λύση αλλά μία πολύ κοντά σε αυτήν.

Πίνακας Α΄.5: Πρόβλημα bier127

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	135,752	0 msec
Hill climbing	129,644	4 msec
2-opt	132,020	16 msec
Christofides	132,825	5 msec
ACO	141,483	5 sec
Gurobi	121,472	300 sec

ch130

Το επόμενο πρόβλημα που μελετήθηκε είναι το ch130 με 130 κόμβους. Αυτό το πρόβλημα έχει βέλτιστη τιμή 6,110.

Πίνακας Α΄.6: Πρόβλημα ch130

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	7,575.01	0 msec
Hill climbing	7,404.87	3 msec
2-opt	7,022.66	32 msec
Christofides	7,333.77	5 msec
ACO	8,172.38	5.1 sec
Gurobi	6,310.81	300 sec

tsp225

Ένα ακόμα πρόβλημα είναι το tsp225 με βέλτιστη λύση 3,916. Εδώ φαίνεται ότι έχει αρχίσει να αυξάνεται αρκετά ο χρόνος που θέλει ο ACO για να τελειώσει και ο Gurobi βρίσκει ακόμα λύση αλλά όχι βέλτιστη.

Πίνακας Α΄.7: Πρόβλημα tsp225

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	4,826.35	1 msec
Hill climbing	4,589.03	7 msec
2-opt	4,207.08	104 msec
Christofides	4,698.38	19 msec
ACO	4,799.37	24.4 sec
Gurobi	4,409.42	300 sec

a280

Εδώ έχουμε το πρόβλημα a280 και έχουμε φτάσει στους 280 κόμβους. Η βέλτιστη λύση σε αυτό είναι 2,579.

Πίνακας Α΄.8: Πρόβλημα a280

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	3,148.11	1 msec
Hill climbing	3,100.49	7 msec
2-opt	2,797.03	253 msec
Christofides	3,054.42	33 msec
ACO	3,701.17	45.3 sec
Gurobi	-	Time-Out

pr439

Το επόμενο πρόβλημα πάνω στο οποίο έτρεξαν οι αλγόριθμοι είναι το pr439. Πλέον το μέγεθος έχει φτάσει στους 439 κόμβους και ο Gurobi αδυνατεί να τερματήσει και ο ACO είναι πολύ χρονοβόρος. Η βέλτιστη λύση σε αυτό το πρόβλημα είναι 107,217.

Πίνακας Α΄.9: Πρόβλημα pr439

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	131,282	5 msec
Hill climbing	127,920	26 msec
2-opt	117,611	971 msec
Christofides	122,011	103 msec
ACO	146,947	171 sec
Gurobi	-	Time-Out

rat575

Ένα ακόμα πρόβλημα που μελετήθηκε ήταν το rat575 με 575 κόμβους και η βέλτιστη λύση σε αυτό ήταν 6,773 όπως φαίνεται στο TSPLIB. Από εδώ και πέρα δεν θα συμπεριλαμβάνεται ο Gurobi καθώς δεν κατάφερε να τερματίσει με λύση σε κανένα απλό τα προβλήματα που του δόθηκαν.

Πίνακας Α΄.10: Πρόβλημα rat575

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	8,449.32	7 msec
Hill climbing	8,211.86	54 msec
2-opt	7,445.53	2.5 sec
Christofides	7,922.89	249 msec
ACO	-	Time-out

p654

Το επόμενο πρόβλημα στη σειρά είναι το p654 στο οποίο κατάφεραν να τερματήσουν μόνο οι 4 αλγόριθμοι που φαίνονται παρακάτω. Η βέλτιστη λύση σε αυτό το πρόβλημα ισούται με 34,643.

Πίνακας Α΄.11: Πρόβλημα p654

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	43,410.5	9 msec
Hill climbing	41,900.5	69 msec
2-opt	38,578.9	3.6 sec
Christofides	41,213.5	361 msec

rat783

Ένα ακόμα απο τη βιβλιοθήκη TSPLIB με μέγεθος 783 είναι το rat783 και η βέλτιστή λύση του είναι 8,806.

Πίνακας Α΄.12: Πρόβλημα rat783

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	11,255.1	16 msec
Hill climbing	10,825.2	205 msec
2-opt	9,919.04	7.2 sec
Christofides	10,378.8	631 msec

pr1002

Το πρόβλημα αυτό είναι το πρώτο με μέγεθος πάνω από 1,000 πόλεις και η βέλτιστή λύση του είναι 259,045.

Πίνακας Α΄.13: Πρόβλημα pr1002

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	315,597	26 msec
Hill climbing	305,436	235 msec
2-opt	278,846	13.1 sec
Christofides	300,603	1,2 sec

pcb1173

Το πρόβλημα αυτό είναι μεγέθους 1,173 πόλεων και η βέλτιστή λύση του είναι 56.892.

Πίνακας Α΄.14: Πρόβλημα pcb1173

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	70,277,9	32 msec
Hill climbing	67,881,3	363 msec
2-opt	63,174,5	20.1 sec
Christofides	67,139	1,7 sec

u1432

Εδώ το βέλτιστο μονοπάτι έχει κόστος 152,970 και η λύση βρίσκεται ανάμεσα σε 1,432 διαφορετικές πόλεις.

Πίνακας Α΄.15: Πρόβλημα u1432

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	188,815	46 msec
Hill climbing	180,956	627 msec
2-opt	169,486	37.4 sec
Christofides	180,906	3.3 sec

rl1889

Σε αυτό το πρόβλημα το βέλτιστο μονοπάτι έχει κόστος 316,536 και καλύτερη πιθανή διαδρομή βρίσκεται ανάμεσα σε 1,432 κόμβους.

Πίνακας Α΄.16: Πρόβλημα rl1889

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	400,685	85 msec
Hill climbing	393,536	1927 msec
2-opt	362,013	71.7 sec
Christofides	356,487	3.6 sec

pr2392

Ένα ακόμα πρόβλημα που μελετήθηκε είναι το pr2392 με βέλτιστη λύση ίση με 378,032. Ακόμα και οι 4 αλγόριθμοι τρέχουν κανονικά μέσα σε λογικά χρονικά πλαίσια για το μέγεθος τους.

Πίνακας Α΄.17: Πρόβλημα pr2392

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	461,207	137 msec
Hill climbing	448,229	1,4 sec
2-opt	412,534	178 sec
Christofides	434,253	12 sec

pcb3038

Το επόμενο πρόβλημα είναι το pcb3038 και η βέλτιστη λύση του ισούται με 137,694.

Πίνακας Α΄.18: Πρόβλημα pcb3038

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	175,573	223 msec
Hill climbing	170,948	3 sec
2-opt	154,846	299 sec
Christofides	160,127	23.8 sec

fnl4461

Το προτελευταίο πρόβλημα που μελετήθηκε από τη βιβλιοθήκη TSPLIB αλλά και γενικά είναι το fnl4461 και η βέλτιστη λύση του είναι ίση με 182,566. Εδώ ο αλγόριθμος 2-opt έφτασε και αυτός στο όριο του και δεν μπόρεσε να τερματίσει.

Πίνακας Α΄.19: Πρόβλημα fnl4461

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	227,158	544 msec
Hill climbing	220,001	9.3 sec
2-opt	-	Time-Out
Christofides	217,705	88.5 sec

rl5934

Το τελευταίο πρόβλημα είναι το μεγαλύτερο που μελετήθηκε με μέγεθος 5,934 και η βέλτιστη λύση του είναι ίση με 556,045 και εδώ μόνο 3 αλγόριθμοι κατάφεραν να τερματίσουν.

Πίνακας Α΄.20: Πρόβλημα rl5934

Αλγόριθμος	**Λύση**	**Χρόνος**
NN	683,806	898 msec
Hill climbing	669,774	47.5 sec
2-opt	-	Time-Out
Christofides	616,856	78.8 sec