ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συγκριτική Μελέτη Αλγορίθμων για το Πρόβλημα του Πλανόδιου Πωλητή

Διπλωματική Εργασία

Φωτοπούλου Δ. Ασημίνα

Επιβλέποντες καθηγητές: Πλόσκας Νικόλαος
Στεργίου Κωνσταντίνος

Κοζάνη, Οκτώβριος 2019

ΠΕΡΙΛΗΨΗ

Ένας πλανόδιος πωλητής, επιθυμεί να επισκεφτεί κάθε πόλη από ένα σύνολο κόμβων που διατίθεται, ακριβώς μια φορά και να επιστρέψει στην αρχική του θέση. Το κύριο πρόβλημα που καλείται να αντιμετωπίσει, αφορά την εύρεση της συντομότερης διαδρομής. Στην παρούσα διπλωματική εργασία, παρουσιάζονται κατηγορίες αλγορίθμων όπως οι άπληστοι, προσεγγιστικοί αλλά και αλγόριθμοι επαναληπτικής βελτίωσης, για το πρόβλημα του Πλανόδιου Πωλητή, μερικές πρακτικές εφαρμογές αλλά και κάποιες παραλλαγές του κύριου προβλήματος. Παρουσιάζεται μια εισαγωγή σε αλγοριθμικές και υπολογιστικές απόψεις του προβλήματος, καθώς επίσης και μία ενδελεχής υπολογιστική σύγκριση διάφορων αλγορίθμων για το πρόβλημα του Πλανόδιου Πωλητή. Η διπλωματική εκθέτει τις διαδικασίες επίλυσης του Προβλήματος του Πλανόδιου Πωλητή ανάλογα με το μέγεθος και τη δομή του. Αναλύει και παρουσιάζει επίσης τα αποτελέσματα που προέκυψαν με σαφή και κατανοητό τρόπο μέσω γραφικών παραστάσεων και τέλος, συγκρίνει τα αποτελέσματα αυτά με αποδεδειγμένα βέλτιστες λύσεις.

Λέξεις Κλειδιά: Πρόβλημα του Πλανόδιου Πωλητή, Ελάχιστο Επικαλυπτόμενο Δέντρο, Βέλτιστη Διαδρομή, Αλγόριθμοι Επαναληπτικής Βελτίωσης, Προσεγγιστικοί Αλγόριθμοι, Άπληστοι Αλγόριθμοι.

ABSTRACT

A traveling salesman wants to visit every city, from a given set of cities, exactly once, starting from and returning to his depot. The main problem that has to overcome is to find the shortest path. In this thesis, various algorithmic categories to solve the Traveling Salesman Problem are presented, such us greedy, approximate, and local search heuristics; some practical applications and related problems of the main problem are also presented. An introduction to algorithmic and computational aspects of the Traveling Salesman Problem are presented. A thorough computational comparison of various algorithms is also conducted. This thesis is also investigating various aspects of the Traveling Salesman Problem, such as the size and its structure. Results are presented and analyzed in a concrete way using graphs. The solutions obtained from the algorithms are compared with the optimal solutions.

Key Words: Traveling Salesman Problem, Minimum Spanning Tree, Optimal Route, Iterative Improvement Algorithms, Approximate Algorithms, Greedy Algorithms.

ΔΗΛΩΣΗ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Υπογραφή Φοιτητή

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, επιθυμώ να εκφράσω τις ευχαριστίες μου προς τους καθηγητές μου και επιβλέποντες, Επίκουρο Καθηγητή κ. Πλόσκα Νικόλαο και Καθηγητή κ. Στεργίου Κωνσταντίνο, του Τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, για την βοήθεια και την καθοδήγησή τους στο πρόβλημα που κλήθηκα να αντιμετωπίσω.

Επιπλέον, θα ήθελα να απευθύνω ένα μεγάλο ευχαριστώ στην οικογένεια μου και τους φίλους μου, για όλη τη στήριξη και το κουράγιο που μου παρείχαν, σε ολόκληρη την διάρκεια των σπουδών μου. Τέλος, ένα ιδιαίτερο ευχαριστώ στην αγαπημένη μου φίλη Αγγελική, για την βοήθεια, την στήριξη και την υπομονή της, όλο αυτό το διάστημα.

Κοζάνη 2019,

Φωτοπούλου Ασημίνα

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1^{ο}

Εισαγωγή

1.1 Το Πρόβλημα του Πλανόδιου Πωλητή

Το πρόβλημα του Πλανόδιου Πωλητή (Traveling Salesman Problem, TSP), περιλαμβάνει την εύρεση της μικρότερης δυνατής διαδρομής, που συνδέει και ενώνει ένα πλήθος πόλεων, πιθανόν δεκάδων ή και εκατοντάδων, όπως ακριβώς κάνει και ένας περιπλανώμενος πωλητής, με σκοπό να επισκεφτεί διάφορες πόλεις και να πουλήσει τα προϊόντα που διαθέτει. Αν ένας πωλητής ξεκινήσει από την πόλη του, θελήσει να επισκεφτεί ακριβώς μία φορά κάθε πόλη από μία συγκεκριμένη λίστα που διαθέτει, και τελικώς επιστρέψει στην αρχική του θέση, ένα πολύ σημαντικό ερώτημα είναι, με ποιον τρόπο θα επιλέξει τη σειρά με την οποία θα επισκεφτεί τις πόλεις αυτές, έτσι ώστε η συνολική απόσταση που θα διανύσει, να είναι η μικρότερη δυνατή. Έστω λοιπόν ότι γνωρίζει για κάθε ζευγάρι πόλεων τις μεταξύ τους αποστάσεις. Τότε, θα διαθέτει όλα τα δεδομένα για να βρει την ελάχιστη διαδρομή, όμως δεν θα του είναι καθόλου εύκολο να αξιολογήσει τα δεδομένα αυτά, και να καταλήξει σε ένα συμπέρασμα που θα δίνει λύση στο πρόβλημα του.

Το TSP, αποτελεί ένα NP-Hard πρόβλημα (πρόβλημα το οποίο η λύση του μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο αλλά οι αλγόριθμοι επίλυσης για αυτό το πρόβλημα δεν είναι πολυωνυμικού χρόνου), πολύ σημαντικό στην επιστήμη και στη θεωρία των υπολογιστών. Έτσι, είναι πολύ πιθανό, ο χρόνος εκτέλεσης για οποιονδήποτε αλγόριθμο του

TSP να αυξάνεται εκθετικά, ανάλογα βέβαια και από τον αριθμό των πόλεων που μελετούνται στο εκάστοτε πρόβλημα. Το πρόβλημα του περιοδεύοντος αγοραστή καθώς επίσης και το πρόβλημα δρομολόγησης οχημάτων, αποτελούν γενικεύσεις του παραπάνω προβλήματος.

1.2 Κίνητρο για τη Συγγραφή της Εργασίας

Η παρούσα διπλωματική εργασία, δημιουργήθηκε με σκοπό την μελέτη συγκεκριμένων αλγορίθμων από διαφορετικές κατηγορίες, όπως αυτή των άπληστων, των προσεγγιστικών αλλά και των αλγορίθμων επαναληπτικής βελτίωσης, την ανακάλυψη των κατάλληλων μονοπατιών μετά και την εκτέλεση τους και τελικώς, τη σύγκριση των αλγορίθμων αυτών, έτσι ώστε να αποκτηθεί μια πιο ουσιαστική και σφαιρική εικόνα γύρω από το πρόβλημα του Πλανόδιου Πωλητή, που θα οδηγήσει και στην αποδοτικότερη μελέτη του.

1.3 Σκοπός και Στόχος της Εργασίας

Όπως έχει ήδη αναφερθεί, ο σκοπός αυτής της μελέτης αφορά τη σύγκριση των αλγορίθμων που θα παρουσιαστούν στη συνέχεια, για την αναλυτικότερη και πιο ουσιαστική προσέγγιση στο TSP πρόβλημα. Οι στόχοι της εν λόγω εργασίας, δεν είναι άλλοι από την υλοποίηση των αλγορίθμων με όσο το δυνατόν πιο αποδοτικές δομές δεδομένων και την υπολογιστική σύγκριση των αποτελεσμάτων που θα προκύψουν, αποτελέσματα που αφορούν τους χρόνους εκτέλεσης του εκάστοτε αλγορίθμου.

1.4 Δομή της Εργασίας

Στο 2^{o} κεφάλαιο, θα παρουσιαστούν αρχικά κάποια ιστορικά στοιχεία που αφορούν το πρόβλημα. Θα ακολουθήσουν δύο αναλυτικά παραδείγματα, τα οποία θα εξετάζουν την ανακάλυψη και εύρεση της συντομότερης διαδρομής, και στη συνέχεια τον υπολογισμό της βέλτιστης διαδρομής. Επιπλέον, θα μελετηθεί το πρόβλημα από την πλευρά της θεωρίας των γράφων. Τέλος, θα παρουσιαστεί η μοντελοποίηση των Miller – Tucker – Zemlin, που προσεγγίζει το πρόβλημα από την πλευρά του ακέραιου γραμμικού προγραμματισμού και θα ακολουθήσουν μερικές ειδικές κατηγορίες του TSP προβλήματος, όπως το Graphic TSP, το aTSP και το mTSP.

Στο 3^{o} κεφάλαιο, θα ακολουθήσει η ανάλυση όλων των αλγορίθμων που μελετήθηκαν, καθώς επίσης και οι μεθοδολογίες επίλυσής τους, παρουσιάζοντας αναλυτικά τα βήματα και τους ψευδοκώδικες για κάθε αλγόριθμο. Επιπλέον, θα παρασταθεί ένα γραφικό παράδειγμα εκτέλεσης κάθε αλγορίθμου, για την ουσιαστικότερη κατανόησή του.

Όσον αφορά το 4^{o} κεφάλαιο, εκεί θα αποτυπωθούν όλα τα υπολογιστικά αποτελέσματα για καθένα από τα προβλήματα που εξετάστηκαν, προβλήματα που είτε παράχθηκαν τυχαία μέσω μιας Γεννήτριας Τυχαίων Αριθμών, είτε που ανακτήθηκαν από την βιβλιοθήκη TSPLIB. Εκεί, θα παρουσιαστούν οι λύσεις που προέκυψαν μετά και την υλοποίηση των αλγορίθμων, τα αντίστοιχα μονοπάτια για κάθε πρόβλημα ξεχωριστά, καθώς επίσης και οι συνολικοί χρόνοι εκτέλεσης. Επιπλέον, για τους αλγορίθμους των Christofides, Prim και Kruskal, θα υπολογιστούν και οι χρόνοι εκτέλεσης για τη δημιουργία των Ελάχιστων Επικαλυπτόμενων Δέντρων (MST). Τέλος, θα συμπεριληφθούν όλες οι γραφικές παραστάσεις που αφορούν τους συνολικούς χρόνους εκτέλεσης, καθώς επίσης και οι αποκλίσεις για τις τιμές των μονοπατιών, συναρτήσει των βέλτιστων λύσεων για κάθε αλγόριθμο.

Ολοκληρώνοντας, στο 5^{ο} και τελευταίο κεφάλαιο, θα ακολουθήσει η σύνοψη για τη συγκεκριμένη μελέτη. Θα παρουσιαστούν τα συμπεράσματα που προέκυψαν μετά και το πέρας όλης της πειραματικής διαδικασίας, μερικά συμπεράσματα για τη συμπεριφορά κάθε αλγορίθμου στα προβλήματα που αντιμετωπίστηκαν και θα συνοψίσουμε με τις μελλοντικές εξελίξεις και βελτιώσεις, που θα αφορούν την παρούσα εργασία.

Κεφάλαιο 2^{ο}

Το Πρόβλημα του Πλανόδιου Πωλητή

2.1 Ιστορική Αναδρομή

Η προέλευση του προβλήματος του Πλανόδιου Πωλητή, δεν είναι σαφής. Υπάρχει ένα διαθέσιμο εγχειρίδιο που αφορά το πρόβλημα, δημοσιευμένο το 1832, που περιλαμβάνει παραδείγματα περιηγήσεων τόσο στην Ελβετία, όσο και στη Γερμανία, χωρίς ωστόσο να παρουσιάζει μαθηματικές αποδείξεις και τέτοιου είδους προσέγγιση.

Το TSP εφαρμόστηκε τον 19^{ο} αιώνα από τους μαθηματικούς Hamilton & Kirkman (1985). Ο Hamilton, δημιούργησε το Icosian Puzzle, που περιελάμβανε την εύρεση ενός κύκλου σε έναν γράφο με δώδεκα έδρες, με σκοπό την επίσκεψη κάθε σημείου μονάχα μια φορά και το τελικό σημείο, να είναι ίδιο με το αρχικό. Ο κύκλος αυτός αργότερα ονομάστηκε κύκλος Hamilton, ο οποίος αποτελεί ένα μονοπάτι σε μη κατευθυνόμενο γράφημα, με στόχο την επίσκεψη κάθε κόμβου, ακριβώς μια φορά.

Στην γενική του μορφή, το πρόβλημα του Πλανόδιου Πωλητή, φαίνεται να μελετάται από μαθηματικούς, την περίοδο του 1930 στην Βιέννη, οι οποίοι έδωσαν μια πιο σαφή εικόνα του προβλήματος, Επίσης, την δεκαετία του ‘50 έχουμε την πρώτη προσπάθεια μαθηματικής προσέγγισης από τον M. M. Flood (1956), ο οποίος προσπάθησε να επιλύσει ένα πρόβλημα, σχετικά με τη δρομολόγηση των σχολικών λεωφορείων. Εκείνη την χρονική περίοδο μελετητές από το πανεπιστήμιο του Princeton, εισήγαγαν το όνομα TSP.

Τις δεκαετίες του 1950 & 1960, το πρόβλημα άρχισε να γίνεται ιδιαίτερα γνωστό και δημοφιλές σε επιστημονικούς κύκλους της Ευρώπης και της Αμερικής, καθώς η εταιρία RAND, προσέφερε χρηματική αμοιβή σε όποιον κατάφερνε να δώσει λύση. Σημαντική ήταν η συμβολή των George Dantzig, Delbert R. Fulkerson και Selmer M. Johnson (1959), καθώς κατάφεραν να το εκφράσουν ως πρόβλημα ακέραιου γραμμικού προγραμματισμού και εφάρμοσαν τη μέθοδο αποκοπής επιπέδων (cutting plane). Παρόλο που η μελέτη τους, δεν έδωσε άμεσα αλγοριθμική λύση στο πρόβλημα του Πλανόδιου Πωλητή, οι ιδέες τους οδήγησαν αργότερα στη δημιουργία μεθόδων επίλυσης του προβλήματος, καθώς πέραν των μεθόδων αποκοπής επιπέδων, χρησιμοποίησαν για πρώτη φορά τον αλγόριθμο διακλάδωσης και οριοθέτησης (branch and bound).

Στις επόμενες δεκαετίες, το πρόβλημα μελετήθηκε από πολλούς ερευνητές, από τον χώρο των μαθηματικών, της επιστήμης των υπολογιστών, της φυσικής αλλά και της χημείας. Τότε ήταν που εισήχθη η έννοια του Ελάχιστου Επικαλυπτόμενου Δέντρου (Minimum Spanning Tree, MST), το οποίο ουσιαστικά είναι ένα υποσύνολο ακμών ενός γράφου, που ενώνει όλους τους κόμβους και τα σημεία μεταξύ τους και στο οποίο το συνολικό βάρος των ακμών, είναι το ελάχιστο.

Το 1972, μια ομάδα μελετητών, έδειξε ότι το πρόβλημα με τον κύκλο Hamilton, αποτελούσε ένα πλήρες NP πρόβλημα, από το οποίο καταλήγουμε στο συμπέρασμα ότι και το TSP ανήκει στην κατηγορία των NP προβλημάτων, λόγω της πολυπλοκότητάς του. Με αυτόν τον τρόπο, παρέχεται μια μαθηματική εξήγηση για την υπολογιστική δυσκολία εύρεσης των βέλτιστων μονοπατιών. Αναπτύχθηκαν νέες αλγοριθμικές τεχνικές που εφαρμόστηκαν στο πρόβλημα του Πλανόδιου Πωλητή και απέδειξαν την αποτελεσματικότητά τους. Τέτοια τεχνική αποτελεί η μέθοδος διακλάδωσης και οριοθέτησης (branch and bound).

Το 1976, ο Nicos Christofides (1976), εισήγαγε την έννοια που είναι γνωστή σήμερα σε όλους μας ως το χειρότερο σενάριο. Ο αλγόριθμός του βρίσκει μία λύση στο πρόβλημα του Πλανόδιου Πωλητή το πολύ 1.5 φορές χειρότερη από τη βέλτιστη λύση. Καθώς ο αλγόριθμος του, ήταν σχετικά απλός και γρήγορος, πολλοί υπέθεσαν ότι θα αποτελούσε και μια λύση κοντά σε αυτή της βέλτιστης. Αυτή, αποτελεί μέχρι και σήμερα, την καλύτερη περίπτωση χειρότερου σεναρίου για το πρόβλημα. Ωστόσο, για μερικές συγκεκριμένες περιπτώσεις, υπάρχουν αλγόριθμοι που λειτουργούν καλύτερα στο TSP, οι οποίοι παρουσιάστηκαν το 2011.

Την δεκαετία του ’80, επετεύχθη μεγάλη πρόοδος, όταν ο Grotschel και ο Reinelt (1984) κατάφεραν να δώσουν λύση σε προβλήματα που περιελάμβαναν ως και 2392 πόλεις, χρησιμοποιώντας τις μεθόδους αποκοπής επιπέδων (cutting planes) και διακλάδωσης και οριοθέτησης (branch and bound). Επίσης, λίγα χρόνια αργότερα, δημιουργήθηκε και δημοσιεύτηκε η βιβλιοθήκη TSPLIB από τον Gerard Reinelt (1991), που είναι μια συλλογή από συγκριτικές αξιολογήσεις περιπτώσεων διαφορετικής δυσκολίας και χρησιμοποιείται από πολλούς ερευνητές, για σύγκριση των αποτελεσμάτων.

Τα επόμενα χρόνια, μία ομάδα συνεργατών, που περιελάμβανε τους Applegate, Bixby, Chvatal & Cook (2006), ανέπτυξαν ένα πρόγραμμα γνωστό και ως Concorde, το οποίο έχει χρησιμοποιηθεί σε πολλές πρόσφατες καταγεγραμμένες λύσεις. Τέλος, η συγκεκριμένη ομάδα, υπολόγισε την περίπτωση βέλτιστης περιήγησης, περνώντας μέσα από 33810 διαφορετικές πόλεις, λύση που προέκυψε από ένα πρόβλημα διάταξης ενός microchip, η οποία αποτελεί μέχρι και σήμερα, το μεγαλύτερο πρόβλημα που επιλύθηκε. Σε άλλες περιπτώσεις, όπου υπάρχουν εκατομμύρια διαφορετικές πόλεις, οι λύσεις που μπορούν να προκύψουν, εγγυώνται μονάχα σε ποσοστό 1%, ότι αποτελούν μια βέλτιστη περιήγηση.

Ένας από τους λόγους που έκαναν το TSP ένα τόσο δημοφιλές πρόβλημα, ήταν η στενή και άμεση σχέση που έχει με θέματα συνδυαστικών προβλημάτων, που προέκυπταν από τη νέα μέθοδο γραμμικού προγραμματισμού, με προβλήματα όπως αυτό της μεταφοράς.

Το πρόβλημα του Πλανόδιου Πωλητή, ήταν σαν όλα αυτά τα προβλήματα, με την διαφορά ότι η επίλυση του ήταν δυσκολότερη, και για αυτό η πρόκληση έγινε αρκετά ενδιαφέρουσα. Επίσης, ένας ακόμη λόγος που έγινε τόσο γνωστό, ήταν γιατί το όνομα που το αποδόθηκε, υπενθυμίζει σε όλους και άλλα πράγματα. Ο Πλανόδιος Πωλητής, υπήρξε μια από τις πιο κλασσικές φιγούρες στην Αμερικανική Ιστορία. Για το λόγο αυτό, το TSP έλαβε μεγάλη απήχηση.

2.2 Το Πρόβλημα του Πλανόδιου Πωλητή ως Πρόβλημα Θεωρίας Γράφων

Έστω λοιπόν, ότι υπάρχει ένα πωλητής με μια λίστα πόλεων, τις οποίες επιθυμεί να επισκεφτεί ακριβώς από μια φορά. Ο πωλητής αυτός, επιθυμεί επίσης να ξεκινήσει από την πόλη του και τελικώς, να επιστρέψει και πάλι σε αυτή. Το πρόβλημα στη συγκεκριμένη περίπτωση, ανάγεται σε πρόβλημα εύρεσης του ελάχιστου κατά μήκος μονοπατιού ανάμεσα στις διαθέσιμες πόλεις. Στη συγκεκριμένη περίπτωση, για την ουσιαστικότερη μελέτη του προβλήματος, μπορούμε να αναπαραστήσουμε σε ένα μη κατευθυνόμενο σταθμισμένο γράφο το πρόβλημα (δηλαδή, ως γράφημα στο οποίο οι ακμές του δεν έχουν προσανατολισμό και έχει δοθεί βάρος σε κάθε ακμή), έτσι ώστε κάθε κόμβος να αναπαριστά τις πόλεις και οι ακμές του γραφήματος, να παρουσιάζουν τους άμεσους δρόμους που συνδέουν τις πόλεις αυτές. Επίσης, το μήκος της κάθε ακμής, δηλώνει την απόσταση του δρόμου μεταξύ δυο οποιονδήποτε πόλεων. Συνήθως, το μοντέλο αναπαριστάται με ένα πλήρες γράφημα. Αν δεν υπάρχει ακμή που να συνδέει δυο κόμβους μεταξύ τους, τότε ανάμεσα τους προστίθεται μια αυθαίρετη ακμή με μεγάλο μήκος (άπειρο).

Έστω ότι $K_n = (V_n, E_n)$, ένα πλήρες και μη κατευθυνόμενο γράφημα που έχει:

- $n = |V_n|$ κόμβους
- $m = |E_n|$ ακμές

Μια ακμή με τελικά σημεία τα i και j, μπορεί να αναπαρασταθεί ως $e_{i,j}$. Επίσης, συμβολίζουμε με x, ένα διάνυσμα το οποίο κατατάσσεται από την ακμή (i, j) και συμβολίζεται με x_{ij}. Τότε, αν συνδέσουμε κάθε ακμή $e_{i,j}$ με ένα κόστος c_{ij}, το συμμετρικό πρόβλημα του Πλανόδιου Πωλητή, αποτελείται από την εύρεση ενός κύκλου Hamilton, έτσι ώστε το συνολικό μήκος c της διαδρομής, να είναι το ελάχιστο δυνατό. Το πρόβλημα διατυπώνεται και ως εξής:

Ένα δίκτυο ορίζεται από ένα σύνολο κόμβων V, και ένα σύνολο ακμών E, που συνδέουν τα σημεία αυτά. Έστω ότι το δίκτυο έχει συνολικά n κόμβους. Η απευθείας απόσταση μεταξύ των κόμβων i και j συμβολίζεται με c_{ij}. Αν δεν υπάρχει ακμή που να συνδέει δύο κόμβους, τότε το κόστος θεωρείται άπειρο ($c_{i,j} = \infty$). Επιθυμούμε να βρούμε την ελάχιστη διαδρομή που ξεκινάει από τον κόμβο 1, επισκέπτεται όλα τα σημεία του γραφήματος, το καθένα ακριβώς μια φορά, και επιστρέφει πάλι στον κόμβο 1. Ουσιαστικά, αναζητούμε την εύρεση ενός πλήρους κύκλου Hamilton, έτσι ώστε το συνολικό μήκος του κύκλου, να είναι το ελάχιστο.

Επίσης, ενδιαφέρον έχουν τα προβλήματα όπου οι αποστάσεις των κόμβων θεωρούνται Ευκλείδειες. Σε αυτές τις περιπτώσεις, το πρόβλημα καθορίζεται από τους κόμβους, οι οποίοι αντιστοιχούν στα σημεία του επιπέδου δυο διαστάσεων και η απόσταση μεταξύ τους, είναι η Ευκλείδεια απόσταση μεταξύ των σημείων. Σε αυτή την περίπτωση, ισχύει η τριγωνική ανισότητα, δηλαδή πρέπει να πληρείται το κριτήριο:

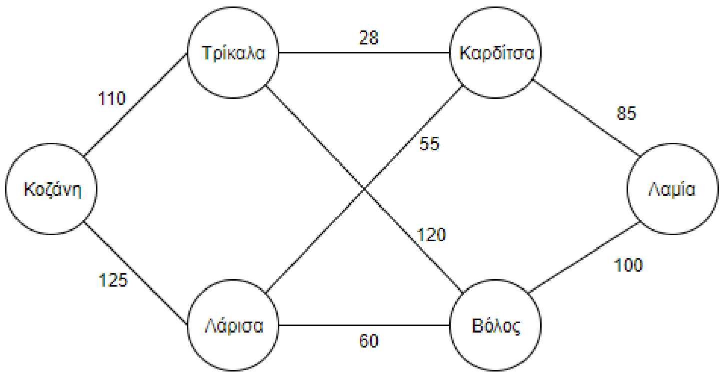
$$c_{i,j} + c_{j,k} \geq c_{i,k} \text{ για κάθε } i, k, j.$$

Στο Ευκλείδειο πρόβλημα TSP, η απόσταση μεταξύ δυο κόμβων ορίζεται ως η Ευκλείδεια απόσταση (έστω δυο πόλεις με συντεταγμένες (x_1,y_1) & (x_2,y_2), τότε η Ευκλείδεια απόσταση ορίζεται ως $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$). Η συγκεκριμένη τακτική, έχει το πλεονέκτημα ότι σε ένα σύνδεσμο από πόλεις μαζί με της χωρικές συντεταγμένες τους, παρέχει ένα σύνολο από συνδέσμους κόμβων, οι οποίοι μας επιτρέπουν να έχουμε πάντοτε ένα πλήρες συνδεδεμένο γράφημα. Επίσης, μια τέτοια παραδοχή, μας επιτρέπει να θεωρήσουμε ότι το πρόβλημα είναι συμμετρικό, δηλαδή η απόσταση μεταξύ δυο κόμβων A και B, είναι ίδια με την απόσταση των B και A.

Όπως γίνεται εύκολα αντιληπτό, το πρόβλημα του Πλανόδιου Πωλητή ανάγεται σε ένα πρόβλημα κύκλου Hamilton. Δηλαδή, δοθέντος ενός πεπερασμένου συνόλου πόλεων και του κόστους μετάβασης μεταξύ αυτών, να βρεθεί ο ελάχιστος κύκλος Hamilton. Με άλλα λόγια, σε ένα πεπερασμένο σύνολο πόλεων και του κόστους διαδρομής μεταξύ κάθε ζεύγους πόλεων, να βρεθεί η βέλτιστη δυνατή διαδρομή, η οποία θα διέρχεται από όλες τις πόλεις ακριβώς μια φορά και θα καταλήγει στην πόλη αφετηρία, ελαχιστοποιώντας το συνολικό κόστος της διαδρομής.

2.3 Παράδειγμα Συντομότερης Διαδρομής

Το πιο αντιπροσωπευτικό παράδειγμα των προβλημάτων του δυναμικού προγραμματισμού, είναι η εύρεση της συντομότερης (ή και μεγαλύτερης, αναλόγως του προβλήματος) διαδρομής, μέσω ενός πεπερασμένου μη κυκλικού δικτύου. Σε τέτοιου είδους παραδείγματα, τα προβλήματα συντομότερης και μεγαλύτερης διαδρομής είναι εναλλάξιμα, δηλαδή το ένα μετατρέπεται στο άλλο, πολλαπλασιάζοντας απλώς τις αποστάσεις και τα βάρη όλων των ακμών με -1. Κοινό χαρακτηριστικό αυτής της κατηγορίας, αποτελεί η ύπαρξη ενός δικτύου το οποίο βρίσκεται σε μία μόνο κατάσταση κάθε φορά, από ένα σύνολο καταστάσεων και το οποίο σε κάθε βήμα του, μεταβαίνει από μια κατάσταση σε μια άλλη, μέσω μιας απόφασης, από ένα σύνολο αποφάσεων. Από κάθε απόφαση που προκύπτει, δημιουργείται ένα κόστος μετάβασης και έτσι προκύπτει επίσης μια ακολουθία από κόστη, κατά τη μετακίνηση του συστήματος. Στην Εικόνα 1, παρουσιάζεται ένα γραφικό παράδειγμα του προβλήματος, το οποίο αναζητά και βρίσκει την συντομότερη δυνατή διαδρομή.



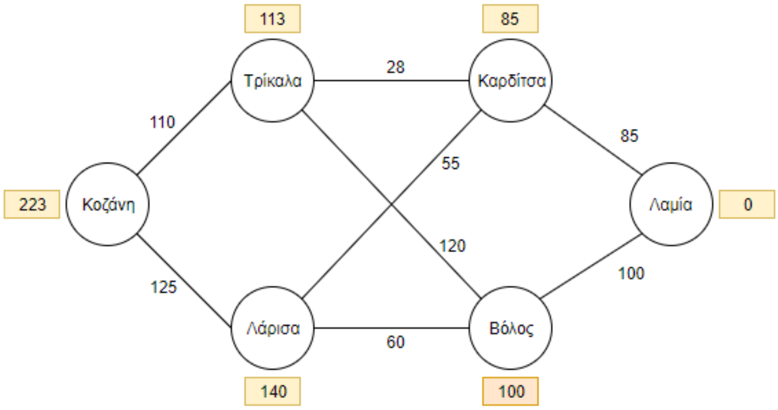
Εικόνα 1: Παράδειγμα Συντομότερης Διαδρομής

Έστω ότι βρισκόμαστε στη πόλη της Κοζάνης και επιθυμούμε να ταξιδέψουμε στη Λαμία, με το αυτοκίνητο. Έστω επίσης, ότι οι πιθανές διαδρομές και οι χιλιομετρικές αποστάσεις, είναι ίδιες με αυτές που περιγράφονται παραπάνω. Σκοπός μας είναι να καθορίσουμε τη συντομότερη διαδρομή από την Κοζάνη στη Λαμία.

Η λύση του προβλήματος, μπορεί να προκύψει από μια πλήρη απαρίθμηση όλων των πιθανών διαδρομών που ξεκινούν από την Κοζάνη και καταλήγουν στη Λαμία. Για παράδειγμα, η συνολική απόσταση της διαδρομής Κοζάνη – Τρίκαλα – Βόλος – Λαμία είναι ίση με 330 χιλιόμετρα. Εφόσον, οι πιθανές διαδρομές στο γράφο είναι $4\ (= 2x2)$, η συνολική προσπάθεια απαρίθμησης των διαδρομών θεωρείται αποδεκτή. Όσο όμως ο αριθμός των πόλεων στο δίκτυο αυξάνεται, τόσο επίσης αυξάνεται και ο αριθμός των πιθανών διαδρομών. Έτσι, η λύση στο πρόβλημα, πρέπει να επιτευχθεί με πιο αποτελεσματικούς τρόπους.

Η βασική ιδέα, είναι να μετατραπεί η λύση του προβλήματος, σε λύση μιας ακολουθίας αλληλεξαρτώμενων υπό - προβλημάτων. Εφόσον έχουμε υποθέσει ότι βρισκόμαστε στην πόλη της Κοζάνης, έχουμε μπροστά μας δύο πιθανές επιλογές για την πρώτη στάση του ταξιδιού, που δεν είναι άλλες από τις πόλεις Τρίκαλα και Λάρισα, με αποστάσεις 110 και 125 χιλιόμετρα, αντίστοιχα. Η πληροφορία αυτή δεν επαρκεί για να φτάσουμε στην σωστή επιλογή. Θέτουμε λοιπόν ένα ερώτημα που θα μας βοηθήσει να λάβουμε μια απόφαση: Ποια επιπλέον πληροφορία χρειαζόμαστε έτσι ώστε να καταλήξουμε σε μια απόφαση? Αν κάποιος μπορούσε να μας δώσει την μικρότερη απόσταση από τα Τρίκαλα και την Λάρισα στην Λαμία, τότε θα μπορούσαμε απλά να προσθέσουμε τις αποστάσεις 110 και 125 που βρήκαμε προηγουμένως, για να καθορίσουμε ποια είναι η καλύτερη πρώτη στάση. Όμως επειδή αυτό δυστυχώς δε μπορεί να πραγματοποιηθεί, παρατηρείται ότι το αρχικό πρόβλημα εύρεσης της μικρότερης απόστασης από την Κοζάνη στη Λαμία, μπορεί να μετατραπεί σε λύση δύο υπό - προβλημάτων, στην εύρεση της μικρότερης απόστασης από τα Τρίκαλα στη Λαμία και από τη Λάρισα στη Λαμία. Παρατηρούμε επίσης ότι αυτά τα δύο υπό - προβλήματα είναι «μικρότερα» από το αρχικό, καθώς οι αποστάσεις είναι μια στάση πιο κοντά στη Λαμία από την Κοζάνη.

Συνεχίζοντας με αυτό το σκεπτικό, για να δοθεί λύση στα δυο υπό - προβλήματα, χρειάζεται να γνωρίζουμε τη μικρότερη απόσταση από την Καρδίτσα και τον Βόλο. Η λύση εδώ, θα προκύψει μέσω μιας αναδρομικής διαδικασίας, η οποία θα ξεκινά από τις πόλεις που απέχουν μόνο ένα βήμα από την Λαμία και προχωρώντας στο αρχικό μας σημείο, που είναι η Κοζάνη, συγκεντρώνοντας μια σειρά από λύσεις με ενδιάμεσα υπό - προβλήματα, δηλαδή δουλεύοντας προς τα πίσω. Στην Εικόνα 2, ακολουθεί η γραφική εκτέλεση της αναδρομικής διαδικασίας:



Εικόνα 2: Παράδειγμα Συντομότερης Διαδρομής

Η διαδικασία ξεκινάει με τις δύο πόλεις, οι οποίες είναι ένα βήμα μακριά από την Λαμία. Οι αποστάσεις αυτές, δίνονται από το παραπάνω σχήμα και είναι οι εξής:

$$\text{Βέλτιστη τιμή για Καρδίτσα} = 85$$
$$\text{Βέλτιστη τιμή για Βόλο} = 100$$

Στη συνέχεια, στρέφουμε την προσοχή μας στις πόλεις Τρίκαλα και Λάρισα, οι οποίες απέχουν δύο βήματα από την πόλη της Λαμίας. Κοιτάζοντας τα Τρίκαλα, ο επόμενος σταθμός είναι οι πόλεις Καρδίτσα και Βόλος. Αν η επιλογή ήταν η πόλη Καρδίτσα, τότε η καλύτερη πιθανή απόσταση από την Λαμία, θα ήταν ίση με $28 + 85$, όπου ο πρώτος αριθμός είναι η απόσταση ανάμεσα στα Τρίκαλα και την Καρδίτσα, και ο δεύτερος, από την Καρδίτσα στη Λαμία. Ομοίως, αν η επιλογή ήταν η πόλη του Βόλου, τότε η απόσταση θα ήταν ίση με 220 χιλιόμετρα. Εφόσον μας ενδιαφέρει η ελαχιστοποίηση της συνολικής απόστασης, η καλύτερη επιλογή θα πρέπει να είναι αυτή που επιτυγχάνει την μικρότερη τιμή από τα δύο αθροίσματα.

$$\text{Η βέλτιστη τιμή στα Τρίκαλα} = \min \begin{cases} \text{Καρδίτσα}: 28 + 85 \\ \text{Βόλος}: 120 + 100 \end{cases}$$

$$= \min \begin{cases} 113 \\ 220 \end{cases}$$

$$= 113 \text{ χιλιόμετρα}$$

Συνεπώς, η καλύτερη επιλογή στην πόλη των Τρικάλων, είναι να ταξιδέψουμε στην Καρδίτσα. Ομοίως υπολογίζουμε και για την πόλη της Λάρισας.

$$\text{Η βέλτιστη τιμή στη Λάρισα} = \min \begin{cases} \text{Καρδίτσα}: 55 + 85 \\ \text{Βόλος}: 60 + 100 \end{cases}$$

$$= \min \begin{cases} 140 \\ 160 \end{cases}$$

$$= 160 \text{ χιλιόμετρα}$$

Συνεπώς η καλύτερη επιλογή στη Λάρισα, είναι να ταξιδέψουμε μέσω της πόλης της Καρδίτσας.

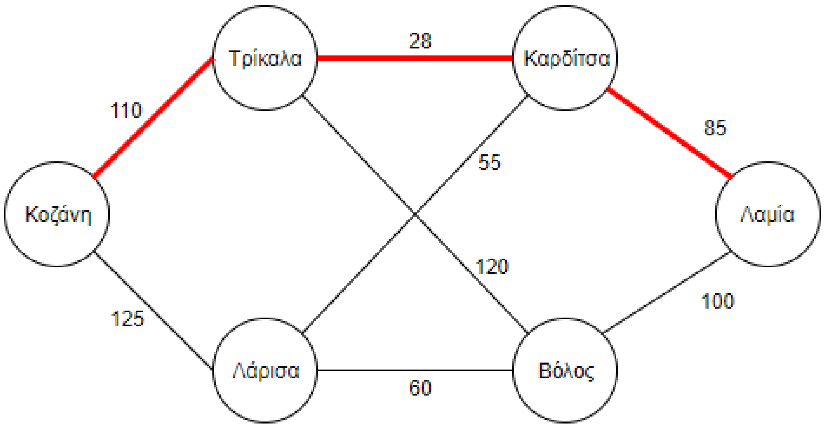
Τελικά, είμαστε στη φάση εξέτασης για την Κοζάνη. Μια παρόμοια σύγκριση μεταξύ των πόλεων των Τρικάλων και της Λάρισας δείχνει ότι:

$$\text{Η βέλτιστη τιμή στη Κοζάνη} = \min \begin{cases} \text{Τρίκαλα}: 110 + 113 \\ \text{Βόλος}: 125 + 140 \end{cases}$$

$$= \min \begin{cases} 223 \\ 265 \end{cases}$$

$$= 223 \text{ χιλιόμετρα}$$

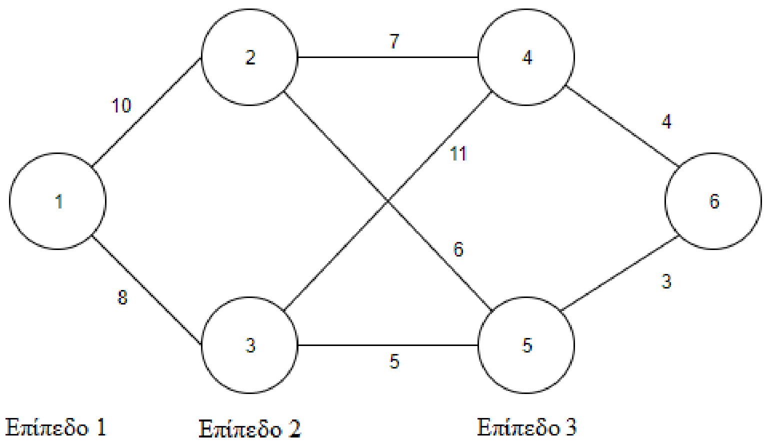
και η καλύτερη επιλογή είναι τα Τρίκαλα. Περιληπτικά λοιπόν, έχουμε καταλήξει στο συμπέρασμα πως η καλύτερη πιθανή συνολική απόσταση από την Κοζάνη στη Λαμία είναι 223 χιλιόμετρα. Επιπλέον, το συντομότερο μονοπάτι που προκύπτει, είναι: **Κοζάνη – Τρίκαλα – Καρδίτσα – Λαμία** Και εδώ ολοκληρώνεται η λύση του προβλήματος. Η Εικόνα 3, παρουσιάζει γραφικά, το μονοπάτι που προέκυψε.



Εικόνα 3: Συντομότερο Μονοπάτι

2.4 Παράδειγμα Βέλτιστης Διαδρομής

Ένας πλανόδιος πωλητής, έχει μερικούς πελάτες σε N πόλεις. Γνωρίζοντας τις αποστάσεις μεταξύ των πόλεων, ο πωλητής ξεκινά το ταξίδι του από την πόλη 1 και επιθυμεί να επιστρέψει σε αυτή, αφού όμως έχει επισκεφτεί κάθε άλλη πόλη, ακριβώς μια φορά, με σκοπό όμως η συνολική διαδρομή που θα διανύσει, να είναι η ελάχιστη δυνατή. Η βασική διαφορά αυτού του προβλήματος, με το πρόβλημα βέλτιστης διαδρομής που παρουσιάστηκε παραπάνω, είναι ότι σε κάθε βήμα πρέπει να γνωρίζουμε πέραν της πόλης στην οποία βρίσκεται, έστω πόλη j, και το σύνολο των πόλεων που έχει ήδη επισκεφτεί. Η Εικόνα 4, παραθέτει ένα παράδειγμα για την εύρεση της βέλτιστης διαδρομής.



Εικόνα 4: Παράδειγμα Βέλτιστης Διαδρομής

Ο πωλητής μας λοιπόν, θέλει να μεταβεί από την πόλη 1, όπου στο σχήμα μας αντιπροσωπεύεται ως ο κόμβος με το νούμερο 1, στην πόλη 6 (κόμβος νούμερο 6). Στη διάρκεια αυτής της διαδρομής, ο πωλητής θα πραγματοποιήσει 2 διανυκτερεύσεις. Η πρώτη θα γίνει σε μία εκ των πόλεων 2 ή 3, και η επόμενη σε έναν από τους κόμβους 4 ή 5. Οι αποστάσεις μεταξύ των πόλεων, αντιστοιχούν στις τιμές των ακμών που φαίνονται στο σχήμα. Το ζητούμενο εδώ, είναι η ελαχιστοποίηση της συνολικής απόστασης που θα διανύσει ο πωλητής, για να μεταβεί από τον κόμβο 1 στον κόμβο 6, πραγματοποιώντας δύο διανυκτερεύσεις.

Επίπεδα

Το πρόβλημα του πλανόδιου πωλητή στη συγκεκριμένη περίπτωση, αναλύεται σε πρόβλημα λήψης διαδοχικών αποφάσεων τριών επιπέδων.

Επίπεδο 1

Στο πρώτο επίπεδο, ο πωλητής πρέπει να επιλέξει, σε ποιον από τους διαθέσιμους κόμβους (2 ή 3) θα μεταβεί, ξεκινώντας από τον κόμβο 1.

Επίπεδο 2

Ο πωλητής καλείται να επιλέξει το επόμενο κόμβο που θα επισκεφτεί (κόμβος 4 ή 5).

Επίπεδο 3

Υπάρχει μόνο μια απόφαση που μπορεί να ληφθεί. Η μετάβαση στον τελικό σταθμό (κόμβος 6).

Καταστάσεις

Σε καθένα από τα 3 επίπεδα, ο πωλητής μπορεί να βρίσκεται σε έναν διαφορετικό κόμβο. Ο κόμβος στον οποίο μπορεί να βρεθεί ο πωλητής, δηλώνει και την κατάστασή του, στο συγκεκριμένο αυτό επίπεδο. Έτσι, αν θεωρήσουμε το επίπεδο 3, οι πιθανές καταστάσεις που μπορεί να βρεθεί είναι οι κόμβοι 4 και 5. Στο επίπεδο 2, οι κόμβοι είναι οι 2 και 3, και για το πρώτο επίπεδο, υπάρχει μόνο μια πιθανή κατάσταση, που αφορά τον κόμβο 1. Ο Πίνακας 1, παραθέτει τα επίπεδα και τις καταστάσεις, του παραδείγματος βέλτιστης διαδρομής.

Επίπεδο	Καταστάσεις
1	1
2	2,3
3	4,5

Πίνακας 1: Παράδειγμα Βέλτιστης Διαδρομής - Καταστάσεις

Αποφάσεις

Σε καθένα από τα 3 επίπεδα που περιγράφηκαν παραπάνω, ο πωλητής θα πρέπει να λάβει μια απόφαση, που θα αφορά την επιλογή του επόμενου κόμβου που θα επισκεφτεί. Ο Πίνακας 2, αναλύει τις πιθανές αποφάσεις που μπορούν να ληφθούν, για τα 3 επίπεδα.

Επίπεδο	Καταστάσεις
1	2,3
2	4,5
3	6

Πίνακας 2: Παράδειγμα Βέλτιστης Διαδρομής - Αποφάσεις

Συνάρτηση αποτελεσμάτων

Ο ορισμός της συνάρτησης αποτελεσμάτων, αποτελεί ένα από τα σημαντικότερα στοιχεία στη διατύπωση ενός προβλήματος που αφορά τον δυναμικό προγραμματισμό. Θα πρέπει να ορίσουμε μια συνάρτηση $f_N(S_N, D_N)$, η οποία αντιστοιχεί σε ένα αποτέλεσμα, ανάλογα με την κατάσταση S_N στην οποία βρίσκεται το σύστημα και την συγκεκριμένη απόφαση που καλείται να λάβει, την οποία συμβολίζουμε με D_N. Πιο συγκεκριμένα, η συνάρτηση $f_N(S_N, D_N)$, αποτελεί την απόσταση από τον κόμβο S_N, ως το τέλος της διαδρομής, δεδομένου όμως ότι η επόμενη απόφαση που θα ληφθεί, (δηλαδή ο κόμβος που θα επιλεχθεί), θα είναι ο D_N.

Ας υποθέσουμε ότι στο επίπεδο N, ο πωλητής βρίσκεται στον κόμβο S_N. Συνεπώς, έχει κάνει την επιλογή του ανάμεσα στις διαθέσιμες αποφάσεις που οδηγούν από τον κόμβο S_N σε κάποιον από το επόμενο επίπεδο. Κάθε απόφαση, έχει ως αποτέλεσμα μια διαφορετική απόσταση $f_N(S_N, D_N)$, της διαδρομής από τον S_N έως το τέλος της διαδρομής. Έτσι λοιπόν, η μικρότερη απόσταση από τον κόμβο S_N ως το τέλος ορίζεται ως:

$$fx(S_N) = min\{f_N(S_N, D_N)\}$$

Αν συμβολίσουμε με $A(S_N,D_N)$ την απόσταση μεταξύ των κόμβων S_N, D_N, τότε θα ισχύει και η επαναληπτική σχέση:

$$f_N(S_N, D_N) = A(S_N, D_N) + fx(N+1)(S_{N+1})$$

Ο παραπάνω επαναληπτικός τύπος, δηλώνει το αποτέλεσμα σε κάθε επίπεδο, εξαρτάται από το αποτέλεσμα του επόμενου επιπέδου και οδηγεί στην εφαρμογή του αλγορίθμου που ακολουθεί στην συνέχεια.

Αλγόριθμος δυναμικού προγραμματισμού

Αν το αποτέλεσμα του επιπέδου N, εξαρτάται από το αποτέλεσμα του επόμενου επιπέδου $N+1$, είναι λογικό να ξεκινήσουμε την ανάλυση από το τελευταίο επίπεδο, θεωρώντας όλες τις πιθανές καταστάσεις του συστήματος του τελευταίου επιπέδου, ως όλες τις πιθανές αποφάσεις που είναι πιθανό να ληφθούν.

Επίπεδο 3

Στο επίπεδο 3, οι πιθανές καταστάσεις είναι δύο. Ο πωλητής βρίσκεται είτε στον κόμβο 4 είτε στον 5. Στο επίπεδο αυτό, υπάρχει μόνο μία δυνατή επιλογή που μπορεί να ακολουθήσει, και είναι η μετάβαση στον κόμβο 6. Ο Πίνακας 3, παραθέτει τα αναλυτικά στοιχεία:

Κατάσταση S_3	Απόφαση D_3	Επόμενος κόμβος S_4	Απόσταση από τον $A(S_3, D_3)$	Απόσταση από τον επόμενο κόμβο ως το τέλος $fx_4(S_4)$	Συνολική απόσταση $f_4(S_4, D_4)$
4	[4,6]	6	4	0	$4 \leftarrow fx3(4)$
5	[5,6]	6	3	0	$3 \leftarrow fx3(5)$

Πίνακας 3: Παράδειγμα Βέλτιστης Διαδρομής - Επίπεδο 3

Με $fx3(4)$ & $fx3(5)$, συμβολίζεται η ελάχιστη απόσταση από κάθε ένα από τους κόμβους του τρίτου επιπέδου, ως το τέλος της διαδρομής.

Επίπεδο 2

Εξετάζοντας το επίπεδο 2, οι πιθανές καταστάσεις είναι και εδώ δύο. Ο πωλητής θα βρίσκεται στον κόμβο 2 ή 3 και σε κάθε κόμβο του επόμενου επιπέδου, θα έχει δύο δυνατές επιλογές (κόμβοι 4 & 5). Υποθέτουμε ότι βρίσκεται στον κόμβο 2. Από εκεί, είτε θα ακολουθήσει την διαδρομή [2-4], είτε την [2-5]. Στο Πίνακα 4, παρουσιάζονται ολοκληρωμένα τα στοιχεία του επιπέδου 2:

Κατάσταση S_2	Απόφαση D_2	Επόμενος κόμβος S_3	Απόσταση από τον $A(S_2, D_2)$	Απόσταση από τον επόμενο κόμβο ως το τέλος $fx_3(S3)$	Συνολική απόσταση $f_2(S_2, D_2)$
2	[2,4]	4	7	4	11
2	[2,5]	5	6	3	$9 \leftarrow fx2(2)$
3	[3,4]	4	11	4	15
3	[3,5]	5	5	3	$8 \leftarrow fx2(3)$

Πίνακας 4: Παράδειγμα Βέλτιστης Διαδρομής - Επίπεδο 2

Σύμφωνα λοιπόν με τον παραπάνω πίνακα, οι συντομότερες διαδρομές από τους κόμβους του επιπέδου 2, ως τον τελικό προορισμό:

- από τον κόμβο 2: ο επόμενος θα είναι ο 5 με συνολική απόσταση έως το τέλος ίση με 9.
- από τον κόμβο 3: ο επόμενος θα είναι επίσης ο 5, με συνολική απόσταση 8.

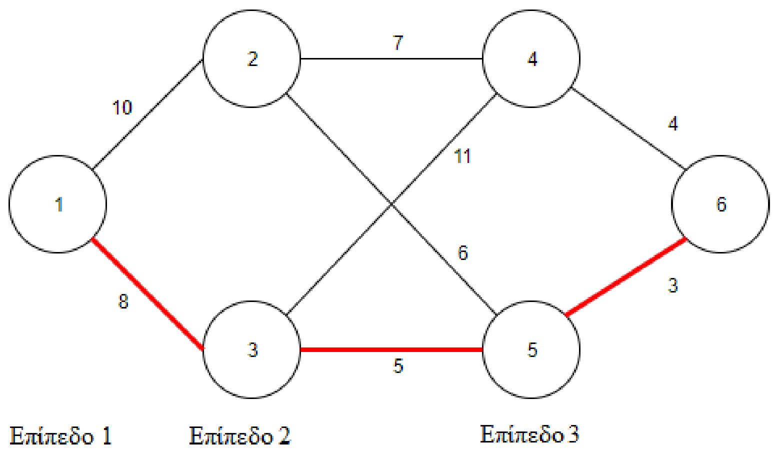
Επίπεδο 1

Εδώ, υπάρχει μονάχα μία κατάσταση στην οποία μπορεί να βρίσκεται ο πωλητής και αφορά τον κόμβο 1. Από εκεί, ο επόμενος κόμβος μπορεί να είναι ο 2 ή ο 3. Τα αποτελέσματα του Πίνακα 5, οδηγούν και στην λύση του προβλήματος.

Κατάσταση S_1	Απόφαση D_1	Επόμενος κόμβος S_2	Απόσταση από τον $A(S_1, D_1)$	Απόσταση από τον επόμενο κόμβο ως το τέλος $fx_2(S_2)$	Συνολική απόσταση $f_1(S_1, D_1)$
1	[1,2]	2	10	9	20
1	[1,3]	3	8	8	$16 \leftarrow fx1(1)$

Πίνακας 5: Παράδειγμα Βέλτιστης Διαδρομής - Επίπεδο 1

Εξάγεται λοιπόν το συμπέρασμα ότι η βέλτιστη απόφαση για τον πωλητή, είναι να ακολουθήσει την διαδρομή [1-3]. Επομένως, στο επίπεδο 2, ο πωλητής θα βρίσκεται στον κόμβο 3 και από εκεί προκύπτει το συμπέρασμα ότι ο πωλητής θα ακολουθήσει το μονοπάτι [3-5], με κόστος μετάβασης ίσο με 5. Τέλος, δεδομένου ότι ο πωλητής βρίσκεται στον κόμβο 5, και έχοντας μπροστά του μονάχα μια επιλογή για μετάβαση, ακολουθεί την διαδρομή [5-6], η οποία έχει κόστος 3. Έτσι, καταλήγουμε στο τελικό μας συμπέρασμα και παρατηρούμε ότι η διαδρομή **1-3-5-6,** αποτελεί την βέλτιστη, με συνολική απόσταση ίση με 16. Η Εικόνα 5, παρουσιάζει με χρώμα κόκκινο, η βέλτιστη διαδρομή:



Εικόνα 5: Μονοπάτι Βέλτιστης Διαδρομής

2.5 Η Μέθοδος Μοντελοποίησης MTZ, με τη Χρήση Ακέραιου Γραμμικού Προγραμματισμού

Το πρόβλημα του Πλανόδιου Πωλητή, παρόλο που είναι αρκετά απλό σε περιγραφή, είναι αρκετά δύσκολο στην επίλυση του και έχει ένα τεράστιο αριθμό από ακριβείς προσεγγίσεις. Σε αυτή την ενότητα, θα μελετήσουμε τη μέθοδο των Miller – Tucker - Zemlin, γνωστή και ως MZT (1960), η οποία είναι αποδοτική για μικρά προβλήματα. Η μέθοδος αυτή είναι αρκετά ξεκάθαρη και μπορεί να υπολογιστεί με συγκεκριμένα προγράμματα. Θα αναφερόμαστε στις πόλεις ως κόμβους και στις αποστάσεις μεταξύ δυο κόμβων, ως ακμές. Οι ακμές λοιπόν, θα αναπαριστούνται με δυαδικές μεταβλητές, 0 αν δεν υπάρχει ακμή, δηλαδή δεν έχουμε σύνδεση δυο κόμβων, και 1 αν υπάρχει η ακμή.

Οι έγκυρες διαδρομές, έχουν το χαρακτηριστικό ότι κάθε κόμβος έχει μία ακμή που εισέρχεται σε αυτόν, και μία που εξέρχεται από αυτόν. Αυτό μπορεί να περιγραφεί, αθροίζοντας όλες τις μεταβλητές των ακμών που εισέρχονται σε έναν κόμβο, ορίζοντας ένα άθροισμα ίσο με 1. Καθώς όλες οι μεταβλητές των ακμών, θα έχουν τιμή είτε 0 είτε 1, η προσέγγιση αυτή θα δουλεύει εξαιρετικά, καθώς θα εγγυάται μονάχα μια εισερχόμενη ακμή. Με τον ίδιο τρόπο, προκύπτουν και οι εξερχόμενες ακμές. Έτσι, έχουμε $2n$ περιορισμούς για n κόμβους, και η μέθοδος έχει σχεδόν ολοκληρωθεί.

Παραμένει ένα πρόβλημα, που αφορά τα υπό - μονοπάτια. Μπορεί να προκύψει μια λύση, η οποία θα αποτελείται από πολλαπλά κλειστά μονοπάτια, αντί για ένα, που θα περιελάμβανε όλους τους κόμβους. Μια τυπική προσέγγιση, αθροίζει τις ακμές για κάθε κατάλληλο υποσύνολο κόμβων, έτσι ώστε να διασφαλιστεί ότι δεν υπάρχουν αρκετές ακμές, για να συνδέσουν πλήρως τους κόμβους του υποσυνόλου σε ένα κλειστό μονοπάτι. Ο αριθμός των πιθανών υποσυνόλων των κόμβων, μπορεί να γίνει τεράστιος και να μην μπορούν να αναπαρασταθούν όλοι μια φορά. Η MTZ μέθοδος είναι αρκετά απλή, παρόλο που ο αριθμός των περιορισμών παραμένει μεγάλος, της τάξης του n^2.

Με αυτόν τον τρόπο, καθορίζουμε τον αρχικό κόμβο 1, και διασφαλίζουμε πως δε θα υπάρχουν καθόλου κλειστά μονοπάτια στο γράφημά μας, που θα περιλαμβάνουν τον κόμβο αυτό. Μαζί με τους περιορισμούς για τις εισερχόμενες και εξερχόμενες ακμές, αναγκάζουμε το μονοπάτι μας, να είναι ένα και μοναδικό κλειστό, περιλαμβάνοντας και τον κόμβο 1. Όλοι οι περιορισμοί που παρουσιάζονται παρακάτω, αφορούν ένα και μοναδικό μονοπάτι, παρόλο που η μέθοδος MTZ μπορεί να εφαρμοστεί και σε πολλαπλά:

$$\min \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} c_{i,j} \; x \; X_{i,j}$$

Αντικειμενική συνάρτηση

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} X_{i,j} = 1, \qquad j = 1,2,\ldots,n$$

περιορισμός (a)

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} X_{i,j} = 1, \qquad i = 1,2,\ldots,n$$

περιορισμός (b)

$$u_i - u_j + nX_{ij} \leq n - 1, \qquad 2 \leq i,j \leq n, \qquad i \neq j$$

περιορισμός (c)

Η αντικειμενική συνάρτηση του προβλήματος. Με $C_{i,j}$ αναπαριστούμε τα κόστη μετάβασης από τον κόμβο i στον κόμβο j και με $X_{i,j}$ το αν υπάρχει σύνδεση μεταξύ των κόμβων i,j. Ο περιορισμός (a) αφορά τις εξερχόμενες ακμές των κόμβων, ο (b) τις εισερχόμενες και με τον περιορισμό (c), διασφαλίζουμε ότι δεν θα υπάρχουν πολλαπλά κλειστά μονοπάτια στον γράφο μας. Μπορούμε να προσδιορίσουμε τους περιορισμούς οι οποίοι αφορούν τα υπό-μονοπάτια:

$$2\binom{n-1}{2} = (\mathrm{n}-1)(\mathrm{n}-2)$$

Όταν εφαρμόζεται ένας περιορισμός που αφορά ένα υπό - μονοπάτι στον γράφο μας, ένα συνδεδεμένο ζεύγος κόμβων δεν θα πάρει τιμή άμεσα και θα χρειάζεται έναν δείκτη:

$$r_i, r_{i+1}$$

Έτσι, ο περιορισμός που αφορά το υπό - μονοπάτι θα τροποποιηθεί και θα πάρει την μορφή:

$$ur_i - ur_{i+1} \leq -1$$

Εάν ακολουθήσουμε το μονοπάτι και αθροίσουμε τους περιορισμούς, μη συμπεριλαμβάνοντας τον κόμβο 1 και τον n, και έχοντας κλειστό μονοπάτι $r_j = r_k$, το συνολικό άθροισμα που προκύπτει είναι:

$$0 = ur_j - ur_k \leq -(k - j - 1) < 0,$$

το οποίο είναι άτοπο. Έτσι, δεν υπάρχουν καθόλου κλειστά μονοπάτια, εάν αφαιρέσουμε τον κόμβο 1 (κανένα υπό - μονοπάτι).

Συγκεκριμένα, για το πρόβλημα που αντιμετωπίσαμε παραπάνω, οι περιορισμοί συνοψίζονται εδώ:

Αντικειμενική συνάρτηση: $C_{1,2}\, x\, X_{1,2} + C_{1,3}\, x\, X_{1,3} + C_{1,4}\, x\, X_{1,4} + C_{1,5}\, x\, X_{1,5}$

$$+ C_{2,1}\, x\, X_{2,1} + C_{2,3}\, x\, X_{2,3} + C_{2,4}\, x\, X_{2,4} + C_{2,5}\, x\, X_{2,5}$$

$$+ C_{3,1}\, x\, X_{3,1} + C_{3,2}\, x\, X_{3,2} + C_{3,4}\, x\, X_{3,4} + C_{3,5}\, x\, X_{3,5}$$

$$+ C_{4,1}\, x\, X_{4,1} + C_{4,2}\, x\, X_{4,2} + C_{4,3}\, x\, X_{4,3} + C_{4,5}\, x\, X_{4,5}$$

$$+ C_{5,1}\, x\, X_{5,1} + C_{5,2}\, x\, X_{5,2} + C_{5,3}\, x\, X_{5,3} + C_{5,4}\, x\, X_{5,4}$$

Περιορισμοί που αφορούν τις εξερχόμενες ακμές: $X_{1,2} + X_{1,3} + X_{1,4} + X_{1,5} = 1$

$$X_{2,1} + X_{2,3} + X_{2,4} + X_{2,5} = 1$$

$$X_{3,1} + X_{3,2} + X_{3,4} + X_{3,5} = 1$$

$$X_{4,1} + X_{4,2} + X_{4,3} + X_{4,5} = 1$$

$$X_{5,1} + X_{5,2} + X_{5,3} + X_{5,4} = 1$$

Περιορισμοί που αφορούν τις εισερχόμενες ακμές: $X_{2,1} + X_{3,1} + X_{4,1} + X_{5,1} = 1$

$$X_{1,2} + X_{3,2} + X_{4,2} + X_{5,2} = 1$$

$$X_{1,3} + X_{2,3} + X_{4,3} + X_{5,3} = 1$$

$$X_{1,4} + X_{2,4} + X_{3,4} + X_{5,4} = 1$$

$$X_{1,5} + X_{2,5} + X_{3,5} + X_{4,5} = 1$$

Τέλος, ο περιορισμός που χρησιμοποιείται για να ελέγχει το εάν υπάρχουν κλειστά μονοπάτια στον γράφο μας, καταλήγει ως εξής: $U_2 - U_3 + 5X_{2,3} \leq 4$

$$U_2 - U_4 + 5X_{2,4} \leq 4$$

$$U_3 - U_4 + 5X_{3,4} \leq 4$$

$$U_4 - U_2 + 5X_{4,2} \leq 4$$

2.6 Διάφορες Παραλλαγές του Προβλήματος

Στη συνέχεια, θα παρουσιάσουμε τους βασικούς ορισμούς για τις διαφοροποιήσεις ή παραλλαγές του προβλήματος του Πλανόδιου Πωλητή, δίνοντας τους κατάλληλους μετασχηματισμούς. Ακολουθούν δυο από τις σημαντικότερες εξ αυτών, i) το ασύμμετρο TSP και ii) το πολλαπλό TSP.

2.6.1 Το TSP σε Γενικές Κατηγορίες Γραφημάτων

Υπάρχουν καταστάσεις όπου επιθυμούμε να βρούμε σε αυθαίρετα γραφήματα $G\ (V, E)$, τους μικρότερους κύκλους Hamilton, κυρίως σε γραφήματα που δεν είναι πλήρη. Ανάλογα με τις απαιτήσεις, μπορούμε να διαχειριστούμε τέτοιες περιπτώσεις με δυο τρόπους.

Αν απαιτείται κάθε κόμβο να τον επισκεφτούμε ακριβώς μια φορά και μόνο οι ακμές που υπάρχουν να χρησιμοποιηθούν, τότε κάνουμε τα παρακάτω βήματα: Προσθέτουμε όσες ακμές λείπουν, δίνοντάς τους ένα αρκετά μεγάλο αριθμό M ως βάρος και εφαρμόζουμε έναν αλγόριθμο για το TSP, πλήρων γραφημάτων. Αν ο αλγόριθμος τερματίσει με μια βέλτιστη λύση που δεν περιέχει καμία από τις ακμές με βάρος M, τότε η λύση αυτή που προκύπτει είναι επίσης βέλτιστη. Αν μια ακμή με βάρος M υπάρχει στη βέλτιστη λύση, τότε το αρχικό γράφημα δεν περιέχει ένα κύκλο Hamilton. Οι ευρετικές συναρτήσεις, δε μπορούν να εγγυηθούν την εύρεση ενός κύκλου Hamilton στο γράφημα G, έστω και αν αυτός υπάρχει. Αυτό επιτυγχάνεται χρησιμοποιώντας ακριβείς αλγορίθμους.

Ο δεύτερος τρόπος για να διαχειριστούμε προβλήματα τέτοιου είδους, είναι να επιτρέψουμε την επίσκεψη στους κόμβους περισσότερο από μια φορές, και να διασχίσουμε τις ακμές επίσης περισσότερο από μια φορές. Αν το γράφημα που δίνεται είναι πλήρες, τότε μπορούμε πάντοτε να βρούμε μια εφικτή διαδρομή με επιστροφή, υπό αυτή τη χαλάρωση. Αυτό αποτελεί το γραφικό TSP, που θα αναφερθούμε λεπτομερώς στη συνέχεια.

2.6.2 Το Γραφικό TSP

Όπως συμβαίνει και στην απλή περίπτωση του προβλήματος του Πλανόδιου Πωλητή, μας δίνεται ένα σύνολο που περιλαμβάνει n πόλεις, οι συνδέσεις μεταξύ αυτών που παρουσιάζονται σε ένα γράφο $G\ (V, E)$, με V τους κόμβους και E τις δοθείσες ακμές, και ένα μήκος c για κάθε σύνδεση. Υποθέτουμε ότι ο γράφος μας είναι πλήρης, αλλιώς δεν θα υπάρχει εφικτή λύση. Το γραφικό TSP (Graphic TSP), αποτελείται από την εύρεση ενός μονοπατιού για τον πωλητή, όπου θα επισκέπτεται κάθε πόλη, με την προϋπόθεση ότι θα διανύσει την μικρότερη δυνατή απόσταση.

Για να πραγματοποιηθεί λοιπόν αυτό το μονοπάτι, ο πωλητής πρέπει να φύγει από την πόλη του (η επιλογή της αρχικής θέσης - κόμβου μπορεί να είναι οποιαδήποτε), να επισκεφτεί επίσης οποιαδήποτε άλλη πόλη επιθυμεί, τουλάχιστον μια φορά και να επιστρέψει στον κόμβο από τον οποίο ξεκίνησε. Επίσης, είναι πολύ πιθανό, να πραγματοποιηθεί σε μια πόλη – κόμβο, επίσκεψη παραπάνω από μία φορές και μια ακμή του δοθέντος γραφήματος G, να την διασχίσουμε επίσης περισσότερες από μια φορές. Ένα τέτοιο ταξίδι του πωλητή, καλείται περιήγηση.

Η παραπάνω εισήγηση, αποτελεί έναν πιο πρακτικό και ουσιαστικό ορισμό του προβλήματος, ίσως γιατί μπορούν να υπάρχουν περιπτώσεις όπου το βασικό μας γράφημα που περιλαμβάνει τις συνδέσεις, να μην ανήκει στην κατηγορία γραφημάτων Hamilton. Έτσι λοιπόν, μετασχηματίζουμε ένα γραφικό TSP σε κλασσικό TSP ως εξής: θεωρούμε το TSP ως πλήρες γράφημα $K_n\ (V_n, E_n)$, όπου για κάθε ακμή i, j η αντικειμενική συνάρτηση του $d_{i,j}$, προκύπτει από το μήκος c του συντομότερου μονοπατιού από το i στο j, στον γράφο G που δίνεται. Λύνοντας το πρόβλημα στο Kn, προκύπτει ένας κύκλος Hamilton. Η λύση του γραφικού TSP μπορεί να εξασφαλιστεί, αντικαθιστώντας κάθε ακμή του κύκλου Hamilton, που δεν ανήκει στο γράφημα G, με τις ακμές ενός συντομότερου μονοπατιού, που τελικώς θα συνδέει όλα τα σημεία του γραφήματος G.

2.6.3 Το Ασύμμετρο TSP

Σε αυτή την κατηγορία προβλημάτων, το κόστος μετάβασης από την πόλη A στην B, δεν είναι απαραίτητα ίδιο με αυτό από την μετάβαση από την πόλη B στην A. Αυτό προκύπτει διατυπώνοντας το Ασύμμετρο TSP (aTSP), ως την εύρεση ενός ελάχιστου κατευθυνόμενου κύκλου Hamilton σε ένα γράφημα με βάρη. Έστω λοιπόν ότι $D = (W, A)$, $W = \{1,2, \dots\dots, n\}$, A υποσύνολο των WxW, είναι ο διγράφος που το aTSP πρέπει να επιλυθεί. Έστω επίσης, ότι $d_{i,j}$ είναι η απόσταση από τον κόμβο i στον j. Ορίζουμε ένα μη κατευθυνόμενο γράφημα $G\ (V, E)$:

$$V = W \cap \{n + 1, n + 2, \dots, 2n\}$$

$$E = \{(i, n + i)/i = 1, 2, \dots, n\} \cup \{(n + i, j)/i, j \in A\}$$

Τα βάρη των ακμών, υπολογίζονται ως εξής:

$$c_{i,n+1} = -M, \qquad i = 1,2, \dots, n$$

$$c_{n+1,j} = d_{i,j}, \qquad (i, j) \in A,$$

όπου M, είναι ένας πολύ μεγάλος αριθμός.

Είναι λοιπόν εύκολο να παρατηρήσει κανείς, ότι για κάθε κατευθυνόμενο κύκλο Hamilton στο D με μήκος d_D, υπάρχει ένας κύκλος Hamilton στο γράφημα G, με μήκος $c_G = d_D - n_M$. Επιπλέον, όλες οι ακμές με βάρος $-M$, υπάρχουν σε ένα βέλτιστο κύκλο Hamilton στο γράφημα G. Επομένως, αυτός ο κύκλος επιφέρει ένα κατευθυνόμενο κύκλο Hamilton στο D.

2.6.4 Το Πολλαπλό TSP

Για την μελέτη του προβλήματος του πολλαπλού TSP (mTSP), αντί να διατίθεται μονάχα ένας πωλητής, έχουμε m διαθέσιμους πωλητές, που βρίσκονται στην πόλη και πρέπει να επισκεφτούν όλες τις υπόλοιπες. Ως συνολικό κόστος λύσης σε αυτή την περίπτωση, αποτελεί η απόσταση που διανύεται από το σύνολο των m πωλητών (όλοι τους πρέπει να ταξιδέψουν). Αυτή αποτελεί τη βασική προϋπόθεση, όταν για παράδειγμα σε προβλήματα δρομολόγησης οχημάτων, τα οχήματα βρίσκονται σε έναν κοινό σταθμό και πρέπει να εξυπηρετήσουν πελάτες.

Υπάρχει η δυνατότητα μετατροπής του προβλήματος σε απλό TSP, χωρίζοντας την πόλη $n+1, m$, σε πόλεις, $n+1, n+2, \dots, n+m$, αντίστοιχα. Οι ακμές $(i, n+k)$, με $1 \leq i \leq n$ και $2 \leq k \leq m$, έχουν βάρος που προκύπτει ως $c(i, n+k) = c(i, n+1)$ και όλες οι ακμές που συνδέουν τους κόμβους $n+1, n+2, \dots, n+m$, λαμβάνουν μια μεγάλη τιμή M, που αφορά το βάρος τους.

Κεφάλαιο 3ο

Αλγόριθμοι Επίλυσης του Προβλήματος του Πλανόδιου Πωλητή

3.1 Αλγόριθμοι & Μεθοδολογίες Επίλυσης του TSP

Θεωρούμε ότι έχουμε έναν πωλητή, που ξεκινά από την πόλη του και πρέπει να επισκεφτεί ένα σύνολο πόλεων $A = \{1,2,\dots,N\}$. Βασική παραδοχή εδώ, είναι ότι επισκέπτεται την πόλη ακριβώς μια φορά, περνάει από όλες τις πόλεις και τελικά, να επιστρέφει και πάλι στην αρχική του θέση. Σκοπός λοιπόν, είναι η εύρεση του δρομολογίου που θα ελαχιστοποιεί το συνολικό μήκος της διαδρομής. Στις επόμενες παραγράφους, παρουσιάζονται οι βασικές μέθοδοι επίλυσης του προβλήματος.

Συγκεκριμένα, μελετήθηκαν έξι αλγόριθμοι, οι οποίοι επιδιώκουν να δώσουν λύση. Αρχικά, μελετήθηκε η συμπεριφορά μιας κατηγορίας αλγορίθμων, των προσεγγιστικών. Οι αλγόριθμοι αυτοί, παρέχουν με εγγυημένη ποιότητα, την λύση του προβλήματος. Σε αυτή τη κατηγορία, ανήκει ο αλγόριθμος Christofides, ο οποίος εγγυάται ότι οι λύσεις θα είναι εντός συντελεστή 3/2 του μήκους της βέλτιστης λύσης.

Στη συνέχεια, μελετήθηκαν τεχνικές άπληστων αλγορίθμων, οι οποίοι κάνουν πάντοτε την επιλογή η οποία φαίνεται η καλύτερη δυνατή, τη δεδομένη στιγμή. Με άλλα λόγια, κάνουν μια τοπικά βέλτιστη επιλογή, με την ελπίδα ότι η επιλογή αυτή θα οδηγήσει αργότερα, σε μια καθολικά βέλτιστη λύση. Τέτοιοι αλγόριθμοι είναι ο αλγόριθμος Prim, ο αλγόριθμος Kruskal και ο αλγόριθμος Πλησιέστερου Γείτονα (Nearest Neighbor).

Επιπλέον, μελετήθηκε και η κατηγορία των αλγορίθμων επαναληπτικής βελτίωσης, με γνωστότερο όλων τον αλγόριθμο 2-opt, ο οποίος διαγράφει ένα ζεύγος μη γειτονικών ακμών και επανασυνδέει τα τερματικά τους σημεία, δημιουργώντας έτσι ένα άλλο δρομολόγιο.

Τέλος, στο πρόβλημα του Πλανόδιου Πωλητή, εξετάστηκε και η μέθοδος Brute – Force που ανήκει στην κατηγορία αλγορίθμων εξαντλητικής αναζήτησης, όπου μελετούνται μία μία όλες οι πιθανές λύσεις στο πρόβλημα, για να βρεθεί τελικώς η βέλτιστη λύση.

3.2 Ο Αλγόριθμος του Christofides

Όπως ήδη έχει αναφερθεί, ο προσεγγιστικός αλγόριθμος που εξετάστηκε στην παρούσα εργασία, δεν είναι άλλος από αυτού του Christofides. Ο συγκεκριμένος, υπολογίζει προσεγγιστικές λύσεις για το πρόβλημα του πλανόδιου πωλητή, σε περιπτώσεις όπου οι αποστάσεις σχηματίζουν έναν μετρικό χώρο (είναι συμμετρικές και ακολουθούν την συνθήκη της τριγωνικής ανισότητας).

Ο αλγόριθμος αυτός, εγγυάται την εύρεση λύσεων το πολύ 3/2 χειρότερες του μήκους της βέλτιστης λύσης και ονομάστηκε έτσι από τον Nikos Christofides, ο οποίος τον δημοσίευσε το 1976. Μέχρι και σήμερα, αποτελεί την καλύτερη υλοποίηση αλγορίθμου για το πρόβλημα του Πλανόδιου Πωλητή.

Έστω λοιπόν ότι έχουμε έναν πλήρη γράφο $G = (V, w)$, όπου V είναι ο αριθμός των κορυφών και w τα βάρη τον εκάστοτε ακμών. Σύμφωνα με την συνθήκη της τριγωνικής ανισότητας, για κάθε τρεις ακμές u, v & x θα ισχύει: $w(uv) + w(vx) \geq w(ux)$.

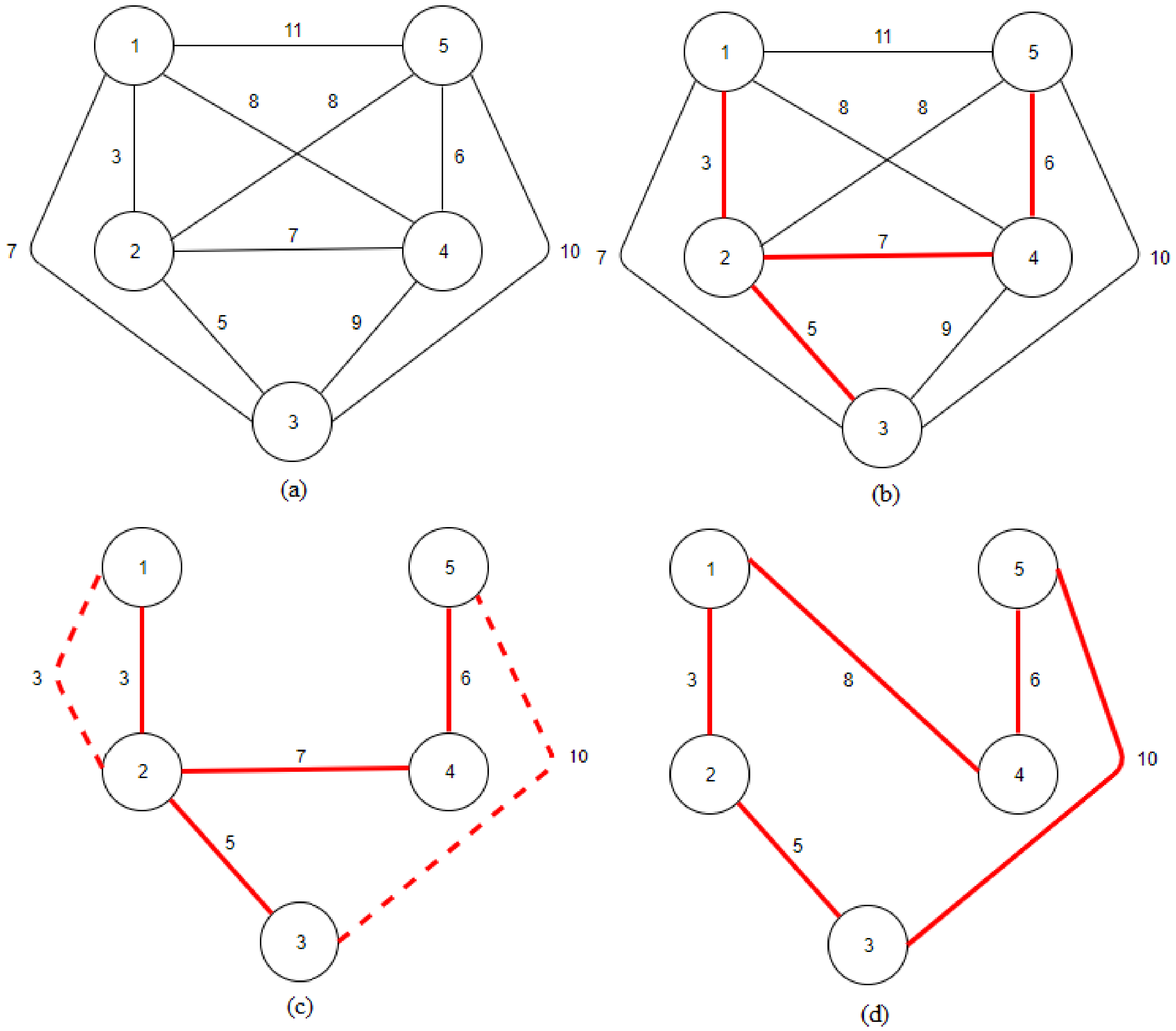
Ο ψευδοκώδικας για τον αλγορίθμο Christofides, παρουσιάζεται στον Πίνακα 6:

```
Αλγόριθμος Christofides (V,w)

    1. ΔΗΜΙΟΥΡΓΙΑ(MST), από G(V,w)
.   2. O =  κορυφές περιττού βαθμού
    3. Από Handshaking Lemma, O = άρτιο αριθμό κορυφών
    4. ΥΠΟΛΟΓΙΣΜΟΣ(M) // Minimum-weight spanning matching
    5. ΔΗΜΙΟΥΡΓΙΑ(H) = κορυφές άρτιου βαθμού
    6. Euler(H) // Δημιουργία μονοπατιού Euler
    7. Hamilton(H) // Δημιουργία τελικού μονοπατιού από το μονοπάτι Euler
```

Πίνακας 6: Αλγόριθμος Christofides

Στην Εικόνα 6, ακολουθεί ένα παράδειγμα, με την γραφική αναπαράσταση του αλγορίθμου, στο οποίο πραγματοποιείται η διάσχιση του γραφήματος, η δημιουργία του ελάχιστου επικαλυπτόμενου δέντρου και ο υπολογισμός του κόστους μεταβάσεων των ακμών, δηλαδή του συνολικού βάρους του κύκλου Hamilton.



Εικόνα 6: Αλγόριθμος Christofides

Εικόνα 6: Εκτέλεση αλγορίθμου Christofides: (a) το πλήρες γράφημα για το πρόβλημα του πλανόδιου πωλητή. Αρχικός κόμβος είναι ο 1 (b) Ακολουθεί η δημιουργία του ελάχιστου επικαλυπτόμενου δέντρου (MST), με έναν συγκεκριμένο αλγόριθμο, στην περίπτωσή μας, είναι ο αλγόριθμος του Kruskal. Παρατηρείται ότι το συνολικό κόστος μετάβασης στις ακμές, ισούται με 21. (c) Το MST με τις επιπλέον ακμές. Στις κορυφές περιττού βαθμού, προστίθενται επιπλέον ακμές έτσι ώστε όλοι οι κόμβοι να τροποποιηθούν, να γίνουν άρτιου βαθμού και να μπορούμε να εφαρμόσουμε στον γράφο, μονοπάτι Euler. (d) Ο τελικός κύκλος Hamilton. Σκοπός είναι η επιστροφή στον αρχικό κόμβο (κόμβος 1). Η μετάβαση σύμφωνα με το σχήμα (c), δηλαδή το μονοπάτι: (1 – 2 – 3 – 5 – 4 – 2 – 1), θα είχε κόστος ίσο με 34. Για το λόγο αυτό, έχουμε μια μικρή τροποποίηση και για την επιστροφή στον κόμβο 1, το μονοπάτι που προκύπτει είναι το (1 – 2 – 3 – 5 – 4 – 1), επιλέγεται δηλαδή η άμεση σύνδεση του κόμβου 4 με τον 1, με βάρος ίσο με 8, οπότε και το συνολικό κόστος μεταβάσεων του κύκλου Hamilton, είναι πλέον 32.

Όσον αφορά την ποιότητα της λύσης που βρίσκει ο αλγόριθμος, έχει ήδη αναφερθεί πως η λύση που προκύπτει είναι το πολύ 3/2 χειρότερη από τη βέλτιστη λύση. Για να αποδειχθεί αυτό, έστω ότι C είναι το βέλτιστο δρομολόγιο για το πρόβλημα του πλανόδιου πωλητή. Αφαιρώντας μια ακμή από αυτό, δημιουργείται ένα επικαλυπτόμενο δέντρο, στο οποίο το βάρος του είναι τουλάχιστον ίσο με το βάρος του ελάχιστου επικαλυπτόμενου δέντρου, δηλαδή $w(T) \leq w(C)$. Στη συνέχεια, μετράμε τις κορυφές του συνόλου O γύρω από το C και χωρίζουμε τις κορυφές αυτές σε δυο μονοπάτια: στο ένα, οι κόμβοι του συγκεκριμένου μονοπατιού είναι περιττού βαθμού και στο άλλο, είναι άρτιου. Κάθε μονοπάτι, αντιστοιχεί σε ένα «καλό ταίριασμα» (perfect matching) των ακμών του συνόλου O, συνδέει τα δύο τελικά σημεία κάθε μονοπατιού και το βάρος του, είναι το πολύ ίσο με το συνολικό βάρος των μονοπατιών. Καθώς τα δύο αυτά σύνολα, διαχωρίζουν τις ακμές του συνόλου C, ένα από τα δυο έχει το πολύ το μισό από το βάρος του C και χάρις στη συνθήκη της τριγωνικής ανισότητας, το βάρος του ταιριάσματος που προκύπτει, θα είναι το πολύ ίσο με το μισό του βάρους του C. Το «ελαχίστου - κόστους καλύτερο ταίριασμα» (minimum-weight perfect matching), δεν μπορεί να έχει μεγαλύτερο βάρος, έτσι προκύπτει ότι: $w(M) \leq w(C)/2$. Προσθέτοντας και τα βάρη των συνόλων M και T, προκύπτει ότι το συνολικό βάρος του μονοπατιού Euler θα είναι το πολύ ίσο με $3w(C)/2$. Λόγω της συνθήκης της τριγωνικής ανισότητας, η μέθοδος shortcutting, η απόρριψη δηλαδή των ακμών που επανεμφανίζονται, δεν αυξάνει το βάρος, και έτσι το συνολικό τελικό βάρος θα ισούται με το πολύ $3w(C)/2$.

3.3 Ο Αλγόριθμος Prim

Όπως αναφέρθηκε και παραπάνω, ο αλγόριθμος του Prim, είναι ένας άπληστος αλγόριθμος, ο οποίος επιθυμεί να βρει την βέλτιστη λύση, κάνοντας μια επιλογή όπου τη δεδομένη χρονική στιγμή φαίνεται η καλύτερη δυνατή και ελπίζοντας πως αυτή, θα τον οδηγήσει στη βέλτιστη λύση. Για το λόγο αυτό, κατασκευάζει ένα Ελάχιστο Επικαλυπτόμενο Δέντρο (Minimum Spanning Tree - MST).

Σε ένα πλήρες δοθέν γράφημα, μη κατευθυνόμενο και με επιβαρυμένες ακμές, το ελάχιστο επικαλυπτόμενο δέντρο θα είναι ένα υποσύνολο των ακμών, το οποίο θα συνδέει όλους τους κόμβους μεταξύ τους, χωρίς να περιλαμβάνει καθόλου κύκλους και με το μικρότερο συνολικά βάρος. Γενικότερα, ένα μη κατευθυνόμενο επιβαρυμένο γράφημα (όχι απαραίτητα συνδεδεμένο), θα περιλαμβάνει ένα ελάχιστο επικαλυπτόμενο δάσος, το οποίο θα ενώνει όλα τα ελάχιστα επικαλυπτόμενα δέντρα μεταξύ τους.

Μια βασική ιδιότητα του αλγορίθμου είναι ότι ακμές του συνόλου (έστω ότι το σύνολό μας είναι E), σχηματίζουν πάντοτε ένα δέντρο ακριβώς. Έτσι, το δέντρο ξεκινά από έναν ριζικό κόμβο r και επεκτείνεται μέχρι να καλύψει όλους τους κόμβους του συνόλου V. Σε κάθε βήμα, προστίθεται στο E μια ελαφριά ακμή ως προς τη σύνδεση του E με κάποιον απομονωμένο κόμβου του γραφήματος $G_E = (V, E)$. Έτσι, όταν ο αλγόριθμος τερματίσει, οι ακμές του συνόλου E συνιστούν ένα ελάχιστο επικαλυπτόμενο δέντρο.

Βασικός παράγοντας για τη γρήγορη υλοποίηση του αλγορίθμου του Prim, είναι να διευκολύνουμε την επιλογή της νέας ακμής που θα προστεθεί στο δέντρο, το οποίο σχηματίζεται από τις ακμές του συνόλου E. Στον ψευδοκώδικα που ακολουθεί, το συνδεδεμένο γράφημα G και ο αρχικός κόμβος r, αποτελούν τα στοιχεία εισόδου του αλγορίθμου. Κατά την εκτέλεσή του, όλοι οι κόμβοι που δεν ανήκουν στο δέντρο, είναι καταχωρισμένοι σε μια ουρά προτεραιότητας ελαχίστου Q, με βάση ένα κλειδί. Για κάθε κόμβο u, το $κλειδί[u]$ ισούται με το ελάχιστο από τα βάρη των ακμών που συνδέουν τον u με κάποιον κόμβο του δέντρου (εάν δεν υπάρχει τέτοια ακμή, τότε κλειδί$[u] = \infty$). Το πεδίο $\pi[u]$, υποδεικνύει τον προκάτοχο του u στο δέντρο. Στη διάρκεια της εκτέλεσης, το σύνολο E της διαδικασίας MST τηρείται ως: $E = \{(u, \pi[u]) : u \in V - \{r\} - Q\}$.

Όταν ο αλγόριθμος τερματίσει, η ουρά προτεραιότητας ελαχίστου Q θα είναι κενή΄ επομένως, το Ελάχιστο Επικαλυπτόμενο Δέντρο E για το G είναι: $E = \{u, \pi[u]) : u \in V - \{r\}\}$

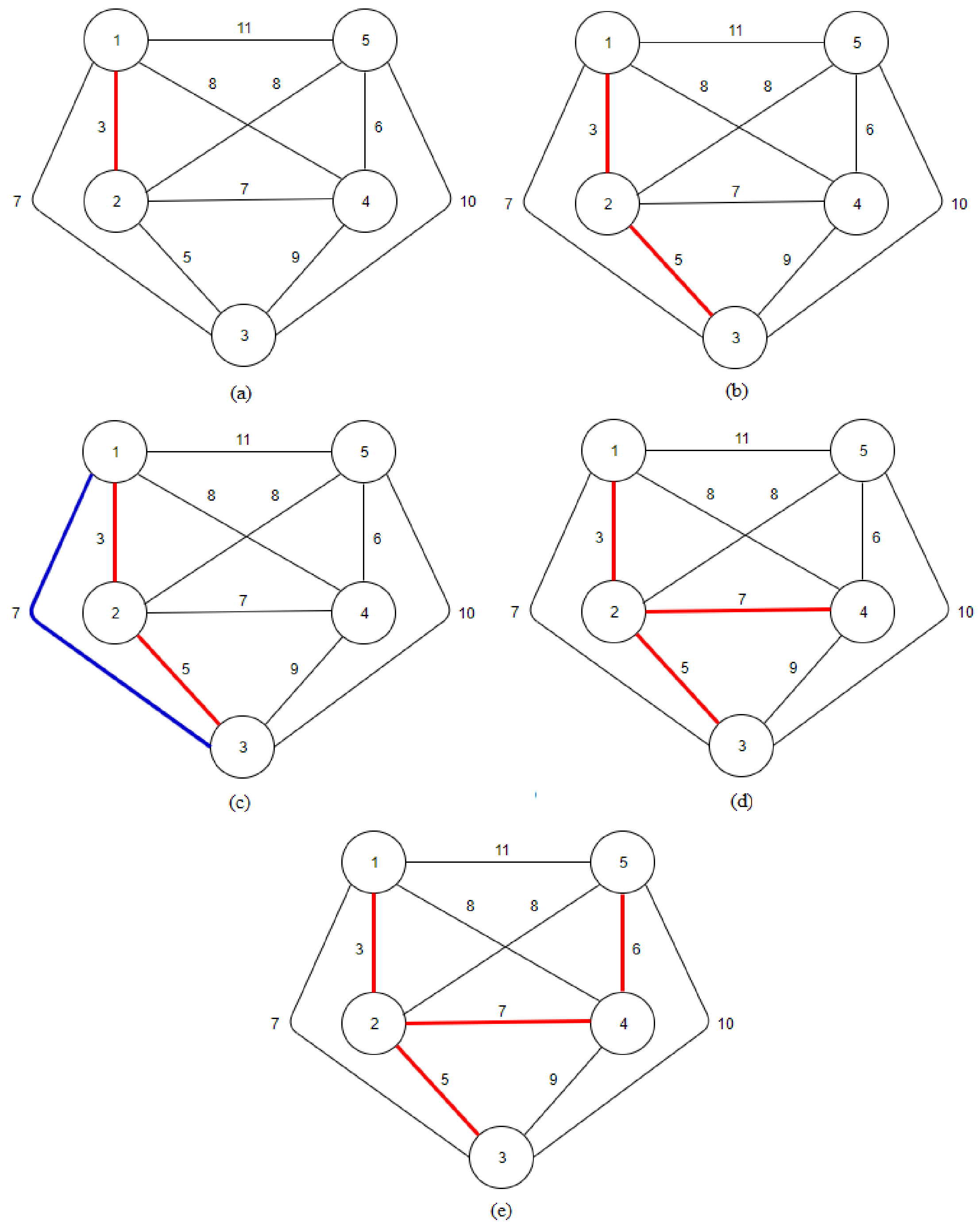
Τα βήματα του άπληστου αλγορίθμου Prim, παρουσιάζονται συνοπτικά στον Πίνακα 7:

Αλγόριθμος Prim(G,w,r)

1. για κάθε $u \in V[G]$
2. $\quad κλειδί[u] = \infty$
3. $\quad \pi[u] = KENO$
4. $κλειδί[r] = 0$
5. $Q = V[G]$
6. Όσο $Q \neq KENOY$
7. $\quad$ u <- ΕΞΑΓΩΓΗ ΕΛΑΧΙΣΤΟΥ(Q)
8. $\quad$ για κάθε $u \in Adj[u]$
9. $\qquad$ αν $v \in Q$ & $w(u, v) < κλειδί(u)$
10. $\qquad\quad$ τότε $\pi[v] = u$
11. $\qquad\qquad κλειδί[v]$

Πίνακας 7: Αλγόριθμος Prim

Στην Εικόνα 7, η οποία ακολουθεί στη συνέχεια, παριστάνονται γραφικά τα βήματα που πραγματοποιούνται, για την εύρεση του κατάλληλου μονοπατιού, κατά την εκτέλεση του αλγόριθμου Prim. Έχει γίνει η παραδοχή ότι αρχικός κόμβος αποτελεί ο 1, και σκοπός είναι η διάσχιση του γράφου με το μικρότερο δυνατό κόστος και η δημιουργία του Ελάχιστου Επικαλυπτόμενου Δέντρου.



Εικόνα 7: Αλγόριθμος Prim

Στην Εικόνα 7, παρουσιάζεται η λειτουργία του αλγορίθμου Prim: (a) Εκκίνηση από τον κόμβο 1 και επίσκεψη στο κόμβο 2, όπου το βάρος της ακμής, είναι το μικρότερο. (b) Μετάβαση στον κόμβο 3, με το μικρότερο βάρος ακμής, τόσο από τον κόμβο 1 αλλά και από τον κόμβο 2. (c) Τυχαία επιλογή επίσκεψης στον κόμβο 1, καθώς εδώ έχουμε δύο ακμές όπου τα βάρη τους είναι ίσα (ακμές 1-3 & 2-4 με κόστος μετάβασης 7). Απόρριψη της συγκεκριμένης ακμής, ο κόμβος 1, έχει ήδη επισκεφθεί. (d) Επίσκεψη στον κόμβο 4 με κόστος 7. (e) Μετάβαση στον κόμβο με το αριθμό 5. Αποτελεί τον τελευταίο κόμβο επίσκεψης, ο αλγόριθμος του Prim, τερματίζει. Το συνολικό κόστος που προκύπτει, μετά την διάσχιση του γράφου και της κατασκευής του ελάχιστου επικαλυπτόμενου δέντρου είναι: 21.

Η επίδοση του αλγορίθμου του Prim, εξαρτάται από τον τρόπο υλοποίησης της ουράς προτεραιότητας Q. Εάν η Q υλοποιηθεί ως δυαδικός σωρός ελαχίστου, η απόδοση των αρχικών τιμών στις γραμμές 1-5, μπορεί να επιτευχθεί σε χρόνο O(V). Ο βρόχος «όσο» εκτελείται V φορές, και γνωρίζοντας ότι κάθε πράξη ΕΞΑΓΩΓΗΣ ΕΛΑΧΙΣΤΟΥ διαρκεί $O(logV)$, ο συνολικός χρόνος για όλες τις κλήσεις είναι $O(VlogV)$. Ο βρόχος «για» εκτελείται $O(E)$ φορές, καθώς το άθροισμα των μηκών όλων των λιστών γειτνίασης είναι $2|E|$. Ο έλεγχος για το εάν υπάρχει ένας κόμβος στην ουρά, πραγματοποιείται σε σταθερό χρόνο και η ανάθεση τιμής στην γραμμή 11, υλοποιείται σε χρόνο $O(logV)$, σε ένα δυαδικό σωρό ελαχίστου. Άρα, ο συνολικός χρόνος που απαιτείται για την εκτέλεση του αλγορίθμου του Prim είναι $O(ElogV)$.

Ο χρόνος εκτέλεσης του αλγόριθμου μπορεί ωστόσο να αλλάξει και να μειωθεί αρκετά, εάν στην υλοποίηση συμπεριλάβουμε σωρούς Fibonacci. Εάν προσθέσουμε V στοιχεία σε ένα σωρό Fibonacci, μπορούμε να εκτελέσουμε την πράξη ΕΞΑΓΩΓΗΣ ΕΛΑΧΙΣΤΟΥ σε χρόνο $O(logV)$. Συνεπώς, αν υλοποιήσουμε τη σειρά προτεραιότητας ελαχίστου Q της διαδικασίας με σωρό Fibonacci, τότε ο συνολικός χρόνος εκτέλεσης του αλγορίθμου του Prim θα μειωθεί και θα ισούται με $O(E + VlogV)$.

3.4 Ο Αλγόριθμος Kruskal

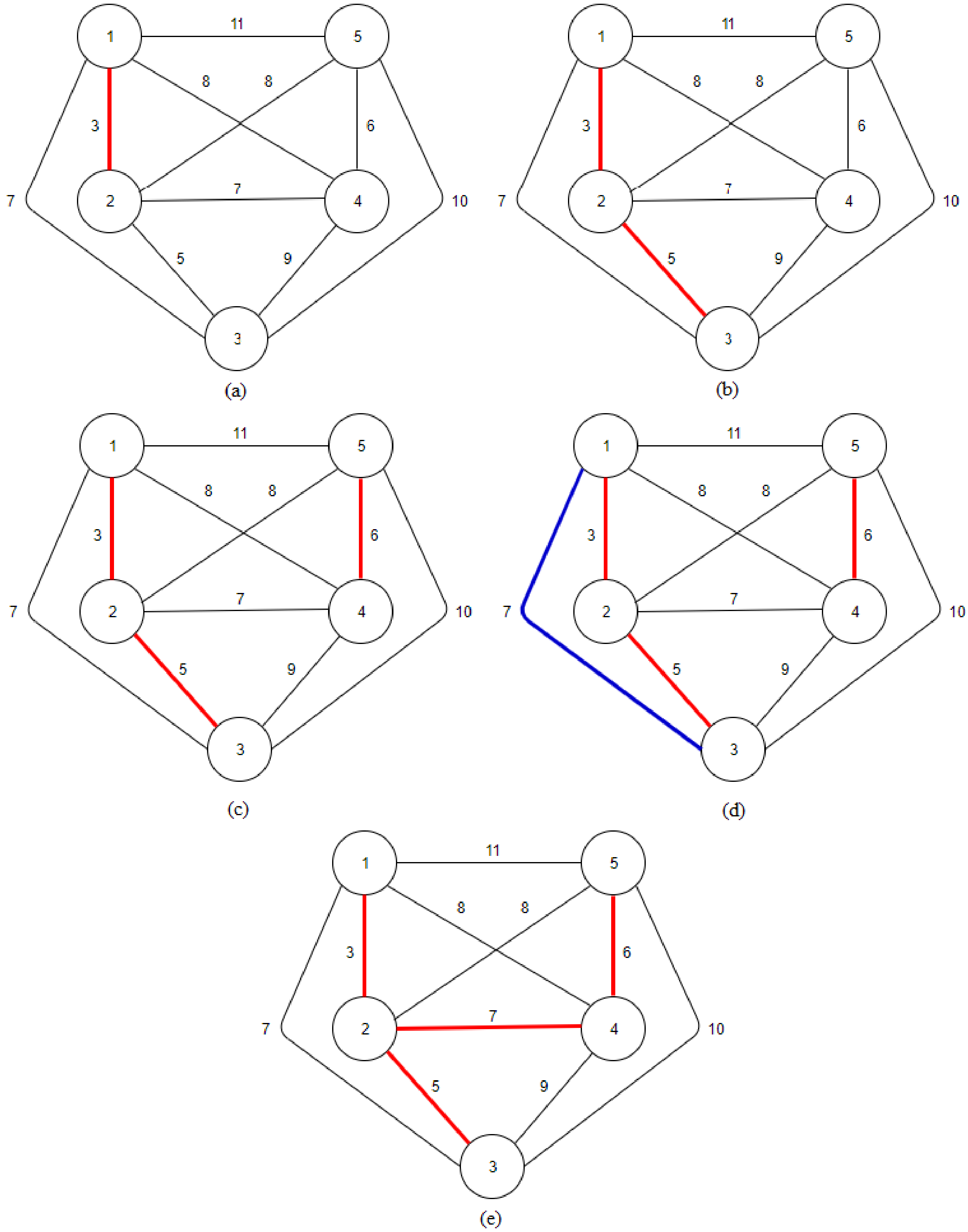
Ο αλγόριθμος του Kruskal ανήκει και αυτός στην κατηγορία των άπληστων αλγορίθμων. Βασίζεται στη δημιουργία ενός MST, όπως συμβαίνει και με τον αλγόριθμο Prim. Προσδιορίζει την ακμή που θα προστεθεί στο δάσος, εντοπίζοντας από όλες τις ακμές που συνδέουν οποιαδήποτε δέντρα του δάσους, την ακμή (u, v) με το μικρότερο βάρος. Η υλοποίηση βασίζεται στην τήρηση κάποιων ξένων συνόλων, από στοιχεία μιας δομής ξένων συνόλων. Κάθε σύνολο, περιέχει τους κόμβους ενός δέντρου του τρέχοντος δάσους. Η πράξη ΕΥΡΕΣΗ ΣΥΝΟΛΟΥ(u), επιστρέφει ένα στοιχείο του συνόλου στο οποίο ανήκει ο κόμβος u. Συνεπώς, μπορούμε να προσδιορίσουμε αν δυο κόμβοι u και v ανήκουν στο ίδιο δέντρο, ελέγχοντας αν το στοιχείο ΕΥΡΕΣΗ ΣΥΝΟΛΟΥ(u) είναι ίδιο με το στοιχείο που προκύπτει από την ΕΥΡΕΣΗ ΣΥΝΟΛΟΥ(v). Η ένωση των δύο συνόλων, πραγματοποιείται με την πράξη ΣΥΝΕΝΩΣΗ.Ο Πίνακας 8, παρουσιάζει τα βήματα του αλγορίθμου:

```
Αλγόριθμος Kruskal(G,w)

1. A = ΚΕΝΟ
2. για κάθε κόμβο v∈V[G]
3.        ΚΑΤΑΣΚΕΥΗ ΣΥΝΟΛΟΥ(v)
4. ταξινόμηση των ακμών του E κατά αύξουσα σειρά ως
   προς το βάρος w
5. για κάθε ακμή (u, v)∈E, κατά αύξουσα σειρά ως προς το
   βάρος w
6.        αν η ΕΥΡΕΣΗ ΣΥΝΟΛΟΥ(u) ≠ ΕΥΡΕΣΗ
          ΣΥΝΟΛΟΥ(v)
7.                τότε A = A∪{(u, v)}
8.                        ΣΥΝΕΝΩΣΗ(u,v)
9. επιστροφή A
```

Πίνακας 8: Αλγόριθμος Kruskal

Στην Εικόνα 8 που ακολουθεί, παρουσιάζεται η γραφική αναπαράσταση του αλγορίθμου του Kruskal.



Εικόνα 8: Αλγόριθμος Kruskal

(a) Ξεκινάει η διάσχιση και η δημιουργία του MST, επιλέγοντας την ακμή με το μικρότερο βάρος (ακμή 1-2 με βάρος 3). (b) Η αμέσως μεγαλύτερη σε κόστος ακμή είναι η 2-3 με 5. (c) Ακολουθεί η ακμή 4-5 με βάρος 6, προστίθεται στο έως τώρα MST. (d) Στο επόμενο βήμα, επιλέγεται τυχαία μία εκ των ακμών 1-3 και 2-4, καθώς διαθέτουν το ίδιο βάρος. Η ακμή 1-3, δεν είναι διαθέσιμη για το MST, καθώς πραγματοποιείται κύκλος, για αυτό απορρίπτεται. (e) Έτσι, επιλέγεται η ακμή 2-4, έχουμε επισκεφτεί όλους τους διαθέσιμους κόμβους και το MST έχει κατασκευαστεί. Ο αλγόριθμος, τερματίζει. Το συνολικό κόστος μετάβασης, μετά την λειτουργία του αλγορίθμου Kruskal είναι: 21 (όμοιος με τον αλγόριθμο του Prim).

Ο χρόνος εκτέλεσης του αλγορίθμου Kruskal για ένα γράφημα $G(V,E)$, εξαρτάται από την υλοποίηση της δομής δεδομένων ξένων συνόλων. Υποθέτουμε ότι έχουμε υλοποίηση δάσους ξένων συνόλων, δεδομένου ότι είναι η ασυμπτωτικά ταχύτερη γνωστή υλοποίηση. Ο αρχικός ορισμός του συνόλου, απαιτεί χρόνο $O(1)$ και η ταξινόμηση των ακμών στην γραμμή 4, απαιτεί O(ElogE). Ο βρόχος «για» στις γραμμές 5 ως 8 εκτελεί $O(E)$ πράξεις ΕΥΡΕΣΗΣ ΣΥΝΟΛΟΥ & ΣΥΝΕΝΩΣΗΣ στο δάσος ξένων συνόλων. Μαζί με τις $|V|$ πράξεις ΚΑΤΑΣΚΕΥΗΣ ΣΥΝΟΛΟΥ, οι εργασίες αυτές απαιτούν χρόνο $O((V+E)a(V))$, με a την βραδύτατα αύξουσα συνάρτηση (ο χρόνος εκτέλεσης m πράξεων n στοιχείων είναι $O(ma(n))$. Εφόσον έχουμε υποθέσει ότι το γράφημα G είναι συνδεδεμένο, έχουμε $|E| \geq V-1$ και επομένως, οι πράξεις ξένων συνόλων θα απαιτούν χρόνο $O(Ea(V))$. Επειδή $a(|V|) = O(logV) = O(logE)$, ο συνολικός χρόνος εκτέλεσης του αλγορίθμου του είναι $O(ElogE)$. Ξέροντας ότι $|E| \leq |V|2$, έχουμε $log|E| = O(log|V|)$ και φτάνουμε στο συμπέρασμα ότι ο τελικός χρόνος, θα ισούται με $O(ElogV)$.

3.5 Ο Αλγόριθμος Πλησιέστερου Γείτονα

Ο τελευταίος άπληστος αλγόριθμος που εξετάστηκε στην παρούσα εργασία, είναι αυτός του Πλησιέστερου Γείτονα (Nearest Neighbor). Η ιδέα για το επόμενο βήμα, προκύπτει πάντοτε από την επίσκεψη της επόμενης μη-επισκέπτουσας πλησιέστερης πόλης. Αποτελεί ένα από τα πιο απλά και εύκολα παραδείγματα αλγορίθμου, για το πρόβλημα του πλανόδιου πωλητή.

Η υλοποίηση του, βασίζεται σε ένα πίνακα γειτνίασης (A), που περιλαμβάνει τις αποστάσεις μεταξύ όλων των κόμβων. Σαν είσοδο, δέχεται επίσης το σύνολο V, που περιλαμβάνει όλους τους κόμβους v. Ο αλγόριθμος, μέσω της συνάρτησης ΑΠΟΣΤΑΣΗ(), υπολογίζει κάθε φορά για ένα κόμβο v_m, τη μικρότερη σε βάρος ακμή του και την προσθέτει στο σύνολο V, που περιλαμβάνει όλους τους κόμβους τους οποίους έχουμε ήδη επισκεφθεί. Ο αλγόριθμος τερματίζει την λειτουργία του, όταν επισκεφτούμε όλους τους διαθέσιμους κόμβους και επιστρέψουμε στην αρχική μας θέση. Η Εικόνα 9, δείχνει αναλυτικά τα βήματα:

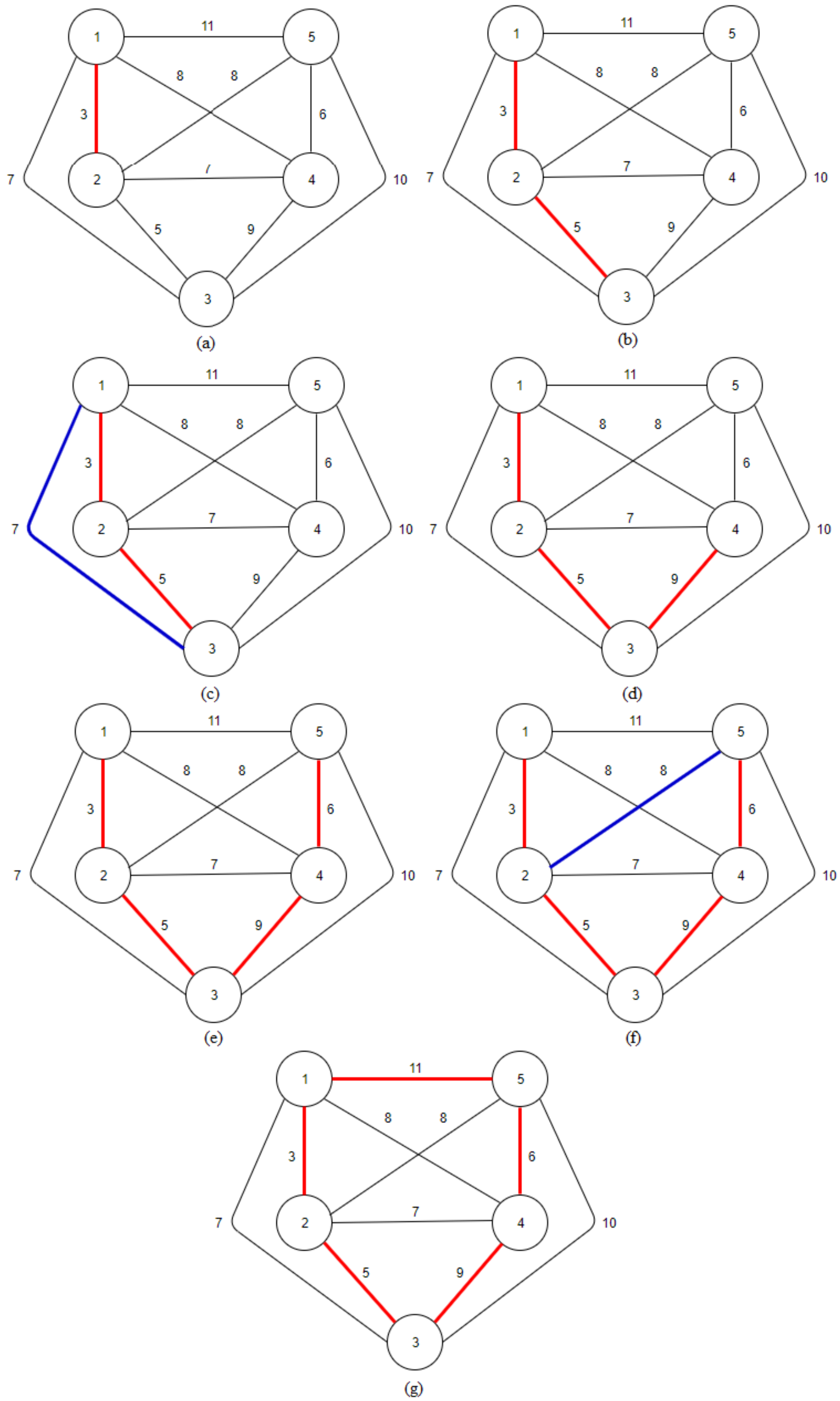
Αλγόριθμος Πλησιέστερου Γείτονα (A,V)

1. $V_1 = \{v_1\}$ // ο πρώτος κόμβος των στοιχείων
2. $V = \{V_1\}$
3. $v = 1$
4. για i = 1 ως V
5. βρες το στοιχείο v_m στο σύνολο V ώστε η ΑΠΟΣΤΑΣΗ$(v_i, v_m) = \min$

6.	αν ΑΠΟΣΤΑΣΗ$(v_i, v_m) < t$ τότε
7.	$V_m = V_m \cup v_i$
8.	αλλιώς
9.	$v = v + 1$
10.	$V_v = \{v_i\}$

Πίνακας 9: Αλγόριθμος Πλησιέστερου Γείτονα

Στην Εικόνα 9, παρουσιάζεται η γραφική απεικόνιση της διάσχισης γράφου για το πρόβλημα των 5 κόμβων που μελετούμε, με τον αλγόριθμο του Πλησιέστερου Γείτονα. Υποθέτουμε ότι ο αρχικός κόμβος, για την δημιουργία του κατάλληλου μονοπατιού, είναι ο 1 (a) Μετάβαση στον κόμβο 2, με το μικρότερο βάρος (κόστος μετάβασης 3). (b) Επίσκεψη στον κόμβο 3, με βάρος ακμής 5. (c) Μετάβαση στον κόμβο 1 (πλησιέστερος), με κόστος 7. Απορρίπτεται η συγκεκριμένη μετάβαση, ο κόμβος 1 έχει ήδη επισκεφθεί. Επίσκεψη στον αμέσως επόμενο κόμβο. (d) Πλησιέστερος γείτονας του 3 είναι ο 4, με βάρος ακμής 9, τοποθετείται και αυτός στους κόμβους που έχουμε επισκεφθεί. (e) Μετάβαση στον κόμβο 5 με κόστος 6. (f) Πλησιέστερος γείτονας του κόμβου 5 αποτελεί ο κόμβος 2. Απορρίπτεται διότι τον συγκεκριμένο κόμβο τον έχουμε ήδη επισκεφθεί. (g) Επίσκεψη του κόμβου 1 με το βάρος της ακμής να ισούται με 11. Ο αλγόριθμος τερματίζει την λειτουργία του. Το συνολικό βάρος των ακμών, μετά την διάσχιση του δέντρου και επιστρέφοντας στην αρχική μας θέση είναι: 34.



Εικόνα 9: Αλγόριθμος Πλησιέστερου Γείτονα

Όσον αφορά την πολυπλοκότητα του αλγορίθμου, οι αναθέσεις των τιμών στις γραμμές 1-3, απαιτούν χρόνο $O(1)$. Στην συνέχεια, στις γραμμές 4 έως και 10, ακολουθεί η επαναληπτική διαδικασία, η οποία θα πραγματοποιηθεί V φορές (όσοι είναι και οι διαθέσιμοι κόμβοι στο παράδειγμά μας). Η συνολική πολυπλοκότητα που προκύπτει, για την συγκεκριμένη υλοποίηση του αλγορίθμου Πλησιέστερου Γείτονα είναι $O(V^2)$.

3.6 Ο Αλγόριθμος 2-Opt

Στην κατηγορία των αλγορίθμων επαναληπτικής βελτίωσης, που ονομάζονται αλλιώς ευρετικές διαδικασίες τοπικής αναζήτησης (local search heuristics), ανήκει και ο αλγόριθμος 2-opt. Σε προβλήματα βελτιστοποίησης, αποτελεί τον πιο απλό αλγόριθμο τοπικής αναζήτησης, ο οποίος παρουσιάστηκε από τον Croes το 1958, παρόλο που αρχικά είχε προταθεί από τον Flood το 1956, για την επίλυση του προβλήματος του πλανόδιου πωλητή. Η κεντρική ιδέα είναι να επιλεχθεί ένα δρομολόγιο το οποίο διασταυρώνεται με τον εαυτό του και να μεταρρυθμιστεί, έτσι ώστε να πάψει να συμβαίνει το προηγούμενο φαινόμενο.

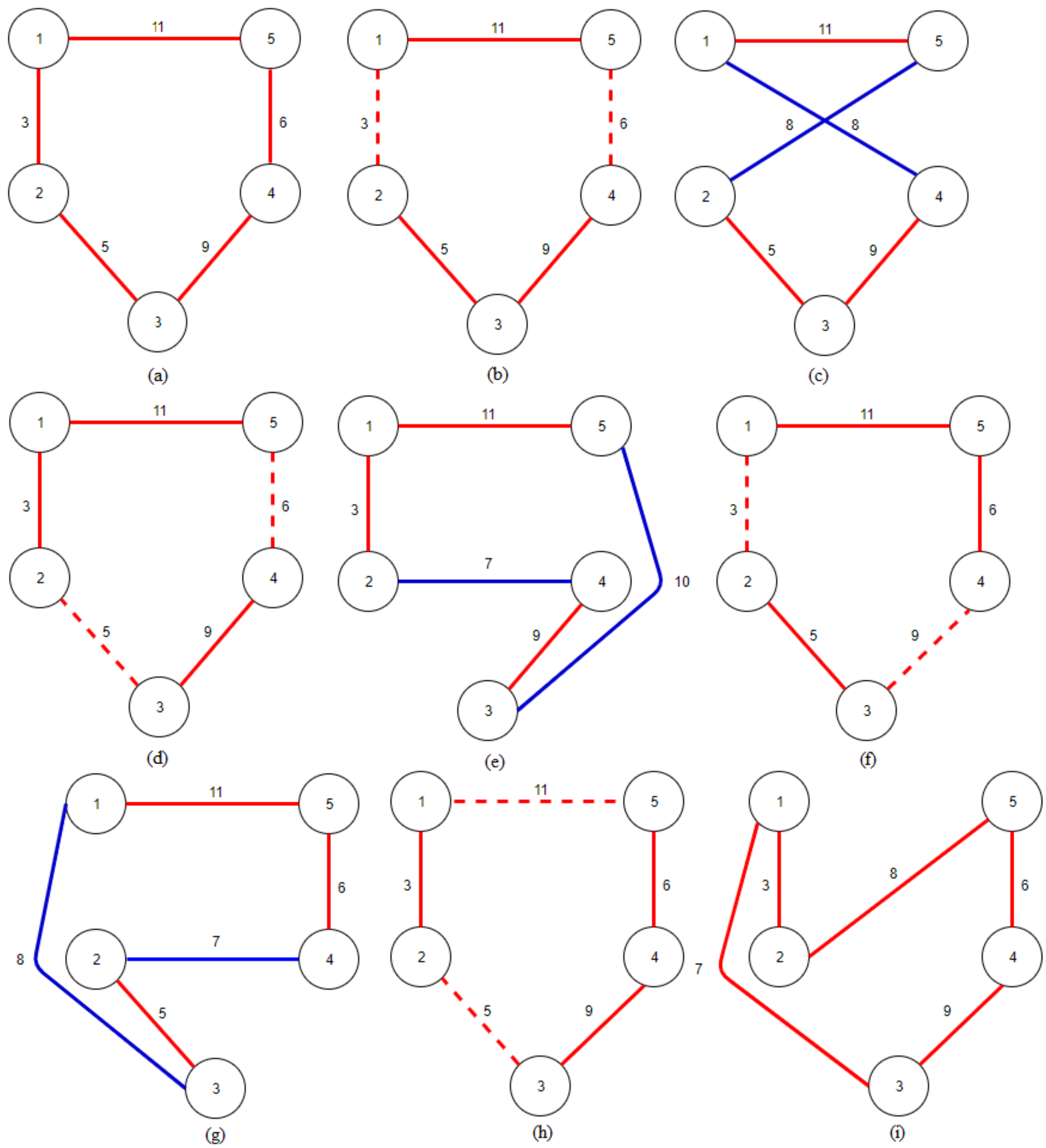
Μια πλήρης τοπική αναζήτηση με τον αλγόριθμο 2-opt, θα συγκρίνει κάθε έγκυρο πιθανό συνδυασμό ακμών, μέσω ενός μηχανισμού ανταλλαγής. Η τεχνική αυτή εφαρμόζεται στο πρόβλημα του πλανόδιου πωλητή, καθώς επίσης και σε άλλα προβλήματα που προκύπτουν από αυτό.

Ο ψευδοκώδικας του αλγορίθμου, ο οποίος δέχεται σαν είσοδο τους κόμβους του γραφήματος και τις συντεταγμένες x και y αυτών, παρουσιάζεται και περιγράφεται συνοπτικά στον Πίνακα 10. Ακολουθεί η γραφική αναπαράσταση εκτέλεσης του αλγορίθμου 2-Opt, η οποία φαίνεται μέσω της Εικόνας 10.

Αλγόριθμος 2-Opt (V,x,y)

1. Υπολογισμός πίνακα αποστάσεων // όλες οι πιθανές αποστάσεις
2. ΕΥΡΕΣΗ(Τ) // μονοπάτι από την εκτέλεση του ΝΝ αλγορίθμου
3. για $i = 1$ ως $i = n - 2$
4.   για $j = i + 2$ ως $j = n$
5. d_1 = συνολικό μήκος 2 ακμών
6. d_2 = νέο συνολικό μήκος των ακμών, όταν ΑΝΤΑΛΛΑΓΗ()
7.   αν $(d_1 > d_2)$ ΤΟΤΕ
8.   ΑΝΤΑΛΛΑΓΗ(d_1, d_2) // κόμβοι στο δρομολόγιο Τ

Πίνακας 10: Αλγόριθμος 2-Opt



Εικόνα 10: Αλγόριθμος 2-Opt

(a) Σε αυτό το σχήμα, παρουσιάζεται το MST, το οποίο έχει προκύψει με τον αλγόριθμου του Πλησιέστερου Γείτονα, με κόστος μετάβασης σε όλες τις ακμές 34. Στη διπλανή σελίδα, πραγματοποιούνται οι κατάλληλες εναλλαγές στις ακμές, με σκοπό την εύρεση μονοπατιού χαμηλότερου κόστους. (b) Τα πρώτα ζεύγη ακμών που εξετάζονται (1-2 με βάρος 3, 4-5 με βάρος 6). (c) Σύμφωνα με το προηγούμενο σχήμα, οι νέες ακμές που θα προκύψουν είναι οι 1-4 με βάρος 8 & 2-5 με επίσης βάρος 8. Απορρίπτονται, καθώς το κόστος που προκύπτει εδώ είναι 41. (d) Επόμενες ακμές προς εξέταση: 2-3 με κόστος 5 & 4-5 με κόστος 6. (e) Μετά τις εναλλαγές, οι νέες ακμές που δημιουργήθηκαν είναι: 2-4 με βάρος 7 και 3-5 με βάρος 10. Και σε αυτή την περίπτωση έχουμε απόρριψη των ακμών, το βάρος που προκύπτει ισούται με 40 > 34. (f) Προς εξέταση οι ακμές 1-2 και 3-4 με βάρη 3 και 9 αντίστοιχα. (g) Οι ακμές που προκύπτουν εδώ είναι: 1-3 (κόστος 7) και 2-4 (κόστος επίσης 7). (g) Το βάρος που προκύπτει ισούται με $37 > 34$, συνεπώς απορρίπτεται. (h) Τελευταίες ακμές που εξετάζονται είναι η 2-3 με βάρος 5 και η 1-5 με βάρος 11. (j) Οι νέες ακμές που προκύπτουν στην προκειμένη περίπτωση είναι η 1-3, κόστος μετάβασης 7 και η 2-5 με κόστος 8. Υπολογίζοντας το νέο κόστος του μονοπατιού, προκύπτει ότι είναι ίσο με $32 < 34$, οπότε αποτελεί και το νέο μονοπάτι Hamilton.

Όταν συζητάμε περί πολυπλοκότητας σε k-opt αλγορίθμους, παραλείπουμε να αναφέρουμε πως για μία εντολή, απαιτείται $O(n)$ χρόνος για να εκτελεστεί. Μια απλοϊκή υλοποίηση του αλγορίθμου 2-opt, θέλει $O(n^2)$ χρόνο για την εκτέλεσή του, που περιλαμβάνει την επιλογή μιας ακμής, (έστω v_1, v_2, ακμή μεταξύ των κόμβων 1 και 2) και την αναζήτηση μιας άλλης ακμής (v_3, v_4). Ο υπολογισμός αυτής της εντολής, θα πραγματοποιηθεί μονάχα, αν οι αποστάσεις των κόμβων $(v_1 v_2) \& (v_3, v_4)$, είναι μεγαλύτερες από τις $(v_2, v_3), (v_1, v_4)$, δηλαδή, έστω ότι διατίθεται μια συνάρτηση $dist$, υπεύθυνη για τον υπολογισμό αποστάσεων μεταξύ των κόμβων, οπότε $dist(v_1, v_2) + dist(v_3, v_4) > dist(v_2, v_3) + dist(v_1, v_4)$.

3.7 Ο Αλγόριθμος Εξαντλητικής Αναζήτησης

Μία πολύ σημαντική κατηγορία αλγορίθμων, είναι αυτή των αλγορίθμων εξαντλητικής αναζήτηση. Πιο συγκεκριμένα, θα μελετήσουμε την συμπεριφορά του αλγορίθμου Brute Force. Σύμφωνα με αυτή, το πρόβλημα επιλύεται με τον πιο απλό τρόπο, ο οποίος σχεδόν σε όλες του τις περιπτώσεις, δεν είναι ο πλέον αποδοτικός. Επειδή ακριβώς προσπαθεί να δώσει λύση στα προβλήματα που καλείται να αντιμετωπίσει με μια απλή προσέγγιση, είναι πολύ εύκολο να υλοποιηθεί και να εκτελεστεί. Χαρακτηριστικό παράδειγμα αυτής της κατηγορίας, αποτελεί η ταξινόμηση μια δομής, έστω ενός πίνακα, με την μέθοδο bubble-sort, όπως επίσης και η linear search. Σαν απόδειξη του πόσο αργοί είναι οι αλγόριθμοι αυτής της ομάδας, η μέθοδος bubble-sort, έχει πολυπλοκότητα της τάξης $O(n^2)$, ενώ άλλες τεχνικές ταξινόμησης, για παράδειγμα αυτή της merge sort, έχουν πολυπλοκότητα $O(nlogn)$.

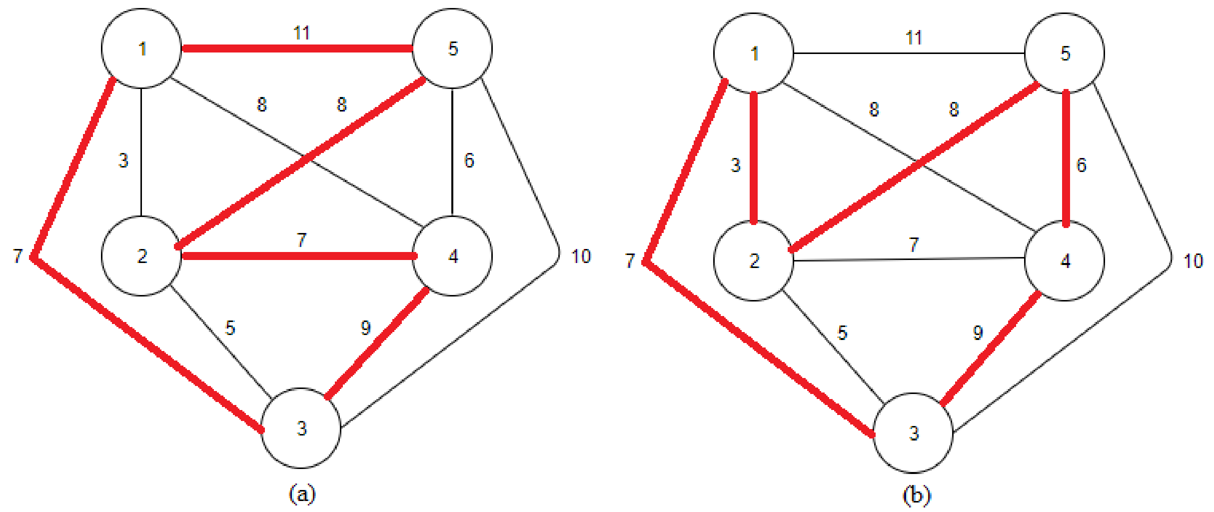
Έτσι λοιπόν, αν μελετήσουμε τη συμπεριφορά του αλγορίθμου, θα παρατηρήσουμε ότι απαιτεί πολύ μεγάλο χρόνο εκτέλεσης για να υπολογίσει το κατάλληλο μονοπάτι, ειδικά σε προβλήματα με αρκετούς κόμβους και κατ' επέκταση, με περισσότερες πιθανές συνδέσεις. Ενδεικτικά, για το πρόβλημα πέντε κόμβων που παρουσιάστηκε και μελετήθηκε παραπάνω, εξετάστηκαν 120 πιθανά μονοπάτια του γράφου (5! μονοπάτια).

Ο Πίνακας 11, αναλύει τα βήματα εκτέλεσης του αλγορίθμου Brute Force, για να ανακαλύψει τη μικρότερη δυνατή απόσταση, ανάμεσα σε δύο σημεία. Σαν είσοδο δέχεται το σύνολο P, που περιλαμβάνει n κόμβους με τις συντεταγμένες τους και επιστρέφει τους δύο κόμβους με την μικρότερη ακμή. Επίσης, στην Εικόνα 11, παρουσιάζονται δύο από το 120 πιθανά μονοπάτια, κατά την εκτέλεση του συγκεκριμένου αλγορίθμου.

Αλγόριθμος Brute Force (P)
1. $dmin = \infty$
2. για $i = 0$ ως $i = n - 1$
3. για $j = i$ ως n
4. $d = sqrt(x_i - x_j)^2 + (y_i - y_j)^2$
5. αν $d < dmin$
6. $dmin = d$

7. $index1 = i, index2 = j$
8. Επιστροφή index1, index2

Πίνακας 11: Αλγόριθμος Brute Force



Εικόνα 11: Αλγόριθμος Brute Force – Εύρεση Μονοπατιών

Κεφάλαιο 4ο

Υπολογιστικά Αποτελέσματα

4.1 Η Βιβλιοθήκη TSPLIB

Η βιβλιοθήκη TSPLIB (2019), δημιουργήθηκε στο πανεπιστήμιο Heidelberg της Γερμανίας και αποτελείται από δείγματα περιπτώσεων για το πρόβλημα του Πλανόδιου Πωλητή, καθώς επίσης και για άλλα σχετικά προβλήματα. Η δομή των αρχείων περιλαμβάνει τρεις στήλες, στις οποίες αποτυπώνονται αρχικά το id των πόλεων των οποίων καλούμαστε να επισκεφτούμε ακριβώς μια φορά, και στη συνέχεια οι συντεταγμένες x και y αυτών, αντίστοιχα. Υπάρχει πληθώρα προβλημάτων, διαφόρων μεγεθών, από προβλήματα με μόλις 52 κόμβους (berlin52.tsp), μέχρι και προβλήματα μεγαλύτερων διαστάσεων (fnl4461.tsp), τα οποία προκύπτουν από την αναζήτηση και εύρεση του ελάχιστου σε μήκος μονοπατιού, ανάμεσα σε πραγματικές πόλεις της Γερμανίας. Πέραν των δεδομένων που υπάρχουν διαθέσιμα, διατίθενται επίσης και οι πιο γνωστές λύσεις που έχουν ανακαλυφθεί, δηλαδή τα βέλτιστα μονοπάτια για το συμμετρικό πρόβλημα του Πλανόδιου Πωλητή. Τα προβλήματα τα οποία μελετήθηκαν στην παρούσα διπλωματική εργασία, από την βιβλιοθήκη TSPLIB φαίνονται στον Πίνακα 12.

Πίνακας 12: Προβλήματα TSPLIB

eil51	lin105	pr226	pr439
berlin52	ch130	pr264	rat783
eil76	ch150	a280	pr1002
rat99	kroA150	pr299	nrw1379
kroB100	kroB200	lin318	

Πέραν όμως αυτών, δημιουργήθηκαν και εξετάστηκαν, άλλα 6 προβλήματα, ίδιας δομής με τα παραπάνω, διότι για τη μελέτη του αλγορίθμου Brute Force και τον υπολογισμό όλων των πιθανών μονοπατιών θα χρειαζόταν υπερβολικά μεγάλος χρόνος. Ενδεικτικά, για το μικρότερο πρόβλημα της βιβλιοθήκης TSPLIB, που αποτελείται από 51 διαθέσιμους κόμβους, ο αλγόριθμος Brute Force, θα έπρεπε να βρει το βέλτιστο μονοπάτι ανάμεσα σε 51! = 1.5511188e+66 πιθανά μονοπάτια, καθιστώντας την εύρεσή του πραγματικά αδύνατη. Έτσι λοιπόν, δημιουργήθηκαν 6 νέα προβλήματα, από 5 έως 10 κόμβους, για την μελέτη του συγκεκριμένου αλγορίθμου. Αυτά παρουσιάζονται στον Πίνακα 13:

Πίνακας 13: Προβλήματα Γεννήτριας Τυχαίων Αριθμών

rng5	rng7	rng9
rng6	rng8	rng10

Σε όλα τα προβλήματα των Πινάκων 12 και 13, υπολογίζονται, η τιμή που προκύπτει για το κάθε μονοπάτι, δηλαδή το κόστος των μεταβάσεων ανάμεσα σε όλους τους κόμβους, η σειρά επίσκεψης των κόμβων αυτών κατά την εκτέλεση των αλγορίθμων, και τέλος, οι χρόνοι που προέκυψαν, τόσο οι συνολικοί αλλά και οι χρόνοι για τη δημιουργία του MST, για τους αλγορίθμους Prim, Kruskal & Christofides. Ωστόσο, για υλοποιήσεις αλγορίθμων και μεγάλα προβλήματα, όπου ο χρόνος εκτέλεσης τους ξεπερνούσε την μία ώρα, επιλέχθηκε να σταματήσει η διαδικασία.

Οι υλοποιήσεις των αλγορίθμων, πραγματοποιήθηκαν στη γλώσσα προγραμματισμού C, χρησιμοποιώντας διαφορετικές δομές δεδομένων για κάθε αλγόριθμο. Για παράδειγμα, στην υλοποίηση του αλγορίθμου Prim, χρησιμοποιήθηκε σωρός ελαχίστου. Όσον αφορά τον αλγόριθμο Nearest Neighbor, εκεί χρησιμοποιήθηκε μια απλούστερη δομή δεδομένων, οι πίνακες.

4.2 Αποτελέσματα για τα Προβλήματα που Δημιουργήθηκαν από την Γεννήτρια Τυχαίων Αριθμών

Πρόβλημα rng5

Το συγκεκριμένο πρόβλημα, αποτελεί το μικρότερο που μελετήθηκε στην παρούσα διπλωματική εργασία, και τα αποτελέσματα εκτέλεσης όλων των αλγορίθμων, παρουσιάζονται στον παρακάτω πίνακα. Κατά την υλοποίηση του αλγορίθμου Brute Force, εξετάστηκαν 5! = 120 πιθανά μονοπάτια και υπολογίστηκε η ελάχιστη τιμή του, δηλαδή η βέλτιστη. Εδώ παρατηρείται ότι ο αλγόριθμος του Kruskal, καθώς επίσης και αυτός του Πλησιέστερου Γείτονα, ανακάλυψαν την ίδια λύση.

Πίνακας 14: rng5

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Συνολικοί Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	512.63	1 5 4 2 3 1	0.0010	0.0002
Kruskal	512.63	1 5 4 3 2 1	0.0078	0.0002
NN	512.63	1 5 4 2 3 1	0.0010	-
Christofides	512.63	1 5 4 2 3 1	0.0010	0.0002
2-Opt	510.78	1 2 3 4 5 1	0.0011	-
Brute - Force	506.91	5 1 4 2 3 5	0.0100	-

Πρόβλημα rng6

Οι διαθέσιμοι κόμβοι στο παρόν πρόβλημα, είναι 6 όπως προκύπτει και από την ονομασία του. Τα συγκεντρωτικά αποτελέσματα, παρατίθενται στην επόμενη σελίδα. Ο αλγόριθμος Brute Force, αναζήτησε την βέλτιστη λύση, ανάμεσα σε 720 πιθανά μονοπάτια (=6!).

Πίνακας 15: rng6

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	1350.82	1 5 4 6 2 3 1	0.0010	0.00018
Kruskal	1350.82	1 5 4 6 2 3 1	0.0086	0.00020
NN	1349.10	1 5 4 2 6 3 1	0.0010	-
Christofides	1350.82	1 5 4 6 2 3 1	0.0010	0.00019
2-Opt	1239.53	3 4 2 6 1 5 3	0.0010	-
Brute - Force	1179.45	3 5 1 6 2 4 3	0.0120	-

Πρόβλημα rng7

Μελετήθηκαν οι αλγόριθμοι του Πίνακα 16, για το πρόβλημα 7 διαθέσιμων κόμβων. Ο Brute Force αλγόριθμος, ανακάλυψε 7! = 5040 πιθανά μονοπάτια.

Πίνακας 16: rng7

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	152.50	1 7 3 2 6 5 4 1	0.0020	0.006
Kruskal	144.02	1 7 3 6 5 2 4 1	0.0171	0.005
NN	156.85	1 4 7 2 6 5 3 1	0.0010	-
Christofides	152.50	1 7 3 2 6 5 4 1	0.0100	0.006
2-Opt	135.63	3 2 4 1 7 5 6 3	0.0010	-
Brute - Force	127.80	5 7 1 4 2 3 6 5	0.0130	-

Πρόβλημα rng8

Εδώ, εξετάστηκε η εύρεση λύσης, ανάμεσα σε 8 κόμβους – πόλεις και με την υλοποίηση του Brute Force, ανακαλύφθηκαν 8! = 40320 πιθανές διαδρομές. Τα αποτελέσματα, παρουσιάζονται στον Πίνακα 17.

Πίνακας 17: rng8

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	9761.63	1 8 4 7 5 2 3 6 1	0.0010	0.0005
Kruskal	9761.63	1 8 4 7 5 2 3 6 1	0.0235	0.0090
NN	9488.88	1 8 4 6 3 2 7 5 1	0.0020	-
Christofides	9761.63	1 8 4 7 5 2 3 6 1	0.0220	0.0130
2-Opt	8917.44	4 5 7 8 1 6 3 2 4	0.0020	-
Brute - Force	8681.07	3 6 1 8 7 5 4 2 3	0.0130	-

Πρόβλημα rng9

Για 9 κόμβους, προκύπτουν 9! = 362880 πιθανά μονοπάτια. Το βέλτιστο, παρουσιάζεται παρακάτω, κατά την εκτέλεση του Brute Force. Στο παράδειγμα αυτό, παρατηρείται ότι και ο αλγόριθμος 2 – opt, κατάφερε να υπολογίσει την βέλτιστη λύση.

Πίνακας 18: rng9

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	9421.67	1 9 4 7 8 6 2 3 5 1	0.0010	0.0008
Kruskal	9421.67	1 9 4 7 8 6 2 3 5 1	0.0168	0.0080
NN	9612.99	1 5 3 7 4 8 6 2 9 1	0.0020	-
Christofides	9421.67	1 9 4 7 8 6 2 3 5 1	0.0200	0.0110
2-Opt	8908.16	3 5 1 9 2 6 8 4 7 3	0.0020	-
Brute - Force	8908.16	3 5 1 9 2 6 8 4 7 3	0.0140	-

Πρόβλημα rng10

Για την εύρεση του βέλτιστου μονοπατιού κατά την εκτέλεση του Brute Force, ανακαλύφθηκαν 36228800 = 10! διαθέσιμα πιθανά μονοπάτια.

Πίνακας 19: rng10

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	1268.57	1 8 9 5 4 3 6 7 10 2 1	0.0020	0.0011
Kruskal	1275.31	1 8 9 7 10 2 6 3 4 5 1	0.0339	0.0110
NN	1237.19	1 8 9 10 7 2 3 6 5 4 1	0.0010	-
Christofides	1268.57	1 8 9 5 4 3 6 7 10 2 1	0.0193	0.0120

2-Opt	1197.18	8 9 10 7 6 5 4 3 2 1 8	0.0020	-
Brute - Force	1158.18	8 1 2 3 4 5 6 7 10 9 8	0.0160	-

4.3 Αποτελέσματα για τα Προβλήματα της Βιβλιοθήκης TSPLIB

Πρόβλημα eil51

Το μικρότερο σε έκταση πρόβλημα που μελετήθηκε στην διπλωματική εργασία, προερχόμενο από την βιβλιοθήκη TSPLIB, αποτελεί το eil51 γνωρίζοντας ότι η βέλτιστη λύση, ισούται με 426. Παρατίθεται ο σχετικός πίνακας με τις μετρήσεις που πραγματοποιήθηκαν:

Πίνακας 20: eil51

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	545.25	1 22 32 11 38 5 9 50 21 29 2 16 34 30 49 10 39 4 18 14 25 13 41 19 40 42 44 15 45 33 37 17 47 12 46 51 27 48 23 24 7 43 36 35 20 3 28 31 8 26 6 1	0.061	0.020
Kruskal	522.66	1 22 32 11 38 5 9 50 21 29 2 16 34 30 49 10 39 36 35 20 3 28 31 8 26 48 27 51 46 12 47 4 18 17 37 15 45 33 44 42 19 40 41 13 25 14 6 23 24 7 43 1	0.059	0.018
NN	513.60	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	0.128	-
Christofides	522.66	1 22 32 11 38 5 9 50 21 29 2 16 34 30 49 10 39 36 35 20 3 28 31 8 26 48 27 51 46 12 47 4 18 17 37 15 45 33 44 42 19 40 41 13 25 14 6 23 24 7 43 1	0.137	0.029
2-Opt	454.42	47 4 17 37 15 33 45 44 42 19 40 41 13 25 24 43 23 7 26 31 28 3 36 35 20 29 21 34 30 39 10 49 9 50 16 11 38 5 12 46 51 27 1 32 2 22 8 48 6 14 18	0.151	-

Πρόβλημα berlin52

Άλλο ένα σχετικά μικρό πρόβλημα που μελετήθηκε, αποτελεί το berlin52, το οποίο όπως προδίδει και η ονομασία του, περιλαμβάνει 52 διαθέσιμους κόμβους – πόλεις. Για αυτό το πρόβλημα, γνωρίζουμε ότι η βέλτιστη λύση είναι ίση με 7542.

Πίνακας 21: berlin52

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος Εκτέλεσης (sec)	Χρόνος για Δημιουργία MST (sec)
Prim	8848.33	1 49 32 45 19 41 8 10 9 33 43 15 5 24 48 46 47 26 27 13 52 14 28 12 51 11 25 4 6 38 40 37 39 36 35 34 44 16 50 29 20 23 30 2 7 42 21 31 18 3 17 22 1	0.085	0.015
Kruskal	8848.33	1 49 32 45 19 41 8 10 9 33 43 15 5 24 48 46 47 26 27 13 52 14 28 12 51 11 25 4 6 38 40 37 39 36 35 34 44 16 50 29 20 23 30 2 7 42 21 31 18 3 17 22 1	0.176	0.024
NN	8980.91	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	0.133	-
Christofides	8848.33	1 49 32 45 19 41 8 10 9 33 43 15 5 24 48 46 47 26 27 13 52 14 28 12 51 11 25 4 6 38 40 37 39 36 35 34 44 16 50 29 20 23 30 2 7 42 21 31 18 3 17 22 1	0.120	0.033
2-Opt	8494.71	15 40 39 49 32 45 10 9 8 41 19 3 17 18 31 21 42 7 2 30 29 47 26 27 13 14 52 11 28 12 51 33 43 4 25 46 16 50 20 23 22 1 44 34 35 36 37 38 48 24 5 6	0.176	-

Πρόβλημα eil76

Το πρόβλημα αυτό, αναζητεί λύση μεταξύ 76 πόλεων. Από την βιβλιοθήκη TSPLIB, γνωρίζουμε ότι η τιμή του βέλτιστου μονοπατιού είναι: 538. Τα αποτελέσματα που προέκυψαν κατά την εκτέλεση των αλγορίθμων, παρατίθενται στον Πίνακα 22.

Πίνακας 22: eil76

Αλγόριθμος	Λύση	Μονοπάτι	Χρόνος Εκτέλεσης (sec)	Χρόνος για Δημιουργία MST (sec)
Prim	645.30	1 73 33 63 56 23 16 49 24 55 25 50 18 32 44 3 9 39 72 58 10 31 38 65 11 66 53 7 35 19 54 13 27 59 14 8 46 34 67	0.261	0.053

		76 75 4 68 6 51 17 40 12 26 52 45 29 48 47 21 36 69 71 60 70 20 37 5 57 15 30 2 74 28 62 22 61 64 42 41 43 1		
Kruskal	645.30	1 73 33 63 56 23 16 49 24 55 25 50 18 32 44 3 9 39 72 58 10 31 38 65 11 66 53 7 35 19 54 13 27 59 14 8 46 34 67 76 75 4 68 6 51 17 40 12 26 52 45 29 48 47 21 36 69 71 60 70 20 37 5 57 15 30 2 74 28 62 22 61 64 42 41 43 1	0.258	0.059
NN	684.66	1 73 33 63 16 51 6 68 75 76 67 26 12 40 17 3 44 32 9 39 72 58 10 38 65 11 66 14 53 35 7 8 46 34 52 27 45 29 5 37 20 70 60 71 36 47 21 48 30 74 28 62 2 4 13 54 19 59 57 15 69 61 22 42 41 43 23 56 49 24 18 50 25 55 31 64	0.111	-
Christofides	645.30	1 73 33 63 56 23 16 49 24 55 25 50 18 32 44 3 9 39 72 58 10 31 38 65 11 66 53 7 35 19 54 13 27 59 14 8 46 34 67 76 75 4 68 6 51 17 40 12 26 52 45 29 48 47 21 36 69 71 60 70 20 37 5 57 15 30 2 74 28 62 22 61 64 42 41 43 1	0.329	0.056
2-Opt	582.65	46 34 4 75 76 67 26 17 51 16 3 44 49 24 18 55 25 50 32 40 12 9 39 31 72 58 10 38 65 66 11 59 14 53 35 7 8 19 54 13 57 15 20 70 60 71 69 36 37 47 21 61 28 74 2 62 22 64 42 41 43 56 23 63 1 73 33 6 68 30 48 5 29 45 27 52	0.192	-

Πρόβλημα rat99

Για το συγκεκριμένο πρόβλημα, γνωρίζουμε ότι η τιμή της βέλτιστης λύσης σύμφωνα με την βιβλιοθήκη TSPLIB είναι 1211.

Πίνακας 23: rat99

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	1380.97	1 10 2 3 12 11 13 22 23 24 25 26 35 27 36 44 34 45 54 53 52 51 60 61 62 63 72 81 80 71 70 69 78 79 96 97 98 99 90 89 88 87 86 95 85 76 84 83 82 93 94 92 91 73 74 75 77 68 67 66 65 64 56 57 58 59 49 50 55 46 47 48 39 38 40 41 32 42 43 33 31 30 29 37 28 19 20 21 15 16 17 18 9 8 7 6 5 14 4 1	1.022	0.189
Kruskal	1380.97	1 10 2 3 12 11 13 22 23 24 25 26 35 27 36 44 34 45 54 53 52 51 60 61 62 63 72 81 80 71 70 69 78 79 96 97 98 99 90 89 88 87 86 95 85 76 84 83 82	0.980	0.175

		93 94 92 91 73 74 75 77 68 67 66 65 64 56 57 58 59 49 50 55 46 47 48 39 38 40 41 32 42 43 33 31 30 29 37 28 19 20 21 15 16 17 18 9 8 7 6 5 14 4 1		
NN	1561.90	1 2 3 12 11 10 20 19 28 37 29 30 31 32 41 40 39 48 38 47 46 55 56 57 58 59 60 51 52 53 54 45 44 34 43 42 33 24 25 26 35 36 27 17 18 9 8 7 6 5 14 4 13 22 21 23 15 16 50 49 61 62 63 72 71 70 69 78 77 76 85 84 83 82 91 92 93 94 95 86 87 88 79 80 81 89 90 99 98 97 96 75 74 65 66 67 68 64 73	0.143	-
Christofides	1380.97	1 10 2 3 12 11 13 22 23 24 25 26 35 27 36 44 34 45 54 53 52 51 60 61 62 63 72 81 80 71 70 69 78 79 96 97 98 99 90 89 88 87 86 95 85 76 84 83 82 93 94 92 91 73 74 75 77 68 67 66 65 64 56 57 58 59 49 50 55 46 47 48 39 38 40 41 32 42 43 33 31 30 29 37 28 19 20 21 15 16 17 18 9 8 7 6 5 14 4 1	1.255	0.193
2-Opt	1260.55	71 70 69 80 79 78 77 76 84 85 86 87 88 89 90 99 98 97 96 95 94 93 92 91 83 82 73 74 75 68 67 66 65 64 56 57 55 46 47 38 48 39 40 41 42 43 34 33 32 31 30 29 37 28 19 10 1 2 3 4 5 6 7 8 9 18 17 16 15 14 13 12 11 20 21 22 23 24 25 26 27 35 36 44 45 54 53 52 51 50 49 58 59 60 61 62 63 72 81	0.205	-

Πρόβλημα kroB100

Σε αυτό το πρόβλημα, περιλαμβάνονται 100 διαθέσιμες κορυφές, και αναζητείται τιμή κοντά στη βέλτιστη λύση, η οποία δεν είναι άλλη από την 22141.

Πίνακας 24: kroB100

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	24763.75	1 95 98 26 100 56 81 79 47 65 37 72 88 22 23 55 50 77 24 18 16 2 13 78 17 41 45 36 96 92 19 44 51 48 31 63 42 14 64 82 33 15 6 4 83 66 74 60 87 89 43 52 54 58 84 7 57 34 94 61 35 27 46 25 9 71 12 32 59 76 99 29 8 97 91 28 3 11 93 85 73 53 70 39 40 67 5 62 69 75 30 80 20 38 49 86 68 10 21 90 1	0.480	0.1480
Kruskal	24763.75	1 95 98 26 100 56 81 79 47 65 37 72 88 22 23 55 50 77 24 18 16 2 13 78 17	0.485	0.1530

		41 45 36 96 92 19 44 51 48 31 63 42 14 64 82 33 15 6 4 83 66 74 60 87 89 43 52 54 58 84 7 57 34 94 61 35 27 46 25 9 71 12 32 59 76 99 29 8 97 91 28 3 11 93 85 73 53 70 39 40 67 5 62 69 75 30 80 20 38 49 86 68 10 21 90 1		
NN	29190.40	1 95 98 12 71 27 61 35 94 57 34 7 84 58 52 54 43 89 87 60 74 66 4 83 6 15 33 64 14 42 2 13 78 17 45 36 96 92 19 44 41 18 24 77 16 50 55 22 23 88 37 72 65 79 81 47 56 100 26 69 62 5 67 40 39 70 53 73 85 93 11 3 28 91 97 99 8 29 32 59 76 90 21 86 49 30 75 80 20 38 46 25 9 68 10 82 48 31 63 51	0.142	-
Christofides	24763.75	1 95 98 26 100 56 81 79 47 65 37 72 88 22 23 55 50 77 24 18 16 2 13 78 17 41 45 36 96 92 19 44 51 48 31 63 42 14 64 82 33 15 6 4 83 66 74 60 87 89 43 52 54 58 84 7 57 34 94 61 35 27 46 25 9 71 12 32 59 76 99 29 8 97 91 28 3 11 93 85 73 53 70 39 40 67 5 62 69 75 30 80 20 38 49 86 68 10 21 90 1	0.601	0.0982
2-Opt	24538.80	24 96 19 44 92 36 41 31 48 63 13 78 17 45 18 16 2 42 14 64 82 51 15 33 6 83 4 66 87 89 43 52 58 84 7 34 57 60 74 94 35 61 27 71 9 25 46 90 21 38 20 80 30 49 86 68 10 95 1 12 98 32 59 29 8 99 76 97 91 28 3 11 93 85 73 53 70 39 40 67 5 62 56 100 26 69 75 79 81 47 65 37 72 54 88 23 22 55 50 77	0.207	-

Πρόβλημα lin105

Για το πρόβλημα lin105, γνωρίζουμε μέσω της βιβλιοθήκης TSPLIB, ότι η τιμή για το βέλτιστο μονοπάτι που έχει ανακαλυφθεί, είναι 14379 μονάδες.

Πίνακας 25: lin105

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	17736	1 2 6 7 10 11 15 103 12 20 21 22 29 30 31 32 33 28 23 27 24 19 16 18 17 25 26 36 37 41 44 47 51 54 57 58 53 52 46 43 42 48 49 104 40 45 50 55 56 59 105 62 69 74 75 81 89 90 99 98 93 92 102 101 97 96 91 85 83 84 79 86 77 72 73 80 76 70 82 78 71 68 67 64 63 61 60 65 66 87 88 94 95 100 39 38 35 34 14 13 5 4 9 8 3 1	1.298	0.190

Kruskal	18463.46	1 2 6 7 10 11 15 103 12 20 21 22 29 30 31 32 100 95 94 88 87 66 65 61 60 63 62 69 74 75 81 89 90 99 98 93 92 102 101 97 96 91 85 83 84 79 86 77 72 73 80 76 70 82 78 71 68 67 64 105 56 59 55 50 48 45 49 44 47 51 54 57 58 53 52 46 43 42 41 40 104 33 28 23 24 27 19 16 18 17 25 26 36 37 39 38 35 34 14 13 5 4 9 8 3 1	1.193	0.192
NN	20362.76	1 2 6 7 10 11 15 103 21 22 29 30 31 32 28 23 20 19 24 27 26 25 18 16 17 9 8 3 12 33 36 37 42 43 46 52 53 58 57 54 51 47 44 41 104 40 45 48 50 55 56 59 105 62 63 70 74 75 81 69 73 76 80 79 77 72 68 67 71 78 82 83 84 85 86 89 90 99 98 93 92 91 96 97 101 102 64 49 39 38 35 34 14 13 5 4 60 61 65 66 87 88 94 95 100	0.146	-
Christofides	18463.46	1 2 6 7 10 11 15 103 12 20 21 22 29 30 31 32 100 95 94 88 87 66 65 61 60 63 62 69 74 75 81 89 90 99 98 93 92 102 101 97 96 91 85 83 84 79 86 77 72 73 80 76 70 82 78 71 68 67 64 105 56 59 55 50 48 45 49 44 47 51 54 57 58 53 52 46 43 42 41 40 104 33 28 23 24 27 19 16 18 17 25 26 36 37 39 38 35 34 14 13 5 4 9 8 3 1	1.703	0.245
2-Opt	14966.97	79 77 72 64 63 62 105 59 56 55 54 57 58 53 52 46 43 42 41 44 47 51 50 48 45 49 40 104 36 37 26 25 18 17 16 19 24 27 33 32 31 30 29 22 21 103 28 23 20 12 15 11 10 7 6 2 1 3 8 9 5 4 13 14 34 35 38 39 60 61 65 66 87 88 94 95 100 99 98 102 101 97 96 92 91 93 89 90 81 75 74 70 69 73 76 80 86 84 85 83 82 78 71 68 67	0.210	-

Πρόβλημα ch130

Το πρόβλημα σε αυτή τη περίπτωση, εξετάζει την εύρεση λύσης ανάμεσα σε 130 διαθέσιμους κόμβους, γνωρίζοντας ότι η βέλτιστη λύση που προκύπτει, ισούται με 6110.

Πίνακας 26: ch130

Αλγόριθμος	**Λύση**	**Μονοπάτι**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	7346.26	1 41 39 117 112 105 62 115 28 128 16 45 76 11 5 109 61 129 124 64 69 88 86 102 6 55 122 47 37 22 93 20 46	2.582	0.361

		118 80 18 114 126 121 78 66 85 125 90 59 30 83 3 108 8 21 33 13 67 96 14 10 23 40 44 42 51 60 120 53 49 91 72 58 106 38 92 73 99 74 75 52 9 57 65 56 82 101 123 111 119 84 36 32 113 25 48 98 68 63 70 97 26 7 127 107 104 43 19 27 34 17 31 100 116 95 79 12 87 81 103 77 94 89 110 29 15 24 4 35 54 2 50 130 71 1		
Kruskal	7347.64	1 41 39 117 112 105 62 115 28 128 16 45 76 11 5 109 61 129 124 64 69 88 86 102 6 55 122 40 47 37 22 93 20 46 118 80 18 114 126 121 78 66 85 125 90 59 30 83 3 108 8 21 33 13 67 96 14 10 23 44 42 51 60 120 53 49 91 72 58 106 38 92 73 99 74 75 52 9 57 65 56 82 101 123 111 119 84 36 32 113 25 48 98 68 63 70 97 26 7 127 107 104 43 19 27 34 17 31 100 116 95 79 12 87 81 103 77 94 89 110 29 15 24 4 35 54 2 50 130 71 1	2.604	0.347
NN	7575.28	1 41 39 71 130 50 2 118 80 46 20 35 54 17 31 34 27 19 100 116 24 15 29 95 79 38 106 58 49 53 120 60 51 42 44 40 47 37 22 23 122 55 14 10 67 96 13 33 21 18 8 108 114 3 83 30 59 121 78 90 125 85 66 28 115 62 105 112 117 128 16 45 76 109 61 129 124 26 97 70 7 63 68 98 110 89 94 77 103 81 87 12 65 56 57 9 52 75 74 99 73 92 72 91 6 102 93 4 126 5 11 64 69 88 86 127 107 104 43 48 25 32 113 36 84 119 111 123 101 82	0.167	-
Christofides	7346.26	1 41 39 117 112 105 62 115 28 128 16 45 76 11 5 109 61 129 124 64 69 88 86 102 6 55 122 47 37 22 93 20 46 118 80 18 114 126 121 78 66 85 125 90 59 30 83 3 108 8 21 33 13 67 96 14 10 23 40 44 42 51 60 120 53 49 91 72 58 106 38 92 73 99 74 75 52 9 57 65 56 82 101 123 111 119 84 36 32 113 25 48 98 68 63 70 97 26 7 127 107 104 43 19 27 34 17 31 100 116 95 79 12 87 81 103 77 94 89 110 29 15 24 4 35 54 2 50 130 71 1	2.597	0.351
2-Opt	6594.85	108 8 126 30 59 121 78 90 125 85 66 28 115 62 112 105 128 16 45 5 11 76 109 61 129 124 64 69 86 88 26 7 97 70 63 68 98 48 25 32 36 119 84 113 110 89 94 77 81 87 12 103 111 123	0.242	-

		101 82 57 9 56 65 52 75 74 99 73 92 38 106 53 58 49 72 91 120 51 60 6 102 10 14 67 13 96 122 55 23 47 40 22 37 93 44 42 4 35 54 17 34 31 27 100 116 79 95 24 29 15 19 43 104 107 127 1 41 130 2 118 80 46 20 33 21 18 50 71 39 117 83 3 114		

Πρόβλημα ch150

Η βέλτιστη λύση για το δοθέν πρόβλημα, σύμφωνα με τον βιβλιοθήκη TSPLIB είναι 6528. Πραγματοποιείται και εδώ, αναζήτηση μεταξύ 150 διαθέσιμων κόμβων και προκύπτει το κατάλληλο μονοπάτι. Για λόγους απλότητας και εμφάνισης, επιλέγεται να μην παρασταθούν τα μονοπάτια που προκύπτουν κατά την εκτέλεση των αλγορίθμων, τόσο σε αυτό όσο και στα επόμενα προβλήματα, τα οποία είναι μεγαλύτερων διαστάσεων.

Πίνακας 27: ch150

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	7559.09	4.366	0.558
Kruskal	7559.09	4.376	0.543
NN	8331.44	0.192	-
Christofides	7559.09		0.531
2-Opt	7037.49	0.324	-

Πρόβλημα kroA150

Πρόβλημα το οποίο μελετά την εύρεση λύσης και υπολογίζει τις ευκλείδειες αποστάσεις, μεταξύ 150 κόμβων – πόλεων, γνωρίζοντας ότι η τιμή της βέλτιστης λύσης είναι 26524.

Πίνακας 28: kroA150

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	32279.96	8.239	0.983
Kruskal	32279.96	8.248	1.023
NN	34130.46	0.170	-
Christofides	32279.96	8.238	1.059
2-Opt	28664.48	0.337	-

Πρόβλημα kroB200

Σε αυτό το πρόβλημα, αναζητείται λύση σε ένα πλήρες γράφημα, το οποίο περιλαμβάνει 200 διαθέσιμους κόμβους – πόλεις. Η βέλτιστη λύση που προκύπτει, με βάση το εργαλείο TSPLIB, ισούται με 29437.

Πίνακας 29: kroB200

Αλγόριθμος	Λύση	Χρόνος Εκτέλεσης (sec)	Χρόνος για Δημιουργία MST (sec)
Prim	34076.84	28.03	3.651
Kruskal	34414.82	21.52	3.027
NN	36599.51	0.216	-
Christofides	34076.84	26.61	4.558
2-Opt	31162.99	0.403	-

Πρόβλημα pr226

Για το συγκεκριμένο πρόβλημα, όπως προκύπτει και από το όνομά του, περιλαμβάνει 226 διαθέσιμες κορυφές και γνωρίζουμε ότι το βέλτιστο μονοπάτι που προκύπτει, έχει τιμή ίση με 80369.

Πίνακας 30: pr226

Αλγόριθμος	Λύση	Χρόνος Εκτέλεσης (sec)	Χρόνος για Δημιουργία MST (sec)
Prim	98794.58	25.35	4.322
Kruskal	96744.80	29.46	4.854
NN	94685.46	0.228	-
Christofides	96776.07	28.07	4.302
2-Opt	87715.18	0.359	-

Πρόβλημα pr264

Για το πρόβλημα pr264, το οποίο μελετά την εύρεση του βέλτιστου μονοπατιού ανάμεσα σε 264 κόμβους – πόλεις, γνωρίζουμε ότι το βέλτιστο μονοπάτι ισούται με 49135.

Πίνακας 31: pr264

Αλγόριθμος	Λύση	Χρόνος Εκτέλεσης (sec)	Χρόνος για Δημιουργία MST (sec)
Prim	58259.23	76.05	10.21
Kruskal	53991.80	36.45	6.129
NN	58022.85	0.268	-
Christofides	53857.50	34.69	5.974
2-Opt	55646.98	0.402	-

Πρόβλημα a280

Όπως προκύπτει και από την ονομασία του, το πρόβλημα a280 αναζητεί λύση μεταξύ 280 διαθέσιμων κόμβων. Σύμφωνα με την βιβλιοθήκη TSBLIB, η τιμή της βέλτιστης λύσης για το πρόβλημα αυτό είναι 2579.

Πίνακας 32: a280

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	2998.02	85.81	15.79
Kruskal	3089.65	121.4	20.72
NN	3242.99	0.310	-
Christofides	3089.65	117.2	18.33
2-Opt	2768.24	0.425	-

Πρόβλημα pr299

Με τις υλοποιήσεις των αλγορίθμων που μελετήθηκαν ως τώρα, αναζητείται λύση και σε αυτό το πρόβλημα, έχοντας υπ' όψιν ότι το βέλτιστο μονοπάτι έχει τιμή 48191.

Πίνακας 33: pr299

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	55639.90	181.6	33.04
Kruskal	55396.64	180.6	34.57
NN	59900.40	0.338	-
Christofides			32.26
2-Opt	53188.81	0.761	-

Πρόβλημα lin318

Στο παρόν πρόβλημα, εξετάζεται η εύρεση λύσης σύμφωνα με τους αλγόριθμους που μελετήθηκαν ως τώρα, μεταξύ 318 κόμβων – πόλεων. Είναι γνωστό, ότι η τιμή της βέλτιστης λύσης που προκύπτει, ισούται με 42029.

Πίνακας 34: lin318

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	51060.09	190.4	39.11
Kruskal	49245.03	275.5	50.81
NN	54033.58	0.572	-
Christofides	49245.03	256.5	47.18
2-Opt	45654.48	1.060	-

Πρόβλημα pr439

Το πρόβλημα αυτό, περιλαμβάνει την αναζήτηση και εύρεση της βέλτιστης λύσης, ανάμεσα σε 439 διαθέσιμους κόμβους. Όπως προκύπτει από την βιβλιοθήκη TSPLIB, η βέλτιστη λύση για αυτό το πρόβλημα είναι ίση με 107217. Όπως έχει ήδη αναφερθεί, παραλείπονται να εμφανιστούν στα προβλήματα αυτών των μεγεθών, τα μονοπάτια που προκύπτουν κατά την εκτέλεση των αλγορίθμων.

Πίνακας 35: pr439

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**	**Χρόνος για Δημιουργία MST (sec)**
Prim	130576.81	1201	127.5
Kruskal	130128.07	1012	96.08
NN	131282.10	0.597	-
Christofides	130128.07	1038	105.4
2-Opt	112031.91	1.475	-

Πρόβλημα rat783

Εδώ, η βέλτιστη λύση, γνωρίζουμε ότι έχει τιμή 8806, μετά και την επίσκεψη όλων των κόμβων και την δημιουργία του μονοπατιού. Παρατίθεται κα ο Πίνακας 36, με τις τιμές των λύσεων που προέκυψαν, καθώς και οι αντίστοιχοι χρόνοι εκτέλεσης των αλγορίθμων.

Πίνακας 36: rat783

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**
Prim	-	Time-out
Kruskal	-	Time-out
NN	11149.48	0.652
Christofides	-	Time-out
2-Opt	9935.51	16.90

Πρόβλημα pr1002

Σε αυτή την περίπτωση, εξετάζεται και αναζητείται λύση ανάμεσα σε 1002 διαθέσιμους κόμβους – πόλεις, υπολογίζοντας τις αποστάσεις μεταξύ αυτών και εμφανίζοντας το κατάλληλο μονοπάτι. Η βέλτιστη λύση εδώ είναι 259045. Όπως έχει ήδη αναφερθεί, λόγω του μεγάλου μήκους του προβλήματος, επιλέχθηκε να μην παρουσιαστεί το μονοπάτι που προέκυψε κατά την πειραματική διαδικασία.

Πίνακας 37: pr1002

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**
Prim	-	Time-out
Kruskal	-	Time-out
NN	315596.78	0.711
Christofides	-	Time-out

2-Opt	280797.78	7.342

Πρόβλημα nrw1379

Το μεγαλύτερο πρόβλημα που μελετήθηκε στην παρούσα διπλωματική εργασία, είναι το nrw1379.tsp, το οποίο αναζητεί λύση ανάμεσα σε 1379 διαθέσιμους κόμβους. Σύμφωνα με την βιβλιοθήκη TSPLIB, η βέλτιστη λύση σε αυτή την περίπτωση, ισούται με 56638.

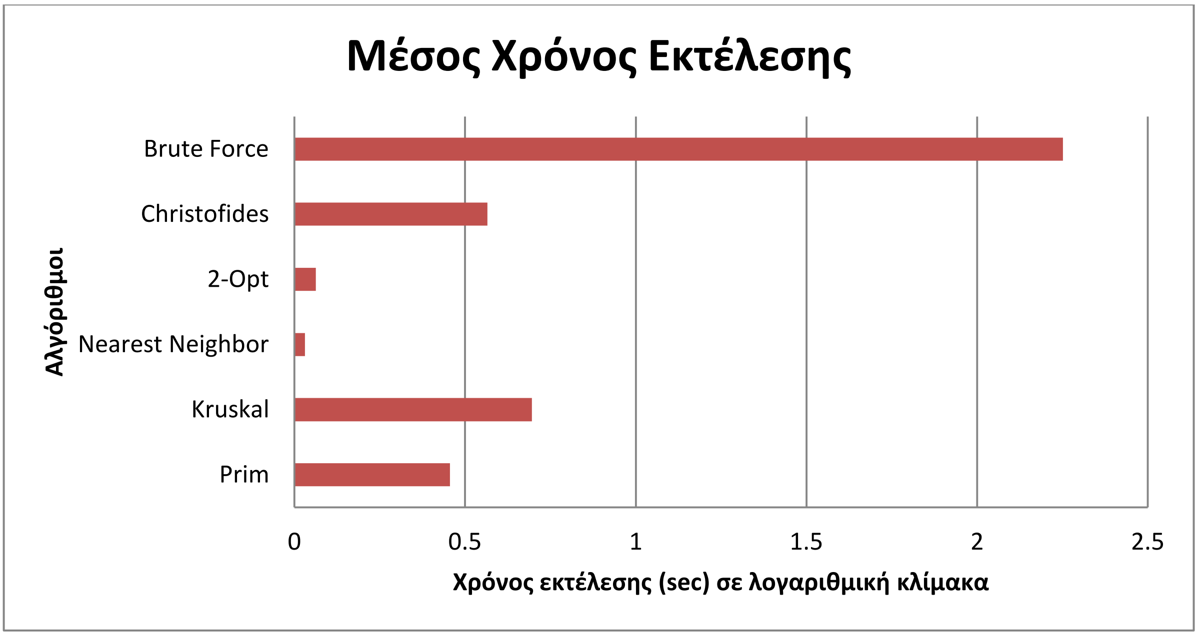
Πίνακας 38: nrw1379

Αλγόριθμος	**Λύση**	**Χρόνος Εκτέλεσης (sec)**
Prim	-	Time-out
Kruskal	-	Time-out
NN	69685.71	1.468
Christofides	-	Time-out
2-Opt	63344.08	152.17

4.4 Γραφικές Παραστάσεις Χρόνων Εκτέλεσης και Αποκλίσεων

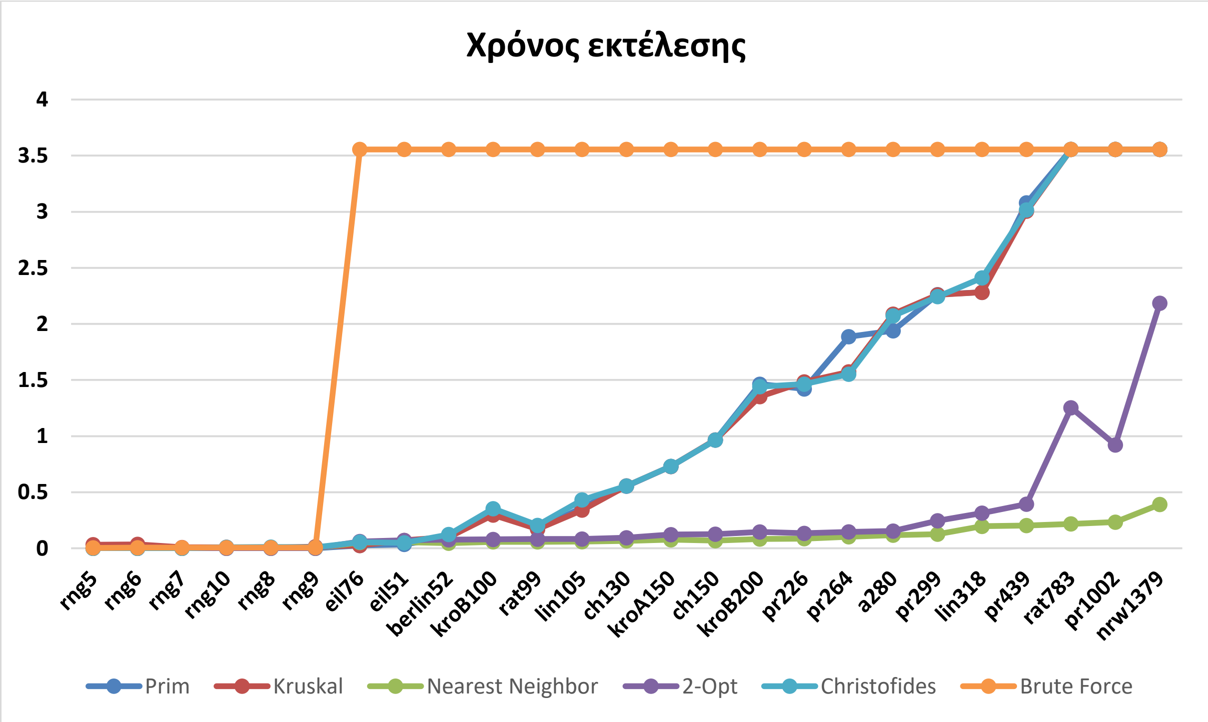
Μετά το πέρας όλων των υπολογιστικών διαδικασιών, ακολουθεί η παρουσίαση των τελικών, συγκεντρωτικών γραφημάτων, που αφορούν τους μέσους χρόνους εκτέλεσης για κάθε πρόβλημα που μελετήθηκε, καθώς επίσης και τις αποκλίσεις που προέκυψαν για τις τιμές των λύσεων, σε σύγκριση πάντοτε με τις τιμές των βέλτιστων λύσεων.

Στη Εικόνα 12, παρουσιάζονται οι μέσοι χρόνοι εκτέλεσης των αλγορίθμων που μελετήθηκαν στα συγκεκριμένα προβλήματα. Όπως είναι φανερό, ο αλγόριθμος εξαντλητικής αναζήτησης Brute Force, είναι αυτός που απαιτεί τον περισσότερο χρόνο εκτέλεσης, καθώς όπως έχει ήδη αναφερθεί, εξετάζει όλα τα πιθανά μονοπάτια, για να καταλήξει στην εύρεση της βέλτιστης λύσης.



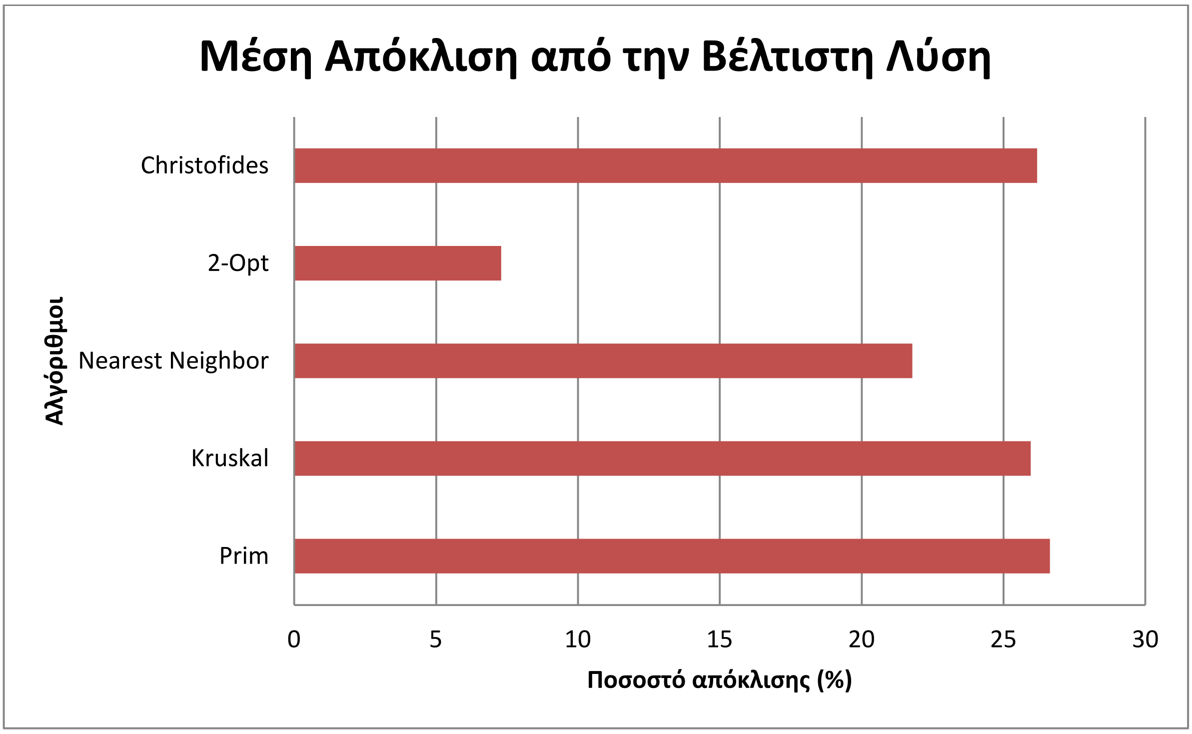
Εικόνα 12: Μέσοι Χρόνοι Εκτέλεσης Αλγορίθμων

Στην Εικόνα 13, παρουσιάζονται όλοι οι χρόνοι εκτέλεσης των αλγορίθμων, για όλα τα προβλήματα που εξετάστηκαν. Η τιμή για τον αλγόριθμο Brute Force είναι αυτή, καθώς δεν κατάφερε να βρει λύσεις για προβλήματα που ξεπερνούν τους 10 κόμβους. Οι χρόνοι εκτέλεσης των αλγορίθμων Prim, Kruskal και Christofides, είναι παρόμοιοι για τα προβλήματα που εξετάστηκαν, όπως προκύπτει από την γραφική παράσταση. Ο αλγόριθμος με την γρηγορότερη εκτέλεση είναι αυτός του Nearest Neighbor, αλλά όπως δούμε στη συνέχεια, δεν είναι μια αποδοτική υλοποίηση. Οι χρόνοι που αφορούν τον αλγόριθμο 2-Opt, είναι αρκετά ικανοποιητικοί.

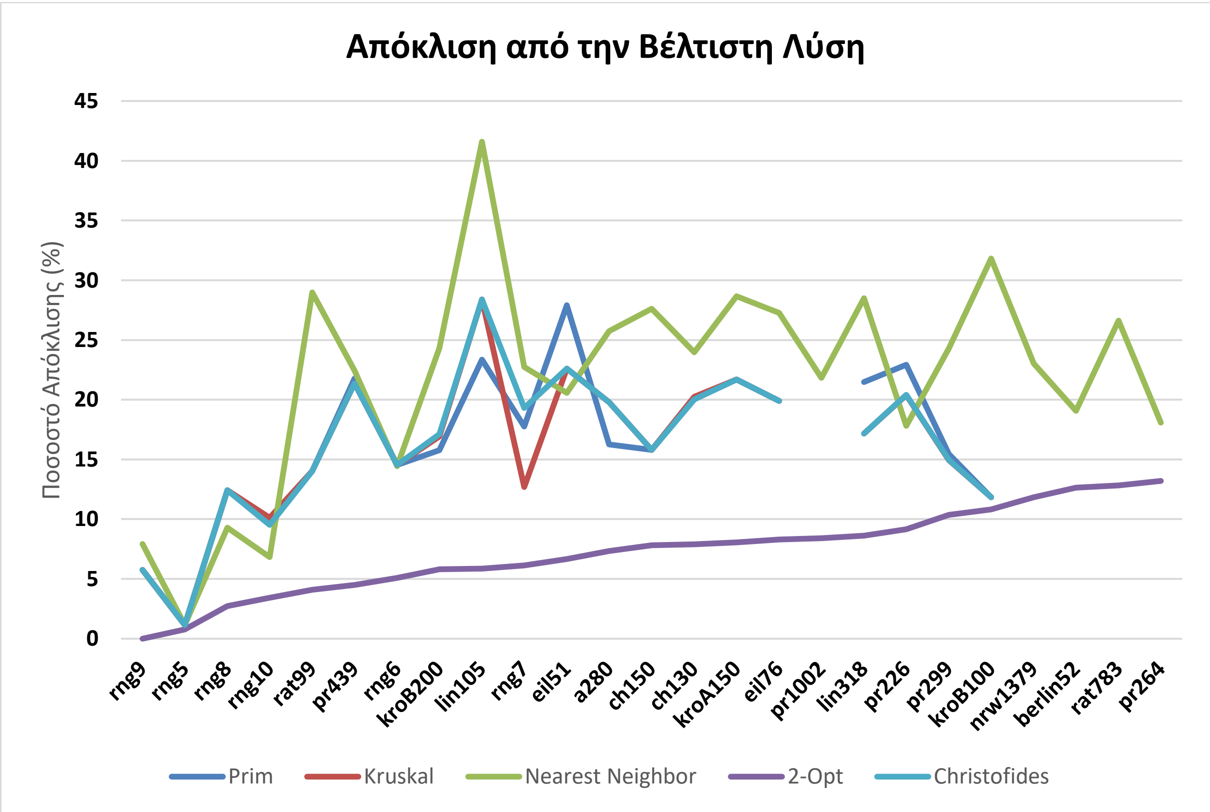


Εικόνα 13: Χρόνοι Εκτέλεσης Όλων των Προβλημάτων

Όσον αφορά τις μέσες αποκλίσεις στις τιμές των αλγορίθμων, συναρτήσει με τις βέλτιστες λύσεις που γνωρίζουμε για τα προβλήματα που εξετάστηκαν, αυτές παρουσιάζονται στην Εικόνα 14. Μικρότερο ποσοστό απόκλισης, παρουσιάζει ο αλγόριθμος επαναληπτικής βελτίωσης 2-Opt, ο οποίος έχει μέση τιμή ίση με 7%. Ακολουθούν οι υλοποιήσεις των αλγορίθμων του Christofides και του Nearest Neighbor, και τέλος, αυτές των άπληστων αλγορίθμων Prim και Kruskal. Επιπλέον, στην Εικόνα 15, παρουσιάζονται τα αποτελέσματα για τις αποκλίσεις των αλγορίθμων, για καθένα πρόβλημα που μελετήθηκε ξεχωριστά. Όπως φαίνεται στο γράφημα, την μεγαλύτερη απόκλιση φέρει ο αλγόριθμος Nearest Neighbor, καθώς φτάνει το ποσοστό του 42% για το πρόβλημα lin105. Επίσης, αξίζει να σημειωθεί πως για προβλήματα μικρών διαστάσεων, οι αλγόριθμοι των Prim, Kruskal και Nearest Neighbor, κατάφεραν να ανακαλύψουν ίδιες λύσεις. Τέλος, να σημειωθεί πως για το πρόβλημα rng9, ο αλγόριθμος 2-Opt, μετά και τις επαναληπτικές διαδικασίες που εκτελεί, κατάφερε να ανακαλύψει το βέλτιστο μονοπάτι.



Εικόνα 14: Μέση Απόκλιση από την Βέλτιστη Λύση



Εικόνα 15: Απόκλιση από την Βέλτιστη Λύση

Κεφάλαιο 5ο

Συμπεράσματα

5.1 Γενικά Συμπεράσματα

Ολοκληρώνοντας λοιπόν τη συγκεκριμένη μελέτη, υλοποιήθηκαν προγραμματιστικά οι παρακάτω αλγόριθμοι:

- Christofides
- Prim
- Kruskal
- Nearest Neighbor
- 2-Opt
- Brute Force

Οι συγκεκριμένοι αλγόριθμοι, υπολόγισαν τα κατάλληλα μονοπάτια σε πλήρη γραφήματα, για προβλήματα που προέρχονταν είτε από την βιβλιοθήκη TSPLIB, είτε από την Γεννήτρια Τυχαίων Αριθμών. Πέραν αυτών, υπολόγιζαν τα βάρη για τις αντίστοιχες διαδρομές, καθώς επίσης και τους χρόνους εκτέλεσης για κάθε αλγόριθμο ξεχωριστά.

Επιπλέον, για τους αλγορίθμους των Prim, Kruskal και Christofides, υπολογίστηκαν και οι χρόνοι εκτέλεσης για τη δημιουργία των Ελάχιστων Επικαλυπτόμενων Δέντρων.

Συμπερασματικά λοιπόν, ο πιο αποδοτικός αλγόριθμος σε αυτή τη μελέτη, είναι αυτός του 2-Opt. Οι λύσεις που προέκυψαν κατά την εκτέλεση της υλοποίησης, πέραν του ότι είχαν πολύ μικρό χρόνο εκτέλεσης, ξεκινώντας για τα μικρά προβλήματα των 5 κόμβων από μόλις 0.0011 δευτερόλεπτα, και φτάνοντας σε προβλήματα 1379 κόμβων, σε χρόνο ίσο με 152,1 δευτερολέπτα, κατάφεραν να υπολογίσουν μονοπάτια των οποίων η τιμή, βρισκόταν πολύ κοντά σε αυτή της βέλτιστης. Συγκεριμένα, οι αποκλίσεις που προέκυψαν για τις τιμές των μονοπατιών, συναρτήσει με αυτές της βέλτιστης, ήταν περίπου της τάξης του 7%, για τα προβλήματα που εξετάστηκαν.

Στη συνέχεια, ο προσεγγιστικός αλγόριθμος του Christofides, κατάφερε να υπολογίσει αρκετά καλές λύσεις. Οι χρόνοι εκτέλεσης, δε διέφεραν κατά πολύ από την υλοποίηση 2-Opt, καταφέρνοντας να βρουν λύση, για προβλήματα 200 κόμβων και πάνω, σε περίπου 26 δευτερόλεπτα. Όσον αφορά τις αποκλίσεις στις τιμές των μονοπατιών, αυτές κυμαίνονται σε ποσοστό ως 28%.

Για τον αλγόριθμο Prim, τα αποτελέσματα που προκύπτουν από τον υπολογισμό των κατάλληλων μονοπατιών, εξέρχονται επίσης σε δευτερόλεπτα. Πιο συγκεκριμένα, για προβλήματα που ξεκινούν από 100 κόμβους, ο αλγόριθμος καταφέρνει να βρει λύση σε μόλις 1.2 δευτερόλεπτα. Φυσικά, αυξάνοντας τα μεγέθη των προβλήματων, αυξάνεται και ο χρόνος εκτέλεσης, ο οποίος φτάνει μέχρι τα 1201 δευτερόλεπτα ~ 20 λεπτά, για ένα πρόβλημα 439 κόμβων. Οι αποκλίσεις που προκύπτουν εδώ για τις λύσεις των διαδρομών, είναι της τάξης του 16%, κάτι που την καθιστά, μια καλή υλοποίηση.

Αντίστοιχη συμπεριφορά έχει και ο αλγόριθμος Kruskal. Για να ανακαλύψει το κατάλληλο μονοπάτι κατά την εκτέλεση της υλοποίησης, οι χρόνοι που απαιτούνται για πρόβλημα 300 κόμβων, είναι περίπου 190 δευτερόλεπτα ~ 3 λεπτά. Όσον αφορά τις αποκλίσεις, οι τιμές τους ξεκινούν από μόλις 1.13%, για προβλήματα 5 κορυφών, και φτάνουν ως 28%, για προβλήματα μεγαλύτερων διαστάσεων.

Όπως ήταν αναμενόμενο, ο άπληστος αλγόριθμος Nearest Neighbor, είναι ένας από τους λιγότερο αποδοτικούς αλγορίθμους, καθώς κάνει πάντοτε την επιλογή η οποία φαίνεται η καλύτερη δυνατή τη δεδομένη στιγμή, ωστόσο δεν οδηγεί σε καλές και αποδοτικές λύσεις. Παρόλο που ο χρόνος εκτέλεσης του αλγορίθμου είναι σχετικά μικρός, οι αποκλίσεις που προκύπτουν εδώ σε σχέση πάντοτε με την βέλτιστη, είναι αρκετά μεγάλες. Πιο συγκεκριμένα, για το πρόβλημα lin105, παρατηρείται η μεγαλύτερη απόκλιση, η οποία είναι ίση με 41.61%.

Για τον αλγόριθμο Brute Force, δεν μπορούμε να έχουμε μια πλήρη εικόνα, καθώς μελετήθηκαν μόνο τα προβλήματα που προήλθαν από την Γεννήτρια Τυχαίων Αριθμών και τα οποία είναι αρκετά μικρά σε μέγεθος. Όπως είναι γνωστό, η υλοποίηση αυτή θα υπολογίσει την βέλτιστη λύση, ωστόσο όμως, θα πρέπει να υπολογίσει και όλα τα πιθανά μονοπάτια σε ένα γράφημα. Για ένα πρόβλημα λοιπόν 10 κορυφών, θα πρέπει να υπολογίσει 10! = 36228800 πιθανά μονοπάτια, κάτι που καθιστά αδύνατη την εκτέλεσή της σε μικρό χρονικό διάστημα.

5.2 Μελλοντικές Εξελίξεις

Το έργο της τρέχουσας διπλωματικής εργασίας, δε σταματά εδώ. Παρόλο που κατάφεραν να υπολογιστούν αποτελέσματα για προβλήματα μεγάλων διαστάσεων και να προκύψουν τα κατάλληλα συμπεράσματα, σίγουρα υπάρχει χώρος για μελλοντικές βελτιώσεις. Στόχος είναι εξεταστούν και να μελετηθούν επιπλέον δομές δεδομένων, που δεν εξετάστηκαν στην παρούσα μελέτη, έτσι ώστε να μειωθούν όσο το δυνατόν περισσότερο οι χρόνοι εκτέλεσης των αλγορίθμων.

Επιπλέον, στοχεύεται να εξεταστούν περισσότερες υλοποιήσεις αλγορίθμων πάνω στο Πρόβλημα του Πλανόδιου Πωλητή, από διάφορες κατηγορίες, έτσι ώστε να προκύψει μια άκομη πιο ολοκληρωμένη εικόνα πάνω στο πρόβλημα. Θα μπορούσε για παράδειγμα να μελετηθεί η συμπεριφορά του αλγορίθμου 3-Opt, όπου θα πραγματοποιείται εναλλαγή ανάμεσα σε 3 έγκυρα μονοπάτια ακμών που διασταυρώνονται, με σκοπό την εύρεση μιας καλύτερης δυνατής λύσης.

Τέλος, είναι γνωστό πως για το TSP, υπάρχουν προβλήματα πολύ μεγαλύτερων διαστάσεων, που φτάνουν έως και τους 16.000 κόμβους. Μια μελλοντική εξέλιξη, αποτελεί η μελέτη και αξιολόγηση τέτοιων περιπτώσεων, κάτι που δεν στάθηκε δυνατόν στην παρούσα προσπάθεια, καθώς οι χρόνοι εκτέλεσης για τους αλγορίθμους ήταν υπερβολικά μεγάλοι.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: a computational study*. Princeton university press

Arora, K., Agarwal, S., & Tanwar, R. (2016). Solving TSP using genetic algorithm and nearest neighbour algorithm and their comparison. *International Journal of Scientific & Engineering Research*, *7(1)*, 1014-1018.

Biswas, B., Basuli, K., Naskar, S., Chakraborti, S., & Sarma, S. S. (2012). A combinatorial algorithm to generate all spanning trees of a weighted graph in order of increasing cost. *arXiv preprint arXiv:1209.4206*.

Christofides, N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem* (No. RR-388). Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1959). On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, *7*(1), 58-66.

Emami Taba, M. S. (2010). *Solving Traveling Salesman Problem with a non-complete graph*. Master's thesis, University of Waterloo.

Flood, M. M. (1956). The traveling-salesman problem. *Operations research*, *4*(1), 61-75.

Grötschel, M., Jünger, M., & Reinelt, G. (1984). A cutting plane algorithm for the linear ordering problem. *Operations research*, *32*(6), 1195-1220.

Hoogeveen, J. A. (1991). Analysis of Christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, *10*(5), 291-295.

Hurkens, C. A., & Woeginger, G. J. (2004). On the nearest neighbor rule for the traveling salesman problem. *Operations Research Letters*, *32*(1), 1-4.

Letchford, A. N., Nasiri, S. D., & Theis, D. O. (2013). Compact formulations of the Steiner traveling salesman problem and related problems. *European Journal of Operational Research*, *228*(1), 83-92.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, *7*(4), 326-329.

Nilsson, C. (2003). Heuristics for the traveling salesman problem. *Linkoping University*, *38*, 00085-9.

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing*, *3*(4), 376-384.